The Pennsylvania State University

The Graduate School

Department of Computer Science and Engineering

DESIGNING ENERGY-EFFICIENT AND RELIABLE CACHES AND INTERCONNECTS

A Thesis in

Computer Science and Engineering

by

Lin Li

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2005

The thesis of Lin Li was reviewed and approved* by the following:

Mary Jane Irwin
A. Robert Noll Chair of Engineering
Professor of Computer Science and Engineering
Thesis Co-Adviser
Co-Chair of Committee

Vijaykrishnan Narayanan
Associate Professor of Computer Science and Engineering
Thesis Co-Adviser
Co-Chair of Committee

Mahmut Kandemir
Associate Professor of Computer Science and Engineering

Yuan Xie
Assistant Professor of Computer Science and Engineering

Kenan Unlu
Professor of Mechanical and Nuclear Engineering

Raj Acharya
Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

The minimum feature size of VLSI technology has shrunk exponentially in the past three decades and this trend is expected to continue in the near future. This phenomenon has resulted in an increase in the number of transistors integrated onto a single chip, a decrease in supply voltages and threshold voltages, and an increase in clock frequencies for each generation. The scaling of these design parameters impacts digital circuits and systems design in two important ways: a dramatic increase in energy consumption and a deterioration of reliability.

This thesis proposes design methodologies and techniques at the micro-architecture level for improving energy-efficiency and reliability in cache memories and on-chip interconnects.

First, architectural techniques are proposed to reduce both leakage and dynamic energy consumption in cache memories. State-preserving and state-destroying leakage control mechanisms are integrated into L2 cache to reduce leakage energy consumption due to the data duplication across different levels of cache hierarchy. A crossbar-connected cache configuration is proposed for on-chip multiprocessor systems. The dynamic energy consumption in L1 cache is reduced due to sharing a single, banked cache among processors.

Next, the impact of crosstalk noise on the reliability of on-chip interconnects is investigated, and a crosstalk-aware interconnect is proposed. A crosstalk analyzer circuit is designed to be incorporated into the sender side of the bus to estimate the delay based

on the data pattern to be transmitted. Then, a variable cycle transmission mechanism is implemented to use a faster clock and dynamically control the number of cycles required for transmission.

Finally, the interaction between energy consumption and reliability is studied. Based on the soft error injection implemented in the cache system of SimpleScalar simulator, the impact of two architectural-level leakage reduction approaches, drowsy cache and cache decay, on the data reliability are evaluated. Then, providing data reliability in an energy-efficient fashion is investigated. An adaptive error coding scheme is proposed that treats dirty and clean data cache blocks differently. Furthermore, an early-write-back scheme is introduced to enhance the ability to use a less powerful error protection scheme for a longer time without sacrificing reliability. Adaptive error protection scheme is also applied to on-chip interconnects for energy efficiency, where the type of error detection coding scheme is modulated dynamically based on the dynamic variations observed in noise behavior.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I want to express my sincere gratitude to my thesis advisers, Dr. Mary Jane Irwin and Dr. Vijaykrishnan Narayanan, for their guidance, encouragement, and support in the past five years of my Ph.D. study.

Dr. Irwin is one of the greatest researchers I have ever met. She has showed me the whole picture of research on digital circuits and systems, guided me in entering and exploring this exciting area, and helped me decide the topic of my research work. Despite her busy schedule, she has always been willing to help me when I have had difficulties and confusions in my research. Dr. Irwin is also a great educator. Having been her teaching assistant in my first semester at Penn State, I still remember clearly how she consistently provided high quality lectures, inspired students' desire to learn, and personally attended to the needs of each student. What I learned from her will benefit me for the rest of my life.

Dr. Vijaykrishnan Narayanan has spent incalculable effort and time working with me on my research projects. I have learned a great deal from hours of inspiring and enlightening discussion with him. Over the course of our interaction, his devotion and persistence in his research, intelligent and insightful thoughts, and diligent work have been an inspiration. Academics aside, Dr. Narayanan has also given me invaluable friendly advice and lent many a helpful hand on my job hunt. I sincerely thank him for his patience in waiting for me to mature in my research field.

Also, special thanks go to Dr. Mahmut Kandemir and Dr. Yuan Xie. I have gotten a lot out of my collaboration with them. I also want to thank Dr. Kenan Unlu and Dr. Richard Brooks for their valuable advice and constructive comments on this thesis.

Next, I am grateful to all my fellow colleagues in MDL group at Penn State. Specifically, I would like to thank the following as I am unable to list all the wonderful people from the group. During my time working in this group, I have enjoyed pleasant and fruitful collaboration with Vijay Degalahal, Yuh-Fang Tsai, Suresh Srinivasan, and Ismail Kadayif. Moreover, former MDL members Hyun-Suk Kim, Soontae Kim, and Byung Tae Kang gave me essential help when I first joined the MDL group. Guangyu Chen, Guilin Chen, and Feihui Li provided me knowledge in the compiler area. Greg Link and Theo Theocharides organized many interesting activities to make MDL group a happy and vigorous family. I am lucky to have been a part of this wonderful team.

Of course, the completion of this thesis would not have been possible without the support of my family. My parents and my younger brother have always provided me their selfless love and unparalleled support. Likewise, my in-laws welcomed me into their family and have since consistently stood by me with encouragement. Last but not least, I sincerely thank my dear wife, Xingjie Lu. She accompanies me in my life at Penn State, and we share every happy and bitter moment of our lives. Her love and encouragement are my fundamental motivations to overcome each and every difficulty. Doubtless, her belief in me has enabled me to move forward when I have been frustrated or depressed. I dedicate all my achievement to my wife and my family.

Chapter 1

Introduction

In the past three decades, the rapid development of VLSI technology has led to exponential shrinking of the minimum feature sizes [63]. This trend is expected to continue in the near future and result in a significant increase in the number of transistors integrated onto a single chip. Moore's Law shows that the average number of transistors on a chip doubles every 18 months. Other consequences of the shrinking feature size include continuous decrease of supply voltage and threshold voltage, higher clock frequency, and smaller nodal capacitance.

The scaling of these design parameters impacts digital circuits and systems design in two important ways: a dramatic increase in energy consumption and a deterioration of reliability. These two impacts are significant not only for high-performance servers, but also for embedded systems, mobile devices, and ubiquitous computing applications. As a result, a considerable amount of research on reducing energy consumption and improving reliability of VLSI technology has been done. This thesis focuses on design methodologies and techniques at the micro-architecture level for improving energy-efficiency and reliability in cache memories and on-chip interconnects.

[1]Part of the material in this thesis has been published (or expected to be published) in [25, 50, 51, 52, 53, 54, 55].

1.1 Targeted Problems

1.1.1 Energy Consumption

Energy consumption of transistors consists of two divisions: dynamic energy consumption E_{dyn} and leakage energy consumption E_{leak}. Dynamic energy consumption is due to charging and discharging load capacitances when transistors are switching, which is defined by Equation 1.1:

$$E_{dyn} = C_L \times V_{DD}^2 \tag{1.1}$$

where C_L is the load capacitance of switching nodes, and V_{DD} is the supply voltage. Leakage energy consumption is due to the current that flows through the transistors in the absence of any switching activity, which is defined by Equation 1.2:

$$E_{leak} = V_{DD} \times I_{leak} \times T \tag{1.2}$$

where V_{DD} is the supply voltage, I_{leak} is the total leakage current, and T is the time. Since high-k dielectrics are projected to be used to reduce gate-oxide leakage dramatically, only subthreshold leakage energy consumption is modeled in this thesis.

With the scaling of VLSI technology, decreasing supply voltage helps to reduce both dynamic and leakage energy consumptions, especially bringing a quadratic effect on dynamic energy consumption. On the other hand, decreasing threshold voltage aggravates the leakage energy consumption since subthreshold leakage current increases exponentially as threshold voltage decreases [17]. Higher clock frequency results in more switching activities of transistors in the same time period and, hence, increases dynamic

energy consumption per time unit. The increase of the number of transistors enables more and more complex components to be integrated onto a single chip and, consequently, deteriorates both chip dynamic and leakage energy consumption.

Therefore, both dynamic and leakage energy consumption of microprocessors are increasing dramatically. For example, based on Intel® datasheets [3, 4], the power dissipation is 33 Watts for Pentium® III Processor at 1.0 GHz and is 122 Watts for Itanium® 2 Processor at 1.60 GHz.

In addition, leakage energy consumption increases more sharply than dynamic energy consumption. Chip leakage power dissipation is expected to increase by five times for each technology generation in the future. This trend will result in leakage power becoming the dominant part of the chip power budget for 0.10 micron technology and below [18]. Researchers in [44] predict that leakage energy consumption will constitute 30%-40% of total energy consumption within three generations.

Increasing energy consumption affects the size and cost of power supply of computer systems. Today, most data centers have power capacities of 40 to 70 Watts per square foot, but they will have to support 500 Watts per square foot in the next few years. In addition, increasing energy consumption also affects the cooling system and device packaging of processors, which leads to two important problems: thermal management and heat removal. Finally, it weakens the efficiency of battery-powered portable electronic systems, such as laptops, mobile devices, and embedded systems. As maximization of battery life becomes a critical goal, designing energy-efficient systems is becoming one of the most interesting and challenging research areas in both academia and industry.

While dynamic energy will still remain a concern for components that are exercised and switched often, leakage energy is of particular concern in the bulky memory structures. This is due to three reasons: sub-threshold leakage current continues to increase; leakage energy increases with the effective number of transistors in the circuit; and a large transistor budget is allocated for on-chip memories in current processors. For example, Intel® Itanium® 2 Processor has 16KB L1 instruction cache, 16KB L1 data cache, 256KB L2 cache, and 3MB L3 cache. L3 cache occupies 180 million transistors, as well as the core occupies 40 million transistors [61]. In [9], it is shown that L1 cache alone consumes over 40% of total dynamic power budget. In addition, leakage energy occupies 30% of L1 cache energy consumption and 80% L2 cache energy consumption [8]. In [44], researchers project that leakage energy consumption will achieve more than 70% of total energy consumption in caches.

1.1.2 Reliability

Besides the dramatic increase in energy consumption, the other significant influence of scaling VLSI technology is the deterioration of reliability in digital circuits and systems. As we know, the reliability of digital circuits is affected by a number of different sources of noise, which include the internal noises such as power supply noise, crosstalk noise, and inter-symbol interference, as well as external noises such as electromagnetic interference, thermal noise, slot noise, and soft errors induced by the striking of high-energy particles [20, 23, 24, 73]. These noises cause signals to deviate from their intended values and bring severe errors in data integrity.

The relationship between bit error rate (BER) and noise is modeled in [32], which models the different noise sources as a single Gaussian noise source. Specifically, assuming a noise voltage V_N distributed according to a normal distribution with a variance of σ^2 and that a noise voltage greater than $V_{dd}/2$ causes an error, the probability of error happening on a signal is denoted as bit error rate (BER) ϵ and can be given by Equation 1.3:

$$\epsilon = Q(\frac{V_{dd}}{2\sigma}), \tag{1.3}$$

where $Q(x)$ is the Gaussian distribution and given by Equation 1.4:

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy \tag{1.4}$$

This model has also been adopted in research works [15, 85]. Note that the probability of error is sensitive to both the supply voltage (V_{dd}) and the variance of noise voltage.

With technology scaling and the consequences of lower supply voltages, reduced capacitive values of the circuit nodes, and more closely routed wires, the noise margin of signals is reducing dramatically. Further, the variation of noise voltage is also increasing significantly. The continued scaling in process technologies makes it imperative to consider reliability as a first-class design constraint. Even the International Technology Roadmap for Semiconductors (ITRS) acknowledges reliability as a cross-cutting problem concerning both designers and test engineers.

Specifically, this thesis focuses on two noise sources, crosstalk noise and soft errors. With dramatic scaling in feature size, the propagation delay of long on-chip interconnect is becoming more significant than the delay of gates in deep sub-micron technology.

Crosstalk between signals caused by increased capacitive coupling is considered one of the major problems affecting the timing of signals and, consequently, the cycle time must be increased. This, in turn, can lead to performance degradation and functional failures [42]. Therefore, alleviating the impact of crosstalk on the propagation delay of interconnect is very important in high performance system design.

The second noise source targeted in this thesis is soft errors or transient errors that are circuit errors caused by excess charge carriers induced primarily by external radiations. The primary source of radiations that induce soft errors can be classified as alpha particles from the packaging materials, high energy neutrons from cosmic radiations, and the interaction of cosmic ray thermal neutron with the ^{10}B isotope of Boron [14]. While the elimination of ^{10}B using new process technologies and the removal of impurities that cause alpha particle radiation have helped in reducing the soft error rates, the smaller nodal capacitances and supply voltages have resulted in higher soft errors. The smaller charge is a particular concern because particles of lower energy occur far more frequently than particles of higher energy in atmospheric radiation [92]. Consequently, more particles can cause soft errors as CMOS device sizes decrease. This has raised reliability concerns due to the increased susceptibility to soft errors [31, 39, 71].

While these errors cause an upset event, the circuit itself is not damaged. In memory, these can cause a particular node to charge or discharge and, thus, cause a bit flip. This is particularly true for SRAM cells used in caches [31].

For a soft error to occur at a specific node in a circuit, the collected charge Q at that particular node should be more than $Q_{critical}$. If the charge generated by a particle strike at a node generates a charge that is more than the $Q_{critical}$, the pulse so generated

is latched on, and results in a bit flip. This concept of critical charge is generally used to estimate the sensitivity of Soft Error Rate (SER). In [31], a method to estimate the SER in CMOS SRAM circuits was developed. In this model an exponential dependence of SER on critical charge was shown as:

$$SER \propto N_{flux} * CS * exp(-\frac{Q_{critical}}{Q_s}) \tag{1.5}$$

where N_{flux} is the intensity of the Neutron Flux, CS is the area of the cross section of the node, and Q_s is the charge collection efficiency (it is strongly dependent on doping). $Q_{critical}$ is proportional to the node capacitance and the supply voltage.

1.2 Related Works

1.2.1 Design for Energy Efficiency

Many techniques have been proposed in the past to reduce cache energy consumption. Among these are partitioning large caches into smaller structures to reduce the dynamic energy [37, 46] and the use of a memory hierarchy that attempts to capture the most accesses in the smallest size memory. By accessing the tag and data array in series, Alpha 21164's L2 cache [18] can access the selected cache bank for energy efficiency. In [36, 68], way-prediction is used to reduce energy consumption of set-associative caches. Selective cache ways [9] varies the number of ways for different application requirements. In [47], a small filter cache is placed prior to L1 cache to reduce energy consumption. Dynamic Zero Compression [82] employs single-bit access for zero-valued byte in the cache to reduce energy consumption.

There have been several efforts [8, 13, 28, 35] at the architectural level to reduce the cache leakage energy when it is idle. Some of these techniques focus on reducing leakage during idle cycles of the component by turning off the supply voltage. One such scheme, gated-Vdd, was integrated into the architecture of caches [88] to dynamically shutdown portions of the cache. This technique was applied at a cache block granularity in [41] and used in conjunction with software to remove dead objects in [19].

However, all these techniques assume that the state (contents) of the supply-gated cache memory is lost. While totally eliminating the supply voltage results in the state of the cache memory being lost, it is possible to apply a state-preserving leakage optimization technique if a small supply voltage is maintained to the memory cell. There are many alternate implementations that have been recently proposed at the circuit level to achieve such a state-preserving leakage control mechanism [8, 65, 80].

As an abstraction of these techniques, the choice between the state-preserving and state-destroying techniques depends on the relative overhead of the additional leakage required to maintain the state as opposed to the cost of restoring the lost state from other levels of the memory hierarchy.

An important requirement to reduce leakage energy using either a state-preserving or a state-destroying leakage control mechanism is the ability to identify unused resources (or data contained in them). In [88], the cache size is reduced (or increased) dynamically to optimize the utility of the cache. In [41], the cache block is supply-gated if it has not been accessed for a period of time. In [91], hardware tracks the hypothetical miss rate and the real miss rate by keeping a tag line active when deactiving a cache line, and the turn-off interval can, subsequently, be dynamically adjusted based on such information. In [28,

44], dynamic supply voltage scaling is used to reduce the leakage in the unused portions of the memory. In contrast to the other schemes, it also preserves data when in low leakage mode. The usefulness and practicality of such state-preserving voltage scaling schemes for embedded power-optimized memories is demonstrated in [69]. Comparatively, the focus in [34] is on reducing bitline leakage power using leakage-biased bitlines. The technique turns off precharging transistors of unused subbanks to reduce bitline leakage, and actual bitline precharging is delayed until the subbank is accessed.

1.2.2 Design for Reliability

There are many schemes employed at different levels to reduce the impact of crosstalk. Net ordering and buffer insertion are employed in the physical design of interconnects [10, 56]. Shielding of the wires and increasing the inter-wire spacing are other options explored for reducing the impact of crosstalk [11]. Another approach transforms the actual data to be transmitted to a coded form in order to reduce crosstalk [81]. Active shielding method employs two shielding wires on both sides of the target wire and keeps the same transition direction as the target wire for fast propagation [40]. However, the above schemes (shielding, active shielding, increased spacing, and the coding schemes) incur a significant area overhead.

Bus wire twisting [30, 90] is another scheme to reduce the impact of crosstalk among wires. But, when twisting is used, the wire length and, hence, the corresponding total resistance and capacitance are larger than when utilizing the original interconnect structure [90]. Also, this scheme incurs more complicated techniques and cost in the

manufacturing process since the reduction of worst-case delay by wire twisting [30] is less than those by bus shielding and spacing [11].

A variety of soft error-tolerant techniques have been proposed, ranging from special radiation-hardened circuit designs to localized error detection and correction to architectural-level redundancy [12, 59, 62, 64, 67]. However, these approaches usually introduce some sort of penalty in performance, power, die size, and design time.

Different designs of low power cache memories can potentially have different immunity to soft errors due to supply voltage changes and circuit structure [26]. These differences in soft error vulnerability are important as they influence the complexity of error detection and correction circuitry employed in on-chip memories.

There are few works that consider both energy consumption and reliability at the same time. A good overview of the noise problems in submicron CMOS is presented in [72]. In addition, this work presents a noise tolerance scheme for achieving energy and performance efficiency in the presence of noise. In [32], the authors present lower bounds on energy consumption in the presence of noise and show the effectiveness of coding schemes such as Hamming codes in reducing the energy consumption when communicating in noisy buses. Hedge and Shanbhag [33] explore the possibility of handling errors by compensation through DSP algorithms. Bertozzi et al. [15], compare the energy behavior of different error correction and detection schemes. Their results show that error detection methods combined with retransmission are more energy efficient than error correction methods under the same reliability constraints. The scheme in [85] exploits the fact that a smaller voltage (hence, a smaller noise margin) would be sufficient for transmission in a less noisy phase of execution. Therefore, they increase (decrease)

the voltage when the noise increases above (decreases below) a threshold point. There have been several techniques that attempt to reduce the energy consumption resulting in buses due to the presence of noise. An example of such optimizations is the work by Kim et al. [43] that optimizes the coupling power consumed by interconnects due to the presence of cross talk noise.

1.3 Contributions

The contributions of this thesis consist of three parts: (1) Reducing both leakage and dynamic energy consumption in cache memories; (2) Reducing the impacts of crosstalk noise on the reliability of on-chip interconnects; and, (3) Investigating the interaction between energy consumption and reliability.

1.3.1 Reducing Cache Energy Consumption

First, architectural strategies to reduce cache leakage energy consumption are proposed by exploiting the data duplication present in an on-chip L1-L2 cache hierarchy. State-preserving and state-destroying leakage control mechanisms are employed to L2 blocks when their data also exist in L1 cache. The effectiveness of the proposed techniques is demonstrated through cycle-accurate simulation using a set of MediaBench and SPEC CINT2000 benchmarks.

Second, the proposed schemes are compared with cache decay policy, a finite state machine (FSM) based strategy that turns off cache subblocks when they are idle for a sufficiently long period of time. This comparison indicates that one of the proposed

schemes is competitive in terms of energy savings. Furthermore, it is shown how both techniques can be applied in conjunction to provide additional energy gains.

Third, a cache configuration, referred to as the *crossbar-connected cache* or *CCC* for short, is proposed, which tries to combine the advantages of single shared L1 cache configuration and private L1 cache configuration without their drawbacks. Specifically, the shared cache is divided into multiple banks, and an N×M crossbar is used to connect N processors and M banks. In this way, the data duplication problem in private L1 cache configuration is removed, thus, rendering sophisticated consistency mechanisms unnecessary. In addition, this solution is scalable, and each bank can be simple in architecture. The experimental results obtained through cycle-accurate simulations indicate that the energy benefits of proposed cache architecture range from 9% to 26% with respect to the private cache option. These savings come at the expense of a small degradation in performance.

1.3.2 Reducing the Impacts of Crosstalk Noise

A crosstalk aware interconnect is designed by adding a crosstalk analyzer circuit to the sender side of the bus. The crosstalk analyzer determines the delay associated with the transition pattern comparing the previous data and current data to be transmitted. The output of the analyzer is used to dynamically control the number of cycles required for transmission in order to improve performance. Experimental results show that this approach, using a faster clock with variable cycles for data transmission, provides significant performance gains as compared to using a single clock based on the worst-case crosstalk. Comparing the proposed scheme with other crosstalk reduction

techniques shows that the proposed technique is an area-efficient mechanism to enhance performance.

1.3.3 Interaction between Energy Consumption and Reliability

First, soft error injection is implemented in the cache system of SimpleScalar simulator, and the impact of two architectural-level leakage reduction approaches, drowsy cache and cache decay, on the data reliability is investigated. Experimental results show that the drowsy cache, while saving significant leakage energy, also incurs a significant increase in soft errors as compared to the original cache configuration. In contrast, the cache decay can reduce both the leakage energy and the amount of effective soft errors. Consequently, using the decay approach may be more effective when considering both reliability and energy optimizations together.

Second, providing data reliability in an energy-efficient fashion in the presence of soft-errors is investigated. In contrast to current commercial caches that treat and protect all data using the same error detection/correction mechanism, an adaptive error coding scheme is proposed, which treats dirty and clean data cache blocks differently. Furthermore, an early-write-back scheme is presented, which enhances the ability to use a less powerful error protection scheme for a longer time without sacrificing reliability. Experimental results show that the proposed schemes when used in conjunction can reduce dynamic energy of error protection components in L1 data cache by 11% on average without impacting the performance or reliability.

Third, an architectural level transmission error monitoring system is designed. It employs an extra bus line to track the variation in noise sources. This extra line

is operated at a lower voltage so that it is more sensitive to the variations in noise behavior. Based on the change in noise behavior detected by the monitoring system, the strength of the error protection scheme is suitably adapted. The effectiveness of the proposed adaptive strategy is evaluated by using a multiprocessor simulator and two different noise profiles. The results show that the proposed dynamic strategy reduces the bus communication energy consumption by approximately 10% over the most powerful detection scheme while providing an error detection rate comparable to that of the latter.

1.4 Thesis Organization

The rest of this thesis is organized as following. Chapter 2 presents the techniques to reduce both leakage and dynamic energy consumption in cache memories. Chapter 3 describes the technique to reduce the impact of crosstalk noise. Chapter 4 investigates the interaction between energy consumption and reliability. Chapter 5 gives conclusions and describes the directions of future work.

Chapter 2

Reducing Cache Energy Consumption

2.1 Leakage Energy Management in Cache Hierarchies

In an on-chip L1-L2 cache hierarchy (which consists of an L1 instruction cache, an L1 data cache, and a unified L2 cache), the data present in L1 is also contained in L2, when destructive interference in L2 does not happen between instruction and data. This data duplication in cache hierarchy is exploited in this section for reducing leakage energy consumption.

The basic idea is to transition the cache subblock in L2 to a standby leakage mode when its data is moved to L1. Therefore, leakage energy is saved by keeping only one active copy of the data. Two-level exclusive cache schemes have also been proposed for improving performance [38]. Technique in this section mimics exclusion by putting a duplicated copy to sleep mode.

2.1.1 Technology/Circuit Support for Leakage Control

Many circuit techniques have focused on limiting the device leakage. One such technique is the use of high-V_t transistors on non-critical paths of the circuit [18]. This technique can help to reduce leakage in these paths when they are used or idle. Another technique that can be employed is to introduce a power-switch between the power supply and the leaking circuit. The PMOS switch provides a virtual supply voltage to the leaking

circuit. The switch is turned off when the circuit is idle to cut off the supply voltage, eliminating leakage energy. The third technique is dynamic scaling of the supply voltages of the circuit. Due to short-channel effects, the leakage current reduces significantly when supply voltage reduces.

In this section, the dynamic scaling of the supply voltages is selected as the leakage reduction technique at the circuit level. Whether the data in memory cell is retained or not depends on the choice of the supply voltage. In normal mode, the supply voltage of memory cell is 1.0V. In low leakage control mode, 0.3V supply voltage is used for state-preserving mechanism and 0V supply voltage for state-destroying mechanism.

There are additional constraints imposed by the state-preserving leakage control. For the state-preserving leakage control, it is also important to prevent memory cells that maintain their state using a small voltage from losing state when connected with the bit lines. This can be avoided by ensuring that the wordline signal that controls the connection between the memory cell and the bitlines is suppressed until the memory cells of a cache line in state-preserving leakage control mode recover to a normal supply voltage. This logic can easily be incorporated in the row decoders.

A final consideration in the state-preserving mode is that the circuit operating at a lower supply voltage is more susceptible to bit flipping due to soft errors from alpha particle strikes [60]. These soft errors for memories are presently addressed using additional error correction bits. However, as the possibility of such errors increases with reduction in voltage, a more detailed analysis is required to accurately account for its effect. There are interesting tradeoffs offered by the amount of leakage energy that can

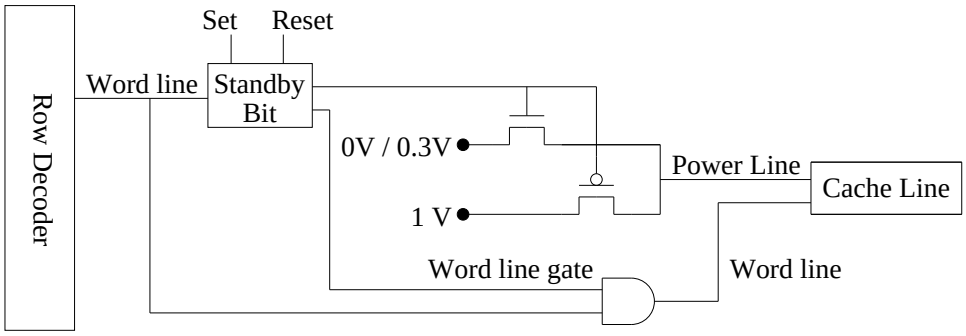

Fig. 2.1. Leakage control circuitry.

be saved and the internal node voltage levels (that in turn influence susceptibility to errors).

A similar circuit to that proposed in [28] is used except for the change in supply voltage for state-preserving and state-destroying. For each cache line, a standby bit is added to control the selection of voltage supply of the cell as shown in Figure 2.1. For both state-preserving and state-destroying, the standby bit of cache blocks is turned on during low leakage control mode and turn it off when the cache subblock needs to be accessed and return to the normal mode. The logic for setting and resetting the standby bit can be incorporated in the cache refill logic of the controller. The conditions for setting and resetting are dependent on the optimization strategy and are explained in the next subsection.

In this work, it is assumed that leakage in the state-preserving mode is 10% of original leakage in the normal mode. On the other hand, these is no leakage energy consumption in the state-destroying mode. These assumption are based on trends obtained from HSPICE circuit simulation (using Berkeley predictive model [1]) using a 1.0V, 0.07

micron technology with a normal V_{Tn}/V_{Tp} of 200mV/-220mV and a high V_{Tn}/V_{Tp} of 400mV/-420mV. Simulation was done using a temperature of 85 degrees C.

There is an additional area penalty associated with the leakage control circuitry. The estimates indicate that the power switch, word line gating logic, and the standby control increase the area of the cache by 3%. This can potentially increase the length of the wires and have a small impact on the dynamic energy. The leakage energy saving parameter is varied in experiments later to accommodate anticipated variations due to operating condition and circuit parameter variations.

There is additional dynamic energy consumed in switching these transistors that is reflected as *control energy* (also called *control overhead*) in experiments. Further, cache line in low leakage control mode must be switched to normal mode (voltage settling to normal 1V) before access is permitted. This can be achieved by using the wordline trigger to reset the standby bit in 1 clock cycle. The voltage settling time of 1 cycle is a key difference from the circuit used in [51], which had a larger penalty. The switching times for both the circuits were validated through HSPICE circuit simulation.

If a cache block is not accessed after being placed into low-power mode (using a state-preserving or state-destroying strategy), it can be expected that the state-destroying mode would save more leakage energy than the state-preserving mode as the latter still consumes 10% of the original leakage energy in the standby state. (A more detailed evaluation of relationship between supply voltage scaling and leakage reduction in employed circuit can be obtained from [26].)

However, if, after being put into the low-power mode, the cache block is accessed (either due to the reference residing there or due to some other reference), the state-destroying mode pays a high performance and energy penalty (as the data needs to be accessed from memory). In contrast, under the same scenario, a cache block in the state-preserving state only needs to be reactivated. Therefore, whether state-preserving mode performs better than state-destroying mode depends largely on the duration of idleness for the cache block in question. This is, obviously, a characteristic of application access pattern and cache hierarchy configuration. In experiments, all cache lines are in the leakage-control mode before their first use for all strategies.

2.1.2 Leakage Optimization Strategies

In this section, a set of strategies is presented that exploit the state-preserving and state-destroying leakage energy optimization mechanisms. As explained below, these strategies differ from each other with respect to the circuit type that they employ (state-destroying versus state-preserving), whether they conservatively or speculatively turn off L2 subblocks, and the time that the L2 subblocks are reactivated (powered-on). All strategies power-manage portions of L2 blocks at the subblock granularity. A subblock of L2 is the same size as a block of L1 and is the unit of transfer between L1 and L2.

- *Conservative:* In this strategy, when a block in L1 is written to, the corresponding subblock in L2 is turned off by setting the standby bit, thereby destroying data and saving leakage in L2. This is a conservative strategy as, before turning off the subblock in L2, it waits until the corresponding block in L1 becomes dirty. Note that this strategy deactivates only dead L2 blocks (as they are written in L1) and

this characteristic makes it different from the remaining strategies considered in this work. It should also be noted that this strategy cannot optimize instruction accesses as instructions are not written.

- *S-SP-Lazy (Speculative, State-Preserving, and Lazy):* In this strategy, when data is brought from L2 to L1, the corresponding L2 subblock is put in a state-preserving leakage control mode. Consequently, as compared to the conservative strategy described above, this strategy has two important differences: it does not wait for the cache block in L1 to become dirty (i.e., it speculatively turns off the L2 subblock) and it does not lose data in L2. If the block in L1 is evicted, no action is performed if the L1 block is not dirty and the corresponding L2 subblock remains in state-preserving leakage control mode. Therefore, number of L2 subblocks in sleep mode can be larger than the number of L1 blocks. However, as in other strategies, if the evicted L1 block is dirty, the corresponding L2 subblock is reactivated and written into. Since a write buffer is employed, the performance penalty of the reactivation period can usually be masked.

- *S-SD-Lazy (Speculative, State-Destroying, and Lazy):* This strategy is similar to S-SP-Lazy, the difference being that the subblock in L2 is put in the state-destroying mode. So, as long as the subblock in L2 is in the powered off state, this strategy saves more leakage energy than S-SP-Lazy (which uses the state-preserving mode). On the other hand, when the L2 subblock needs to be accessed, this strategy pays a higher price than S-SP-Lazy as it needs to access the off-chip memory (as opposed to S-SP-Lazy which simply reactivates the L2 subblock).

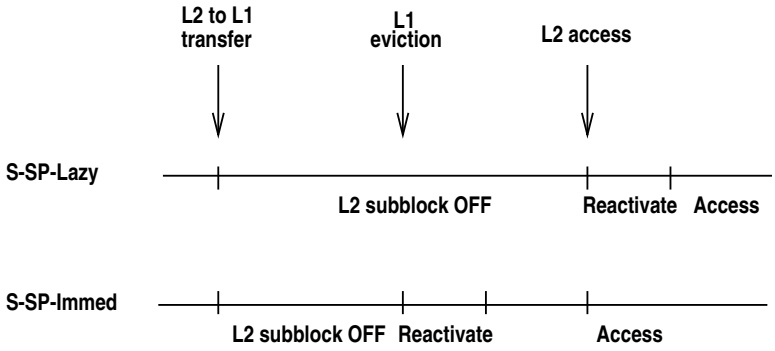

Fig. 2.2. Comparison of S-SP-Lazy and S-SP-Immed.

- *S-SP-Immed (Speculative, State-Preserving, and Immediate):* This is also similar to S-SP-Lazy except that the L2 subblock is reactivated whenever the corresponding L1 cache block needs to be replaced. This early reactivation (as compared to S-SP-Lazy where reactivation occurs only when the L2 block is accessed) can reduce energy savings compared to S-SP-Lazy. However, it has better performance behavior, as when the L2 cache block is accessed, a separate reactivation time is not spent. This situation is depicted in Figure 2.2. In the S-SP-Lazy case, the cache subblock is in the state-preserving leakage control mode between the time it is moved to L1 and the time that the L2 is accessed, whereas in S-SP-Immed, it is reactivated when the L1 eviction occurs. Consequently, in S-SP-Immed, the L2 subblock reactivation time can be hidden.

- *S-SD-Immed (Speculative, State-Destroying, and Immediate):* This strategy is similar to S-SD-Lazy except that the L2 subblock is reactivated and written back whenever the corresponding L1 cache block needs to be replaced. Its relative merits

Strategy	**When L2 subblock turned off?**	**Energy-saving mechanism in L2**	**When L2 subblock reactivated?**
Conservative	when L1 block becomes dirty	state-destroying	when accessed
S-SP-Lazy	when L2 subblock is moved to L1	state-preserving	when accessed
S-SD-Lazy	when L2 subblock is moved to L1	state-destroying	when accessed
S-SP-Immed	when L2 subblock is moved to L1	state-preserving	when L1 block is evicted
S-SD-Immed	when L2 subblock is moved to L1	state-destroying	when L1 block is evicted

Table 2.1. Proposed leakage energy saving strategies.

with respect to S-SD-Lazy are similar to those of S-SP-Immed with respect to S-SP-Lazy. Similarly, its advantages/disadvantages compared to S-SP-Immed are similar to those of S-SD-Lazy compared to S-SP-Lazy.

Table 2.1 summarizes these five strategies highlighting their differences. It should be noted, however, that when an evicted L1 cache block is dirty, the corresponding L2 subblock needs to be reactivated (by resetting the standby bit) irrespective of the energy-saving strategy used. Therefore, this case is not listed separately under the last column in Table 2.1. It also needs to be mentioned that in state-destroying modes with set-associative caches, when all subblocks in a given L2 block are moved to L1, this L2 block is invalidated, becoming a suitable candidate for the next cache block replacement in L2. However, if there is a single valid subblock in the block, the block is considered valid and participates in the LRU replacement process.

Simulation Parameter	Value
Processor Core	
Functional Units	4 integer and 4 FP ALUs 1 integer multiplier/divider 1 FP multiplier/divider
LSQ Size	32 Instructions
RUU Size	64 Instructions
Fetch Width	4 instructions/cycle
Decode Width	4 instructions/cycle
Issue Width	4 instructions/cycle
Commit Width	4 instructions/cycle
Fetch Queue Size	4 instructions
Cycle Time	0.5ns
Cache & Memory Hierarchy	
L1 Instruction Cache	32KB, 32 byte blocks, 2-way, 1 cycle latency
L1 Data Cache	32KB, 32 byte blocks, 2-way, 1 cycle latency
L2 Cache	1MB unified, 2-way, 128 byte blocks, 10 cycle latency
Data TLB	128 entries, full-associative, 30 cycle miss latency
Instruction TLB	64 entries, full-associative, 30 cycle miss latency
Memory	100 cycle latency
Energy Management	
Technology	0.07 micron
Supply Voltage	1.0V
Voltage Supply Settling Time	1 cycle
Dynamic Energy per L1 Access	0.565nJ
Dynamic Energy per L2 Access	5.83nJ
Leakage Energy per L1 Block per Active Cycle	0.551pJ
Leakage Energy per L2 Subblock per Standby Cycle (state-preserving)	0.055pJ
Leakage Energy per L2 Subblock per Standby Cycle (state-destroying)	0pJ
Control Energy	0.055nJ

Table 2.2. Base configuration.

2.1.3 Simulation Parameters and Benchmarks

SimpleScalar 3.0 [16] is used to implement the proposed energy-saving optimization strategies. SimpleScalar is a tool-set to simulate application programs on a range of modern processors and systems using fast execution-driven simulation. It provides a detailed simulator for an out-of-order issue processor that supports non-blocking caches, speculative execution, and state-of-the-art branch prediction. In this work, the *sim-outorder* component is used. Table 2.2 gives the simulation parameters used for the base configuration.

Energy model for 70nm technology from CACTI 3.0 is used to get the dynamic energies of accessing L1 and L2 caches. It is assumed that the leakage energy per cycle of the entire L1 cache is equal to the dynamic energy consumed per access, and the leakage of the L2 subblock is equal to that of the L1 block.

The effectiveness of these strategies is evaluated using a set of benchmark programs. The benchmarks are codes from MediaBench suite [49] and SPEC CINT2000 [5] benchmarks. These two groups of codes are selected as they represent different access patterns. For each code in MediaBench suite, the simulations are run to completion. Except bzip2 and mcf, benchmarks in SPEC CINT2000 suits are first fast forwarded 300 million instructions and then simulated 200 million instructions. No instruction is fast forwarded for bzip2 and mcf due to their specific characteristics [22].

The important characteristics of these benchmarks are listed in Table 2.3. The fourth and fifth columns in this figure give the total leakage and dynamic energy consumptions, respectively, in L1-L2 cache hierarchy assuming all blocks are powered off

Benchmark	Input	Execution Cycles (millions)	Cache Energy	
			Leakage (mJ)	Dynamic (mJ)
adpcm-rawcaudio	clinton.pcm	4.74	1.88 (17.5%)	8.90 (82.5%)
adpcm-rawdaudio	clinton.adpcm	4.00	1.54 (18.4%)	6.85 (81.6%)
cjpeg	testimg.ppm	7.69	54.25 (73.0%)	20.09 (27.0%)
djpeg	testimg.jpg	2.93	13.61 (71.2%)	5.49 (28.8%)
epic	test_image.pgm	21.33	352.65 (85.6%)	59.20 (14.4%)
unepic	test.image.pgm.E	5.53	75.31 (88.3%)	9.98 (11.7%)
g721-decode	clinton.g721	119.09	338.40 (50.5%)	332.10 (49.5%)
g721-encode	clinton.pcm	123.98	353.54 (50.9%)	341.75 (49.1%)
mesa-mipmap	-	37.09	1032.67 (92.8%)	80.75 (7.2%)
mesa-osdemo	-	11.97	315.95 (91.5%)	29.40 (8.5%)
mpeg2-decode	mei16v2.m2v	65.36	638.12 (75.6%)	205.71 (24.4%)
gzip	input.source	105.19	2299.63 (89.3%)	276.60 (10.7%)
vpr	net.in arch.in place.in	160.84	4248.19 (93.6%)	289.69 (6.4%)
gcc	scilab.i	220.77	5464.22 (93.8%)	358.96 (6.2%)
mcf	inp.in	171.50	4861.76 (93.7%)	327.75 (6.3%)
perlbmk	2.1.dict -batch ref.in	133.28	3507.31 (92.7%)	276.95 (7.3%)
vortex	bendian1.raw	325.33	7760.05 (95.4%)	375.26 (4.6%)
bzip2	input.source	779.39	21882.9 (98.7%)	299.60 (1.3%)
twolf	ref	448.16	12462.3 (97.1%)	367.63 (2.9%)

Table 2.3. Benchmarks used in experiments and their important characteristics.

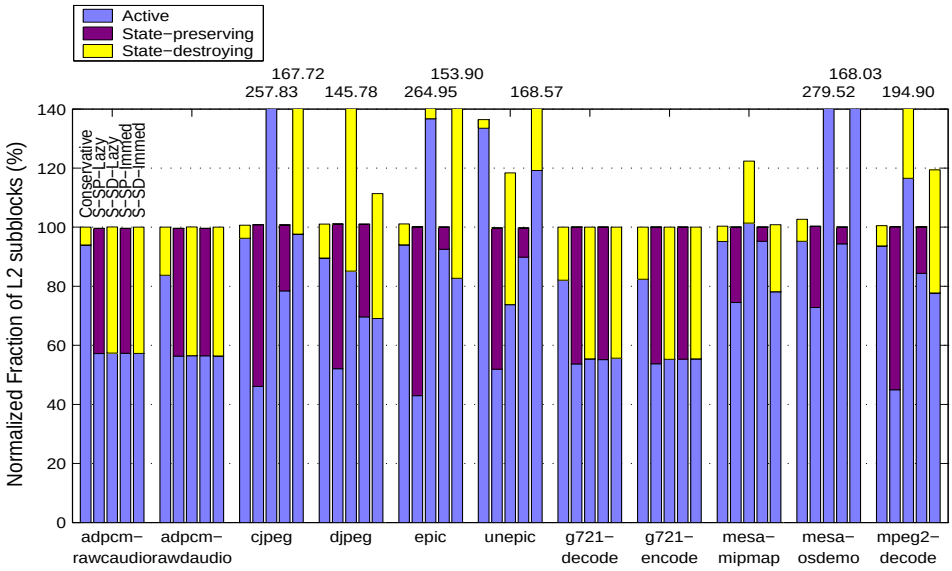

Fig. 2.3. Normalized fraction of subblocks in L2 in different states. (MediaBench)

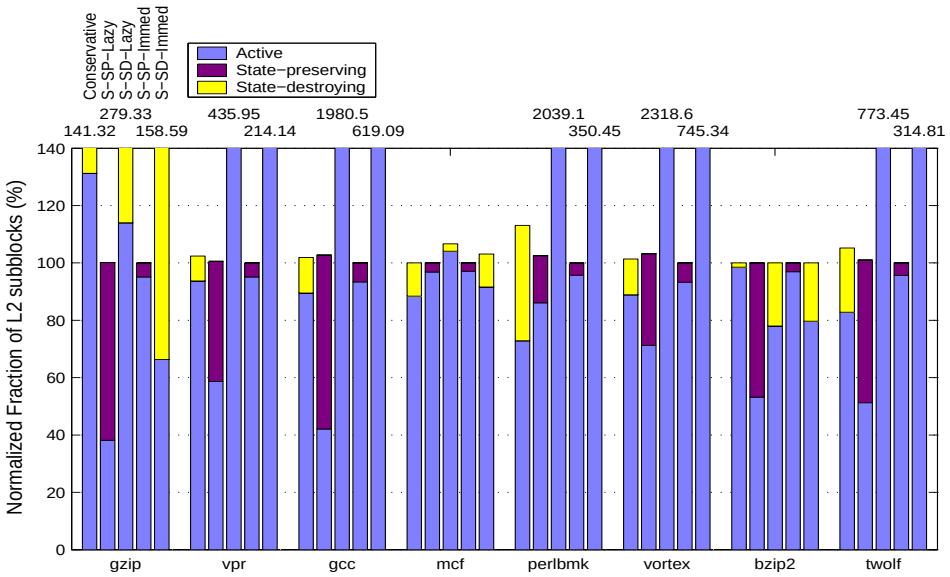

Fig. 2.4. Normalized fraction of subblocks in L2 in different states. (SPEC CINT2000)

until their first use and never turned off after that. It can be observed that these codes expend a large percentage of leakage energy (77.3% of the cache hierarchy energy on the average). Consequently, we can expect large leakage energy savings using proposed strategies. Note that percentage of dynamic energy is dependent on the size of the memory and the number and duration between memory accesses. Most of the energy results given in following subsections are results normalized with respect to the values in the fourth and fifth columns of Table 2.3.

2.1.4 Experimental Results of Proposed Strategies

Figure 2.3 and Figure 2.4 show the fraction of subblocks in L2 in active, state-destroying, and state-preserving states. Each portion of the bar is the sum of the number of L2 subblocks in active, state-preserving, or state-destroying states (excluding those

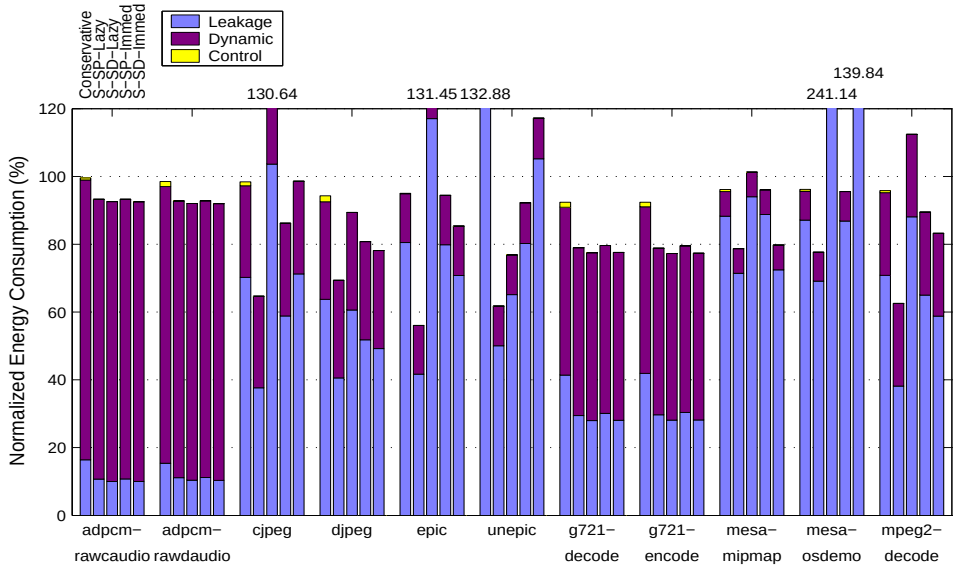

Fig. 2.5. Normalized energy consumption of proposed optimization strategies. (MediaBench)

that were never used in the entire execution) over each clock cycle normalized to the sum of active L2 subblocks over each clock cycle in the original execution.

In Conservative, S-SD-Lazy, and S-SD-Immed, subblocks in L2 are distributed between active and state-destroying states. In S-SP-Lazy and S-SP-Immed, subblocks in L2 are distributed between active and state-preserving states. The reason that some bars exceed 100% is due to the fact that some schemes impact performance and the execution times of benchmarks in such schemes are longer than those in the original execution. Therefore, the total number of subblocks over each clock cycle is larger than 100% after normalization. It can be observed that a large fraction of subblocks is in state-preserving and state-destroying states for most benchmarks.

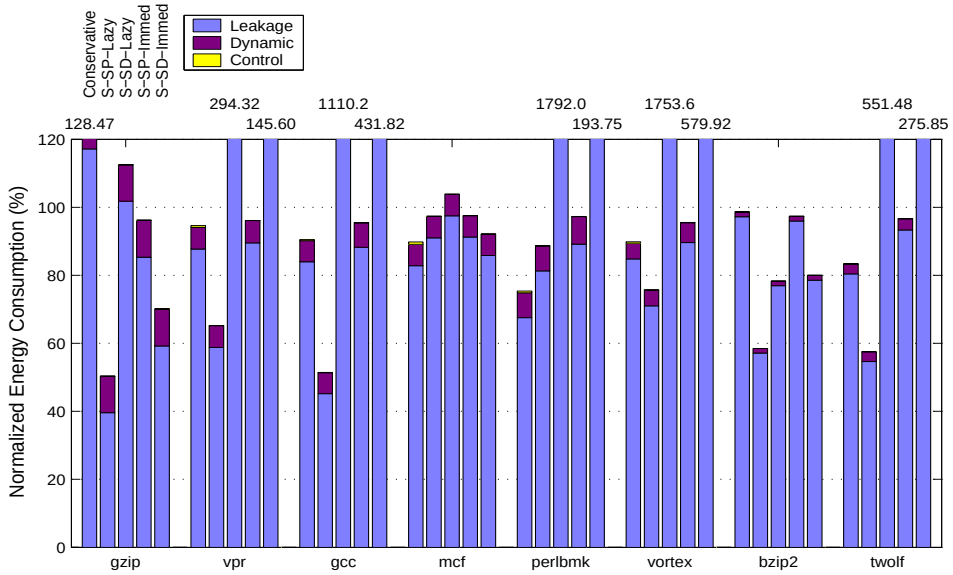

Fig. 2.6. Normalized energy consumption of proposed optimization strategies. (SPEC CINT2000)

Figure 2.5 and Figure 2.6 show the normalized energy consumption for proposed five optimization strategies. Each bar in this figure is divided into three parts: leakage energy, dynamic energy, and control overhead (energy), and is normalized to the total energy consumed by original execution of benchmarks, which is the sum of the fourth and fifth columns of Table 2.3. Dynamic energy is reported because proposed leakage optimization strategies may increase dynamic energy consumption.

The following observations can be made from these results. First, the control overhead is nearly negligible. This is because, compared to the total number of cache accesses, the number of state transitions is very low. In fact, control energy overhead is only observed in the Conservative strategy. This makes sense as in this strategy the control overhead is directly proportional to the number of L1 writes. Second, among

proposed five strategies, S-SP-Lazy generates the best energy results. It reduces leakage energy consumption by 37.7% on the average across all benchmarks and overall cache energy (including dynamic energy and control overhead as well) by 28.5% on the average. On the other hand, the average leakage (the average overall energy) improvements due to Conservative, S-SD-Lazy, S-SP-Immed, and S-SD-Immed are 5.9% (3.0%), -281.0% (-269.4%), 14.6% (7.7%), and -48.1% (-52.1%), respectively. (A negative value indicates an increase in energy consumption).

Comparing S-SP-Lazy and S-SD-Lazy, recall that neither of them reactivates L2 subblock when the corresponding L1 block is evicted. If a clean block in L1 is evicted, the corresponding subblock in L2 is in the state-preserving state for S-SP-Lazy but is in the state-destroying state for S-SD-Lazy. Then, when the next access occurs, S-SP-Lazy will incur 1 cycle delay (for reactivation), whereas S-SD-Lazy will lead to 100 cycle penalty (for memory access). During this long memory access all active L1 and L2 blocks leak. Consequently, in most of codes, S-SP-Lazy exhibits a better energy behavior than S-SD-Lazy. There are, however, exceptions to this general trend: *adpcm-rawcaudio*, *adpcm-rawdaudio*, *g721-decode*, and *g721-encode*. In these codes, the L1 replacement rate (the ratio between the number of L1 replacements and total L1 accesses) is very low (around 0.0003%) and L2 subblocks stay in energy-saving state longer (which works in favor of S-SD-Lazy).

Comparing S-SP-Immed and S-SD-Immed, recall that these two schemes differ from S-SP-Lazy and S-SD-Lazy in that they reactivate the L2 subblock when the corresponding L1 subblock is evicted. When data is moved from L2 to L1, S-SP-Immed places the corresponding subblock into the state-preserving state, whereas S-SD-Immed

puts it in the state-destroying mode. So, as far as a single subblock is concerned, S-SD-Immed seems to be more energy-efficient. However, if all subblocks in a given L2 block are moved into L1, S-SD-Immed invalidates the entire L2 cache block (i.e., makes it available for replacement). After that, when a new access is made to this cache block, a miss is incurred and main memory needs to be accessed (a 100 cycle delay). During this memory access all active cache blocks in L1 and L2 consume leakage energy. Although the early reactivation (i.e., reactivation in L1 block eviction time) tries to write data back from the L1 cache to the L2 cache, this operation succeeds only when the cache block is in the valid state (i.e., there exists at least a single valid L2 subblock in the cache block). In this scenario, the early reactivation succeeds in S-SP-Immed but fails in S-SD-Immed. Consequently, in such cases, S-SP-Immed might perform better than S-SD-Immed. The results in Figure 2.5 and Figure 2.6 indicate that in eight benchmarks S-SP-Immed consumes less energy than S-SD-Immed (due to frequent memory accesses). In the remaining codes, S-SD-Immed performs better than S-SP-Immed (due to lack of the above mentioned scenario).

When comparing S-SP-Lazy and S-SP-Immed, both of them preserve the data in L2, but S-SP-Immed reactivates the subblock in L2 when the corresponding block is evicted from L1. Therefore, it tends to maintain the same execution time as the original (unoptimized) case, incurring some extra energy due to early reactivation. Therefore, its energy behavior is worse than S-SP-Lazy. However, its performance is better than S-SP-Lazy in almost all cases.

Finally, comparing S-SD-Lazy and S-SD-Immed, it can be observed that although both of them destroy data in L2, S-SD-Immed has a better chance for avoiding main

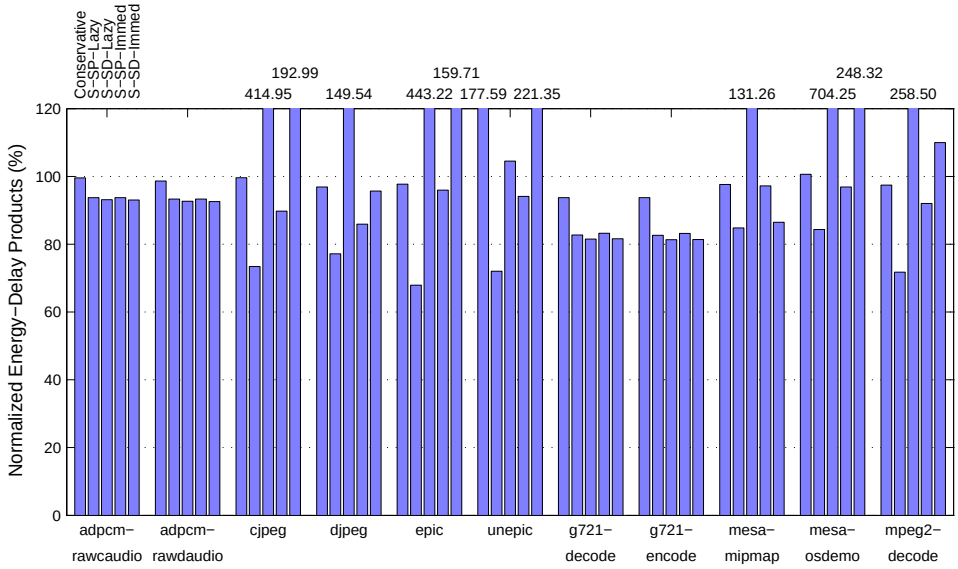

Fig. 2.7. Normalized energy-delay products of proposed optimization strategies. (MediaBench)

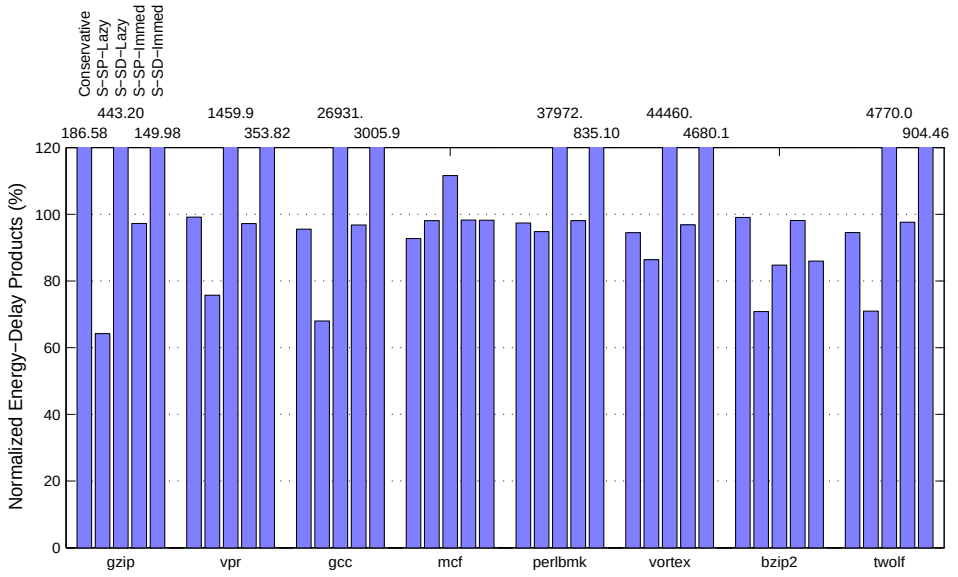

Fig. 2.8. Normalized energy-delay products of proposed optimization strategies. (SPEC CINT2000)

memory access, thanks to the early reactivation. In most of the benchmarks, S-SD-Immed consumes less energy than S-SD-Lazy.

Energy consumed in the cache system is only a part of this picture. For a fair comparison between the different energy optimization strategies, it is needed to account for the additional execution cycles and the additional leakage expended in the other parts of the processor during these additional cycles. The contribution of the rest of the processor (other than the cache subsystem) is conservatively assumed 30% to the leakage energy. The energy-delay product is a suitable metric that allows evaluating the impact of an optimization on both the performance and energy.

The results given in Figure 2.7 and Figure 2.8 are the normalized energy-delay products (with respect to the original cache management without any leakage energy control). It is easy to see that S-SD-Lazy and S-SD-Immed do not perform well due to frequent main memory visits resulting from L2 misses. However, it is observed that the S-SP-Lazy strategy reduces the energy-delay product by 20.4%, on the average. Apart from S-SP-Lazy, only S-SP-Immed improves the energy delay product (6.0% on the average). State-destroying optimization strategies, on the other hand, increase energy-delay product by 5.3% (Conservative), 6152.1% (S-SD-Lazy), and 509.3% (S-SD-Immed). Based on these results, it can be concluded that working with a state-preserving mode is extremely important to improve both energy and the energy-delay product.

The first five groups of bars in Figure 2.9 show the average values (percentage improvements) for proposed five optimizations across all benchmark programs for leakage energy, overall cache energy, and energy-delay product. The last four groups of bars are discussed in the next subsection.

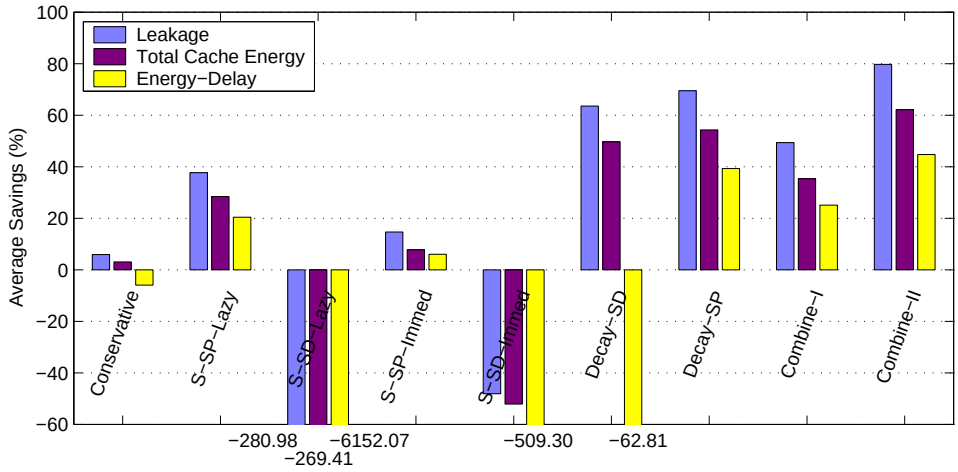

Fig. 2.9. Average savings (over all benchmarks) for different optimization strategies.

2.1.5 Comparison and Integration with Other Strategies

In [41], Kaxiras et al. present a leakage energy reduction technique for cache memories. This technique, called *cache decay*, is based on the idea that a cache block that is not used for a sufficiently long period of time can be considered *dead*. More specifically, with each cache block, they associate a small 4-state FSM (finite state machine). The FSM steps through these states as long as the cache block is not accessed. When the last state is reached, the cache block is turned off.

To compare this technique with the proposed approach, cache decay is implemented in SimpleScalar [16]. The first implementation (called *Decay-SD*) is a straightforward extension of their approach to a cache hierarchy (instead of just the L1 cache). Specifically, the cache decay method is applied to both L1 and L2 using the state-destroying leakage saving technology.

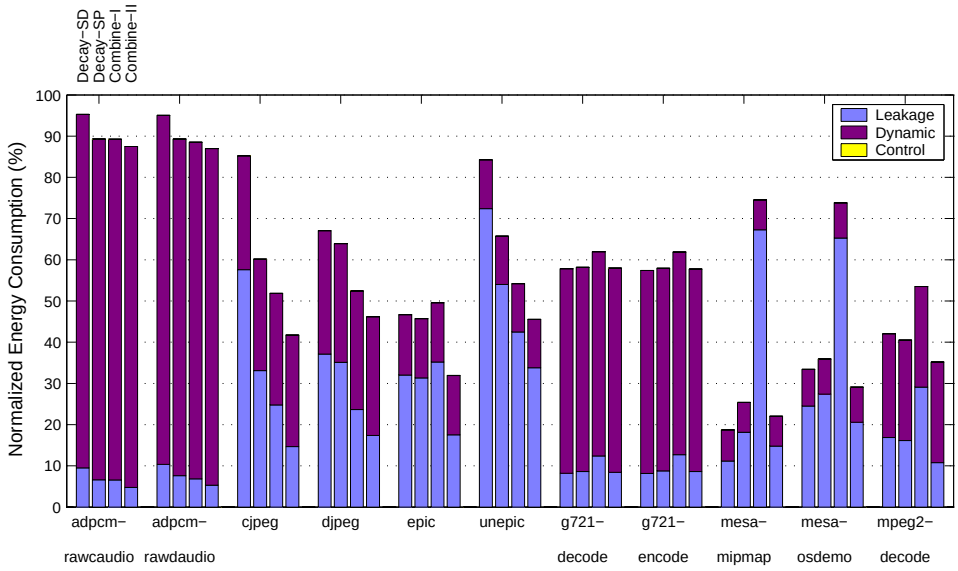

Fig. 2.10. Normalized energy consumption of cache decay and combined strategies. (MediaBench)

Then, this scheme is further enhanced by employing the state-preserving strategy in both L1 and L2. In this second implementation (called *Decay-SP*), the L2 cache is energy-managed at the subblock granularity and the FSM is used to transition L2 subblocks into a state-preserving mode (as opposed to the state-destroying mode in Decay-SD). In both of these implementations, the threshold values of decay interval are the same as those in [41] (i.e., 10K cycles for L1 and 1M cycles for L2).

In addition to these two strategies, two strategies that combine the cache decay scheme with proposed optimization strategy are implemented. *Combine-I* corresponds to a strategy where L1 leakage energy is optimized using cache decay method, whereas the L2 cache energy is optimized using S-SP-Lazy strategy. Finally, *Combine-II* employs cache decay for L1, but uses both cache decay (as in Decay-SP) and S-SP-Lazy strategy

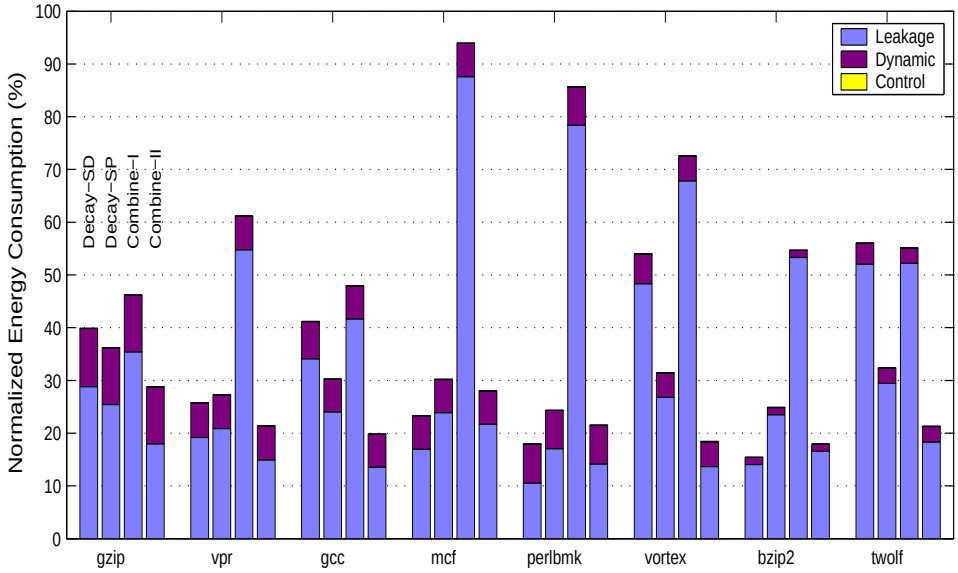

Fig. 2.11. Normalized energy consumption of cache decay and combined strategies. (SPEC CINT2000)

for L2. In both *Combine-I* and *Combine-II*, state-preserving mechanism is employed in L1 and L2. The reason using S-SP-Lazy in these last two versions (instead of other speculative strategies) is that it performs better than others as shown through experimental results discussed earlier.

Figures 2.10 and Figures 2.11 show the normalized energy consumptions. Figures 2.12 and Figures 2.13 show the normalized energy-delay products, respectively, for these last four strategies mentioned above. It can be observed that Decay-SD performs quite well and improves leakage energy consumption and overall cache energy by 63.6% and 49.7%, on the average. However, it increases execution cycles as, under this optimization scheme, it is possible that a cache block can be destroyed in L1 as well as in L2. Consequently, it incurs frequent main memory accesses, thus degrading the energy-delay

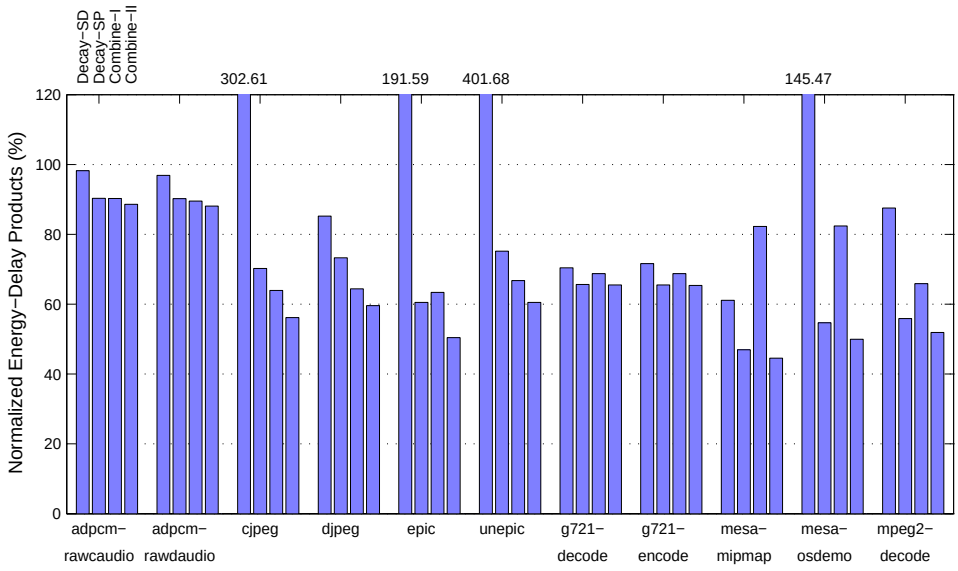

Fig. 2.12. Normalized energy-delay products of cache decay and combined strategies. (MediaBench)

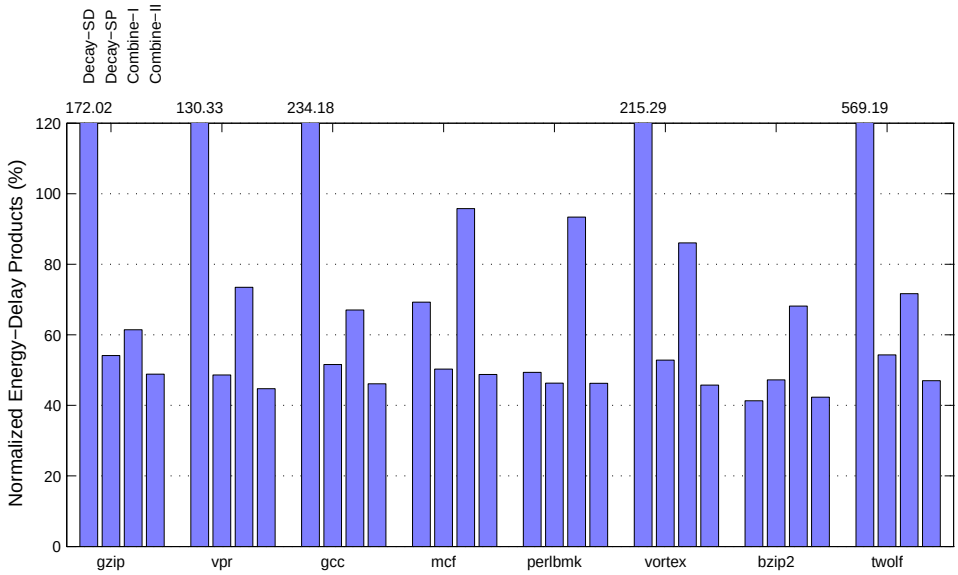

Fig. 2.13. Normalized energy-delay products of cache decay and combined strategies. (SPEC CINT2000)

product by 62.8%. In most cases, Decay-SP improves over Decay-SD in both energy and energy-delay product. It improves leakage energy, total cache energy, and energy-delay product by 69.5%, 54.3%, and 39.3%, respectively. This is because in Decay-SP, both L1 and L2 cache management do not destroy data, thereby preventing frequent main memory accesses. Also, as compared to Decay-SD, it manages L2 leakage energy in subblock granularity. Recall that S-SP-Lazy's leakage energy, total cache energy, and energy-delay product improvements were 37.6%, 28.5%, and 20.4%, respectively. While Decay-SD provides a better energy reduction over S-SP-Lazy, it suffers from long execution times. Decay-SP, however, performs better than S-SP-Lazy in all aspects. It should also be stressed that while S-SP-Lazy targets only the lines in L2 cache that are moved into L1 cache, Decay-SD and Decay-SP target both caches and hence they have potentially larger optimization scope. In fact, comparing the savings only in the L2 cache shows that S-SP-Lazy, Decay-SD, and Decay-SP reduce leakage energy consumption by 48.2%, 59.7%, and 69.2%.

Combine-I improves the unoptimized leakage energy by 49.3%, overall cache energy by 35.3%, and energy-delay product by 25.1%. But the savings are less than that of Decay-SP. This is because S-SP-Lazy targets only the lines in L2 cache that are moved into L1 cache and does not target the cache blocks which just stay in L2 cache for a long period of time (Decay-SP does). Therefore, Combine-II is implemented that employs both S-SP-Lazy and cache decay in L2 cache. Combine-II generates the best energy results among these optimization strategies. As compared to the unoptimized case, it improves leakage and overall cache energy by 79.7% and 62.1%, respectively. These energy benefits are due to its aggressive optimization strategy in L2. More specifically,

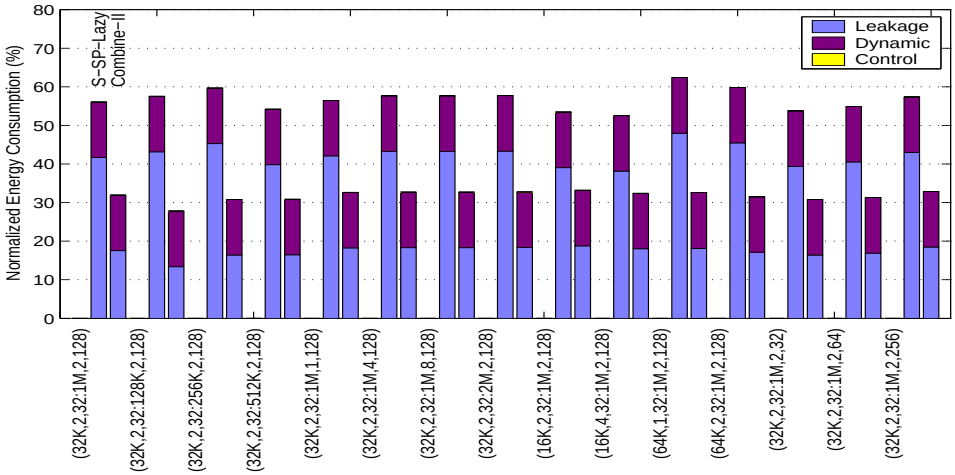

Fig. 2.14. Normalized energy consumptions for different cache hierarchy configurations (*epic*).

the two different methods (S-SP-Lazy and cache decay) complement with each other to optimize L2 energy. The last four groups of bars in Figure 2.9 summarize the average improvements for the four optimization strategies discussed in this section from the leakage energy, overall cache energy, and energy-delay product perspectives.

2.1.6 Sensitivity Analysis

The robustness of the proposed energy optimization strategies is measured using different parameters. In particular, the energy savings is measured when cache configuration parameters (cache capacity, associativity, and block size), relative magnitudes of leakage and dynamic energies, leakage saving factor in the state-preserving mode, the ratio of L2 subblock and L1 block leakage, and the effect of different reactivation time.

First, the L1 and L2 cache parameters are varied for benchmark (*epic*) with S-SP-Lazy and Combine-II strategies. The trends observed in other benchmarks are similar to

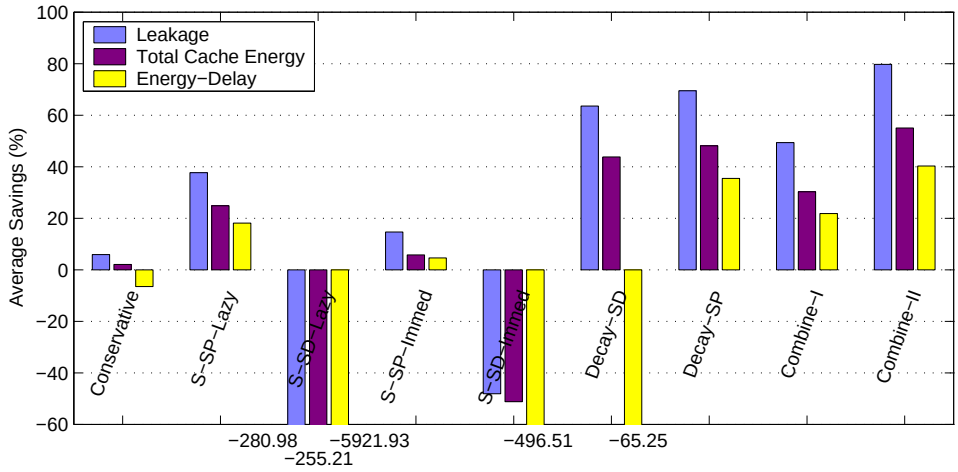

Fig. 2.15. Sensitivity to the relative magnitude of dynamic versus leakage: average leakage energy, total cache energy, and energy-delay product improvements for different optimization strategies.(over all benchmarks)

that of *epic*, so they are not included. Figure 2.14 gives the cache energy consumptions. Each cache configuration is denoted as

(L1-Size,L1-Associativity,L1-Block-Size;L2-Size,L2-Associativity,L2-Block-Size),

with (32KB,2,32;1MB,2,128) being default configuration. Note that in each experiment L1 instruction and L1 data caches have the same configuration. Each bar in this figure represents energy consumption normalized with respect to the energy consumption of the simulation without any leakage control with the default cache configuration. First, it is observed that energy savings are obtained across different configuration. Further, when the L2 cache size increases, a decrease in energy savings is observed. The reason for this is that when the L2 capacity is higher, the number of L2 subblocks that are in active or state-preserving mode increases.

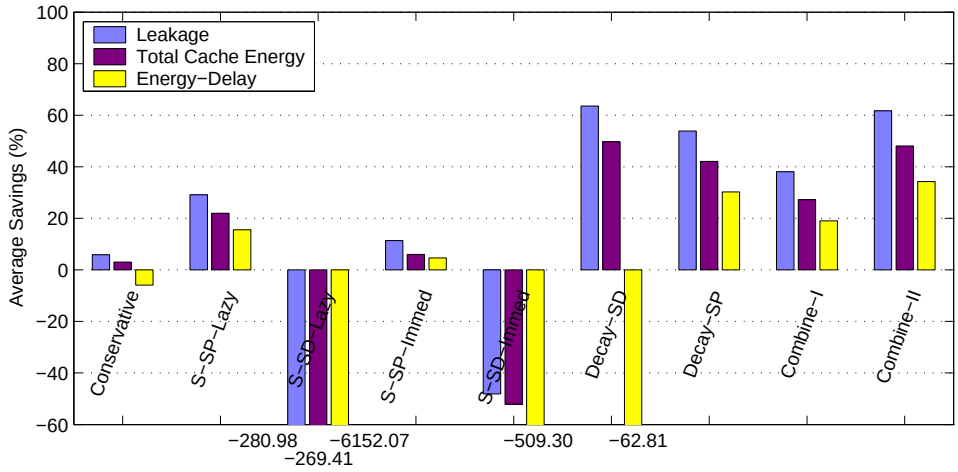

Fig. 2.16. Sensitivity to the leakage saving factor: % improvements in average leakage energy, total cache energy, and energy-delay product for different optimization strategies. (over all benchmarks)

Next, the relative magnitude of leakage energy per cycle with respect to dynamic energy per access is modified. Specifically, the leakage energy per cycle of the entire L1 cache is assumed to be equal to half of the dynamic energy consumed per access. So, this gives more weight to dynamic energy. Figure 2.15 gives the average improvements in this case for leakage energy, overall cache energy, and energy-delay product. As in the previous case (Figure 2.9), Combine-II results in the best energy behavior (79.73% improvement in leakage and 55.0% improvement in total cache energy) and energy-delay product (40.3%). As expected, the leakage energy improvements do not change significantly. But, since the impact of dynamic energy is increased, a decrease is observed in overall cache energy savings compared to Figure 2.9. In addition, in this case, the overhead cost (from the dynamic energy viewpoint) (due to proposed leakage optimization

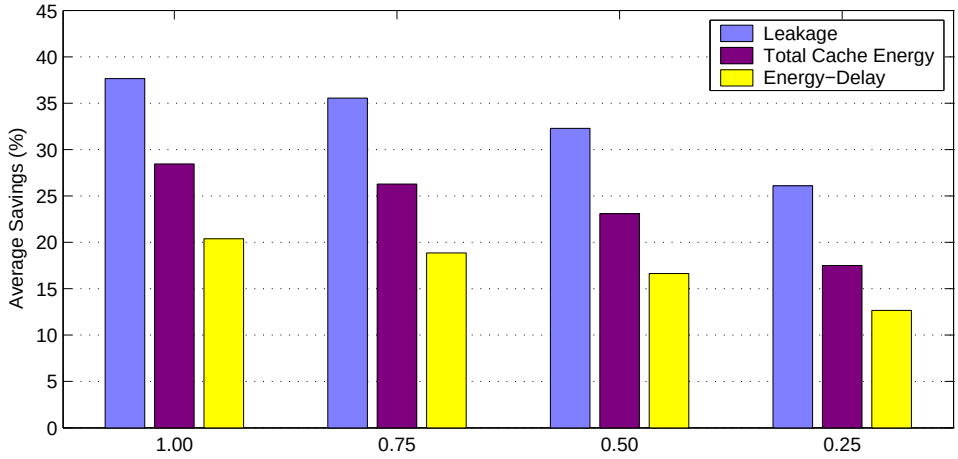

Fig. 2.17. Sensitivity to the ratio of L2 subblock and L1 block leakage in S-SP-Lazy method.(over all benchmarks)

strategies) is higher. A similar reduction is observed in energy-delay product savings. This is a direct result of the decrease in overall energy savings.

Then, the leakage saving factor is modified when state-preserving leakage control mode is employed. Recall that it is assumed earlier that, in each cycle, the state-preserving strategy consumes 10% of the original (active) per cycle leakage energy. This figure is called *leakage saving factor*. In this set of experiments, this figure is changed to 30%. All other simulation parameters are the same as in base configuration. The results given in Figure 2.16 (average values over all benchmark codes) reveal that, even in this case, the state-preserving strategy is superior to the state-destroying one. In particular, both S-SP-Lazy and S-SP-Immed improves leakage energy, overall energy, and energy-delay product. However, the energy savings due to preserving state are not as good. For instance, with this new setup, S-SP-Lazy and S-SP-Immed improve overall

cache energy by 29.2% and 11.4%, respectively. In contrast, the corresponding numbers when I employed a leakage saving factor of 10% were 37.7% and 14.6%.

The ratio between leakage of L2 subblock and leakage of L1 block is also modified. Recall that the default value was 1.00. But, using higher threshold voltage can let L2 cache consume less leakage energy at the expense of extra L2 latency. Figure 2.17 shows the improvements of S-SP-Lazy method in leakage and total cache energy for different ratios. It can be concluded that proposed strategy still brings benefits when the ratio of L2 vs L1 is reduced.

The time to reactivate a cache line in state-preserving or state-destroying mode to its normal state depends on the actual circuit implementation. So far in this work, the voltage scaling approach is used that incurs only a single cycle penalty for this reactivation. An alternate implementation is considered using a modified gated-Vdd technique to study the influence of this parameter.

A 16-bit array of memory cells is custom-designed in 0.07 micron technology and circuit simulation is performed using HSPICE. It is observed that the states of these cells were maintained from a supply voltage of 1.0V down to 120mV. In order to achieve a 120mV voltage, a sized NMOS switch was introduced between the ground rail and the memory cell. Using the NMOS switch can reduce both bitline and cell leakage. When the power-switch is turned-on, a normal supply voltage is provided to the circuit. However, when the power-switch is turned-off, the ground level rises to 0.88V from 0V. Note that this is achieved by having an appropriately sized power-switch that has a controlled leakage to provide the required minimum supply voltage. A 0.68μm/0.07μm

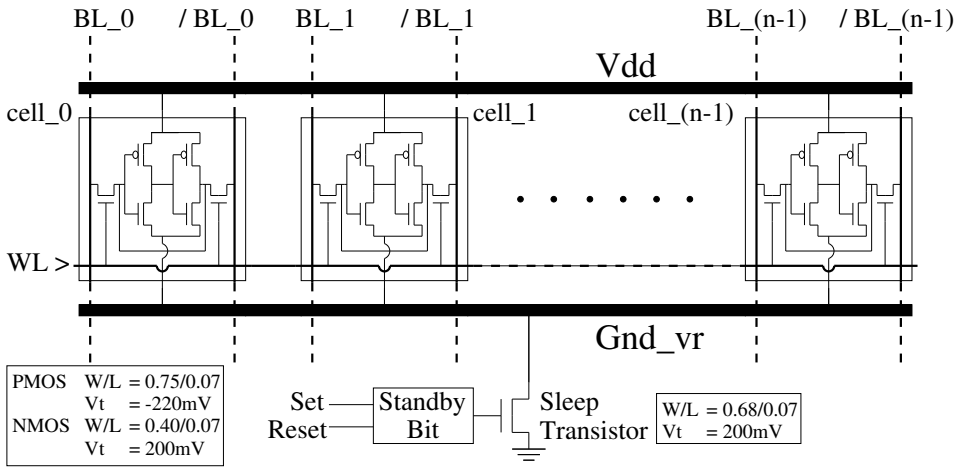

Fig. 2.18. An L2 subblock augmented with leakage control mechanism.

(width/length) NMOS device (with a threshold voltage of 200mV) was used as a power-switch along with each memory array to achieve the required supply voltage of 120mV (See Figure 2.18). It is observed from simulations that the overall leakage of the memory array can be reduced to 4% of original leakage using the state-preserving mechanism. However, in order to activate a cache line in state-preserving mode to a normal mode, simulations indicate that 19ns latency is required for the ground level to settle back to 0V. This would incur a 38 cycle penalty as compared to the 1 cycle latency required for voltage settling in the circuit described in previous subsection.

Thus, it is clear that the circuit choices can influence the latency for reactiving a cache block in standby mode. To study this impact, in Figure 2.19, the *epic* is simulated with different reactivation latency 50 cycles, 20 cycles, and 1 cycle, and the effect on energy consumption and energy-delay product of four strategies is compared. Comparing with long reactivation latency, small reactivation latency performs very small increase in

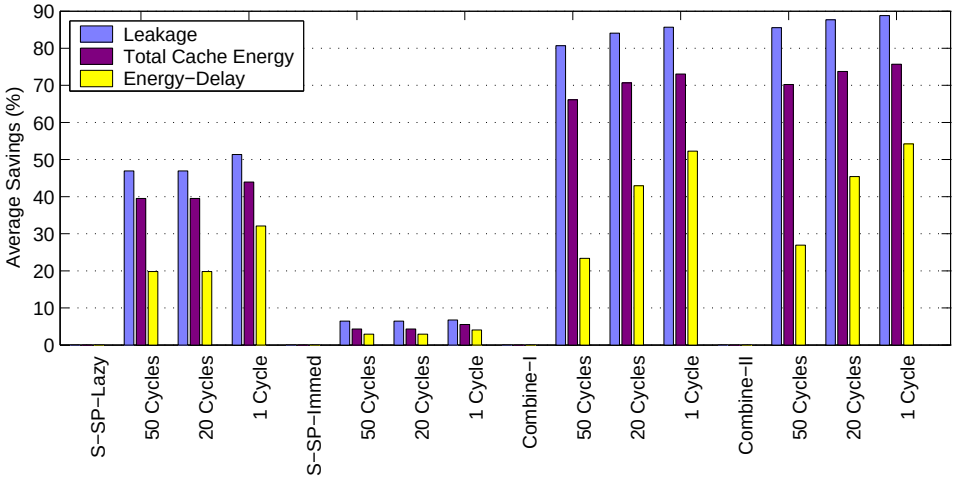

Fig. 2.19. Sensitivity to the reactivation time: % improvements in average leakage energy, total cache energy, and energy-delay product for *epic*.

leakage and total cache energy savings, but it has significant improvements on energy-delay product for S-SP-Lazy, Combine-I, and Combine-II strategies.

2.2 Crossbar Connected Caches for Reducing Energy Consumption

Since feature size of silicon fabrication technology is continuously shrinking and is expected to be so in the near future, the problem of how to utilize large number of transistors is an important one. One option is to invest these transistors to building sophisticated single processor based machines with complex issue logic and functional units. However, complex single processor architectures are difficult to built and verify and, beyond a degree of sophistication, they do not payoff. An alternate strategy is to employ multiple processors, each with a simple issue logic and pipeline. Current examples of such architectures include Stanford Hydra [6], Sun's MAJC [7], and IBM's Power4 [2]. It should be noticed that an on-chip multiprocessor can exploit parallelism

at different granularities such as instruction level, thread level, and process level. In many cases, this can lead to large performance benefits.

There are several alternatives for building cache architecture for on-chip multiprocessors. The first one is to adopt a single multi-ported architecture shared by multiple processors. There are two major advantages of this alternative: (1) constructive interference can reduce overall miss rates and (2) inter-processor communication is easy to implement. However, a large multi-ported cache can consume significant energy. In addition, this is not a scalable option. The second alternative is to allow each processor to have its own private cache [48, 66, 79]. The main benefits of the private cache option are low power per access, low access latency, and good scalability. Its main drawback is duplicate copies in different caches. In addition, one may need to employ a complex cache coherence protocol to maintain consistency.

In this section, a new cache configuration is proposed, referred to as the *crossbar-connected cache* or *CCC* for short, that tries to combine the advantages of the two options discussed above without their drawbacks. Specifically, the shared cache is divided into multiple banks, and an N×M crossbar is used to connect N processors and M banks. In this way, the duplication problem is removed, the sophisticated consistency mechanism is unnecessary, and each bank can be simple in architecture.

2.2.1 Cache Architectures

First, data and instruction sharing between on-chip processors is investigated. Consider the graph given in Figure 2.20. This graph gives the percentage sharing for instructions and data when SPLASH-2 benchmarks [84] are executed using 8 processors

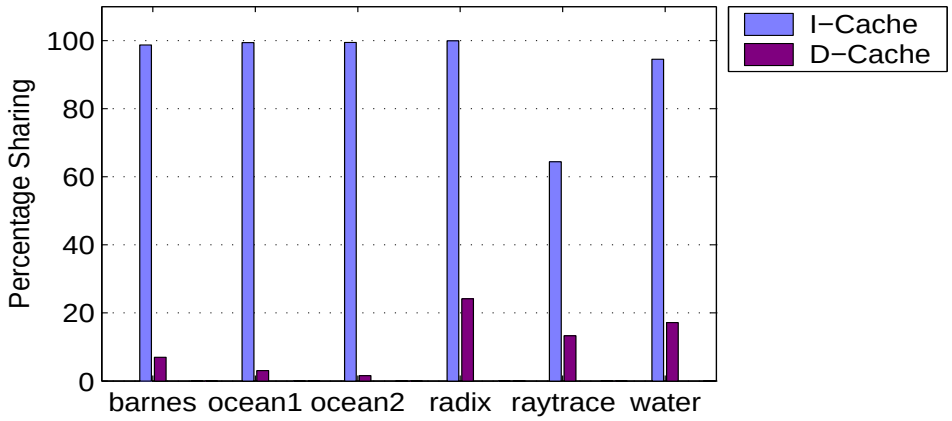

Fig. 2.20. Percentage sharing of instructions and data. (8 processors).

with private L1 caches (Details of experimental platform and benchmarks will be given later). The percentage sharing is referred to the percentage of data or instructions shared by at least two processors when considering the entire execution of the application. The first observation is that instruction sharing is extremely high. This is a direct result of data parallelism exploited in these benchmarks. That is, different processors typically work on (mostly) different sets of data using the same set of instructions. It is observed that instruction sharing is above 90% for all applications except raytrace. These results clearly imply a large amount of duplication across private caches as far as instructions are concerned. Consequently, a strategy that eliminates this duplication can bring large benefits (as there will be more available space). That is, one can increase the effective cache capacity by reducing duplications. Note that another potential problem associated with the duplicates is extra leakage energy to power the corresponding cache lines.

The objective in introducing CCC is to reduce duplicate entries as much as possible. When one looks at the data sharing in Figure 2.20, the picture changes. It is

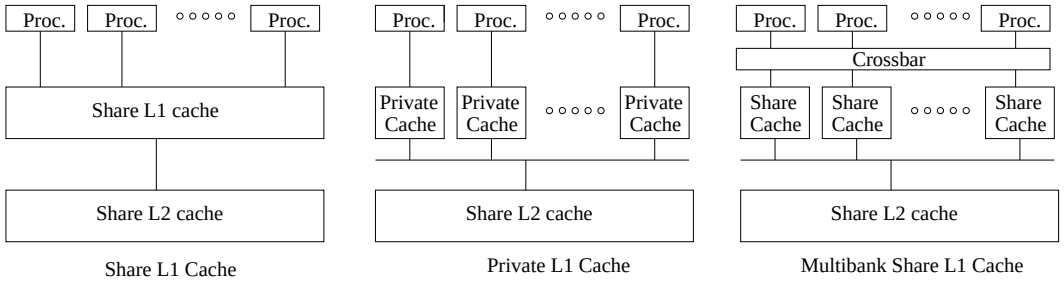

Fig. 2.21. Different cache organizations used in simulations: multi-ported shared cache, private caches, and CCC.

observed that the percentage sharing is low, averaging around 11%. However, employing CCC can still be desirable from the cache coherence viewpoint (since complex coherence protocols associated with private caches can be a major energy consumer). Based on these observations, It is expected that using CCC instead of other alternatives might be very beneficial in practice.

In an attempt to build a high-performance cache system, a crossbar is used as the interconnect between the processors and L1 banks. As compared to a bus-based architecture, the crossbar-based architecture provides simultaneous accesses to multiple cache banks (which, in turn, helps to get better performance). Figure 2.21 illustrates the three different cache organizations evaluated in this section: a multi-ported shared cache, private caches, and CCC. A common characteristic of these configurations is that they all have a shared L2. In other words, they differ only in L1 organization.

Due to the crossbar placed between processors and L1 banks, the pipeline stages of the processor are needed to be restructured. In simulations, the pipeline stages of the private cache based (and multi-ported cache based) architecture are instruction fetch (IF), instruction decode (ID), execution (EX), and writeback (WB). In comparison, the

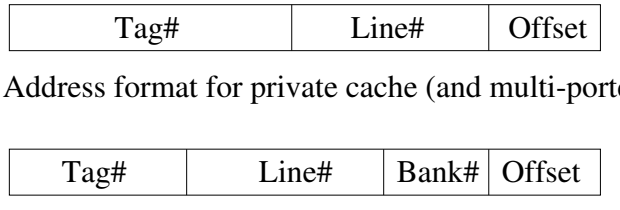

Fig. 2.22. Address formats used by different cache organizations. The bank number field in the CCC format is used to specify the bank to access.

pipeline stages in the CCC-based system are instruction crossbar require (IX), IF, ID, data crossbar require (DX), EX, and WB. In the IX stage, the processor requires the instruction cache access (to a bank) through crossbar. If more than one request target the same bank, only one of them can continue with the next pipeline stage (IF), and all other processors experience a pipeline stall. In the DX stage, if the instruction is load or store, a request to the crossbar is issued.

Figure 2.22 shows the address formats used by the private cache based system (and also the shared multi-ported cache based system) and the CCC based architecture. In the latter, a new field, called the bank number, is added. This field is used to identify the bank to be accessed. Typically, the continuous cache blocks are distributed across different banks in order to reduce the bank conflicts. While CCC is expected to be preferable over other two architectures, it can cause processor stalls when concurrent accesses to the same bank occur. To alleviate this, two optimizations are performed. The first optimization is to use more banks than the processors. For instance, for a target on-chip multiprocessor with four processors, four, eight, sixteen, or thirty-two banks can be used. The rationale behind this is to reduce the conflicts on a given bank.

$E_{total} = E_{L1} + E_{L2}$
(E: Energy)
$E_x = LE_x + DE_x$
(x: L1 or L2)
(LE: Leakage Energy; DE: Dynamic Energy)
$LE = (NA * lea + NS * les)$
(NA: # of active blocks; NS: # of sleep blocks)
(lea: leakage energy per active block per cycle)
(les: leakage energy per sleep block per cycle)
$DE = (N_{hit} + N_{miss} * 2) * de + (N_{hit} + N_{miss}) * ex$
(for crossbar-connected share cache)
$DE = (N_{hit} + N_{miss} * 2 + N_{coherence}) * de$
(for private cache)
(N_{hit}: # of hit; N_{miss}: # of miss)
($N_{coherence}$: # of cache coherence operation)
(de: dynamic energy per access)
(ex: dynamic energy per crossbar access)
$lea = 0.41\ pJ$
$les = 0.04\ pJ$
$de = 104.35\ pJ$ (for 8K L1 cache)
$ex = 7.043\ pJ$ (for 8X16 crossbar)

Fig. 2.23. Formulations used for calculating cache energy consumption.

The second optimization deals with the references to the same block. When two such references occur, instead of stalling one of the processors, the block can be read once and forward to both the processors. This optimization is expected to be more successful when high degree of inter-processor block sharing exists.

2.2.2 Modeling of Energy Consumption of Cache Subsystem

The work in this section focus on the overall cache energy consumption. Both dynamic and leakage energy consumption are considered by using the formulation given in Figure 2.23.

In this work, the cache leakage control strategy proposed in [28] is also implemented on both L1 and L2 caches. The idea is to place a cache line into a leakage control mode if it has not been used for some time. Such a cache line is said to be in the sleep state (otherwise, it is in the active state). In these formulations, the values of lea and les are obtained from the circuit simulation of actual layouts using 70num, 1V Berkeley predictive technology [1]. The value of de is obtained using CACTI 3.0 [74]. The value of ex is obtained by using the model proposed in [29]. The values of N_{hit}, N_{miss}, $N_{coherence}$, NA, and NS for L1 instruction cache, L1 data cache, L2 instruction cache, and L2 data cache are obtained from the simulations as will be discussed later. Finally, NA and NS are the total number of cache blocks in active state and sleep state, respectively, for each clock cycle.

Table 2.4. Benchmarks used in the experiments, their input parameters, and cache energy consumptions (for a private cache based system).

Benchmark	Input	L1 Miss rate (%)	
		I-cache	D-cache
barnes	8192 123 0.025 0.05 1.0 2.0 5.0 0.075 0.25 8	0.085	21.331
ocean1	-n258 -p8	0.150	39.142
ocean2	-n258 -p8	0.026	31.736
radix	-p8 -n2097152 -r1024 -m1048576	0.001	25.751
raytrace	-p8 teapot.env	6.859	16.821
water	1.5e-16 512 3 6 -1 3000 3 0 8 6.212752	0.920	34.139

Benchmark	L1 Energy		L2 Energy	
	Dynamic (mJ)	Leakage (mJ)	Dynamic (mJ)	Leakage (mJ)
barnes	398.0 (40.6%)	583.5 (59.4%)	241.4 (75.9%)	76.7 (24.1%)
ocean1	102.2 (45.4%)	122.8 (54.6%)	90.3 (73.6%)	32.4 (26.4%)
ocean2	99.3 (38.8%)	156.4 (61.2%)	82.7 (68.7%)	37.8 (31.3%)
radix	87.1 (56.8%)	66.2 (43.2%)	38.2 (74.0%)	13.4 (26.0%)
raytrace	91.4 (37.1%)	155.2 (62.9%)	72.3 (70.9%)	29.6 (29.1%)
water	145.5 (35.0%)	270.0 (65.0%)	144.4 (78.2%)	40.3 (21.8%)

2.2.3 Experiments Setup and Benchmarks

In this study, MP_Simplesim [58], a multiprocessor version of SimpleScalar [16], is used as the baseline simulator to simulate an on-chip multiprocessor. In its original form, MP_Simplesim is a functional simulator and simulates only data cache. In order to make it suitable for this study, it is extended by (1) making it cycle accurate, (2) adapting the pipeline stages to the cache architecture simulated, and (3) modifying it to simulate instruction caches (in addition to data caches).

A set of benchmarks is used from the SPLASH-2 suite [84]: barnes, ocean1, ocean2, radix, raytrace, and water. The important characteristics of these benchmarks are listed in Table 2.4. These codes differ from each other in their degree of instruction and data sharing (as pointed out earlier). Each cache configuration is denoted using the quadruple:

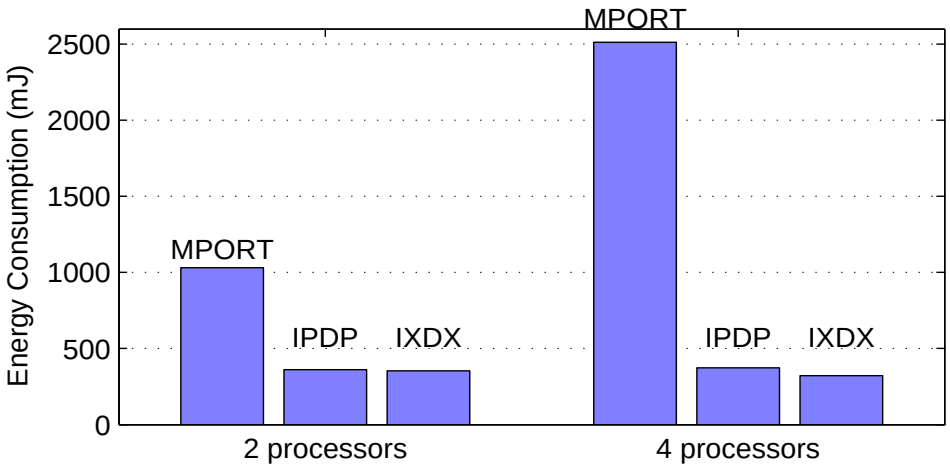

Fig. 2.24. Energy consumption of Ocean1 for three different cache organizations: multi-ported shared cache (MPORT), private cache based organization (IPDP), and proposed CCC based architecture (IXDX).

(processors, banks/private caches, size per bank/private cache, L1 line size)

The four fields in this quadruple represent the number of on-chip processors, the number of L1 banks, size per bank/private cache, and the size of an L1 block (line). It should be observed that, for the private cache-based system, the number of caches is always equal to the number of processors, whereas in proposed CCC-based system the number of banks might be larger than the number of processors. Also, for a given experiment, the number and capacities of the data and instruction caches is the same. As an example, a configuration such as (4,8,4KB,32B) indicates 4 processors and 8 banks (each is 4KB with 32 byte lines).

The initial experiments with the multi-ported shared cache indicated that its energy consumption is much higher than those of the CCC and the private cache-based system. This is due to the excessive access energy associated with increased capacity

Table 2.5. The four cache organizations simulated in this work. In all organizations, L2 is always shared.

	L1 I-cache	**L1 D-cache**
IPDP	private cache	private cache
IXDP	multibank share cache	private cache
IPDX	private cache	multibank share cache
IXDX	multibank share cache	multibank share cache

and multiple ports. As an example, as shown in Figure 2.24, when running one of applications, namely ocean1, the energy consumption of the multi-ported cache was 2.91 times higher than that of CCC for 2 processors system and 7.83 times higher for 4 processors. Therefore, the remainder of this work only focuses on CCC and private cache based configuration.

Table 2.5 lists the cache configurations evaluated in simulations. Unless stated otherwise, (8,16,4KB,32B) is default CCC configuration and (8,8,8KB,32B) is the default for the private cache based architecture. Note that in both the cases, the total cache capacity is the same.

2.2.4 Experimental Results

The (cache) energy consumptions and performance results (execution cycles) of six SPLASH-2 codes for default configurations are given in Figure 2.25 and Figure 2.26, respectively. It can be seen from these results that the best energy behavior is obtained when using the CCC for both instruction and data caches (that is, the IXDX version).

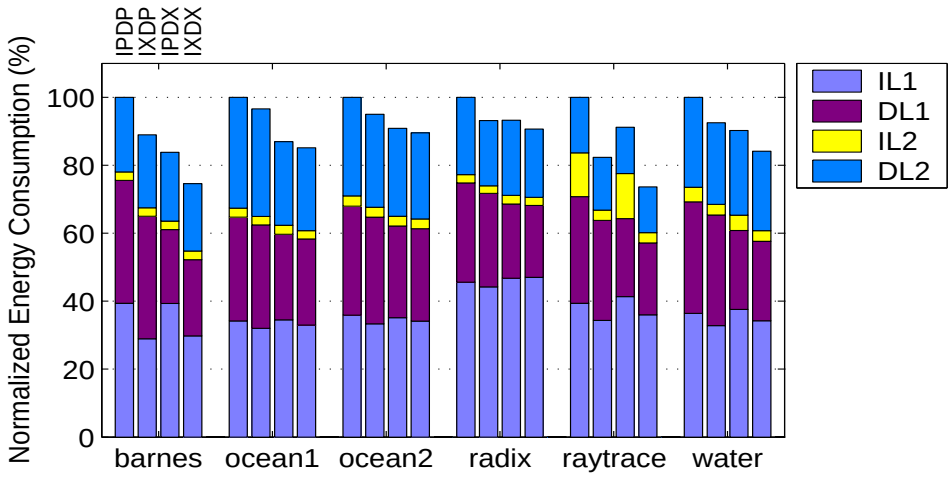

Fig. 2.25. Energy consumptions for different benchmarks on default configuration [(8,16,4KB,32B) for CCC and (8,8,8KB,32B) for the private cache based system]. The results are normalized with respect to IPDP.

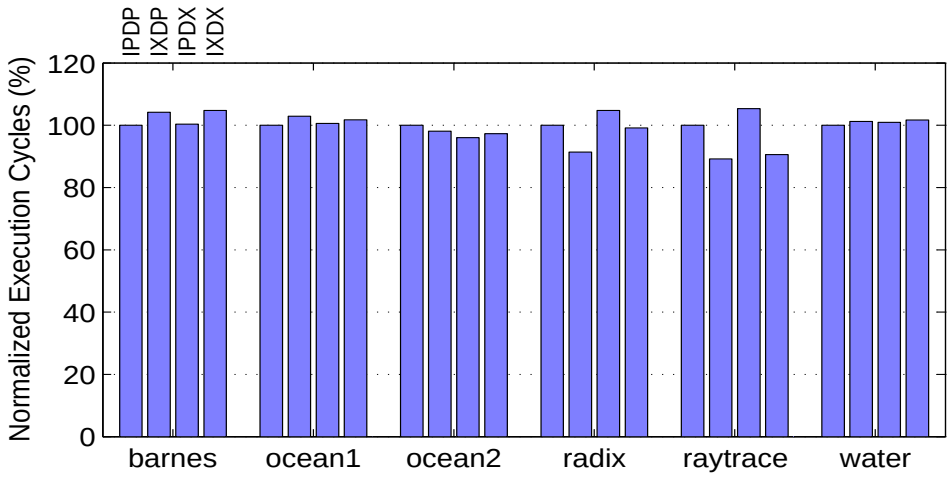

Fig. 2.26. Performance for different benchmarks on default configuration [(8,16,4KB,32B) for CCC and (8,8,8KB,32B) for the private cache based system]. The results are normalized with respect to IPDP.

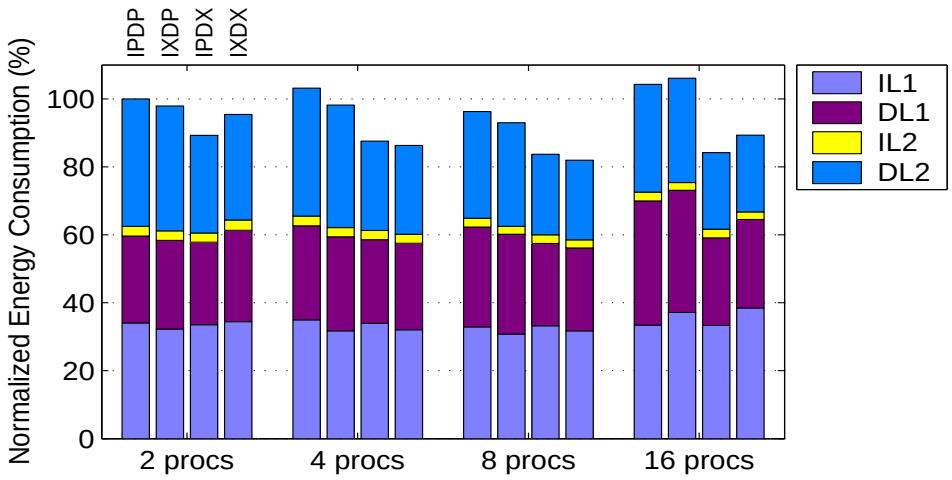

Fig. 2.27. Energy consumptions with different number of processors. The results are normalized with respect to IPDP on two processors.

The percentage energy savings with respect to IPDP range from 9.34% to 26.34%, averaging on 17.04%. In addition, it is observed that a maximum 4.73% degradation in performance (again, as compared to IPDP) when IXDX is employed.

The reasons for very good savings in energy consumption are (1) CCC removes the energy overhead on cache coherence in L1 data cache, (2) CCC has a larger effective L1 cache size than the private cache based system, (3) CCC reduces the miss rate of L1 instruction and data caches and, as a result, (4) it reduces the number of L2 cache access. On the other hand, the main reason for the increase in execution cycles in some benchmarks is the bank conflicts. Despite this, in three of benchmarks (ocean2, radix, raytrace), IXDX slightly improves execution cycles (over IPDP) due to eliminating the duplicates. In the rest of work, only one single benchmark is presented as experiments with other benchmarks led to similar observations.

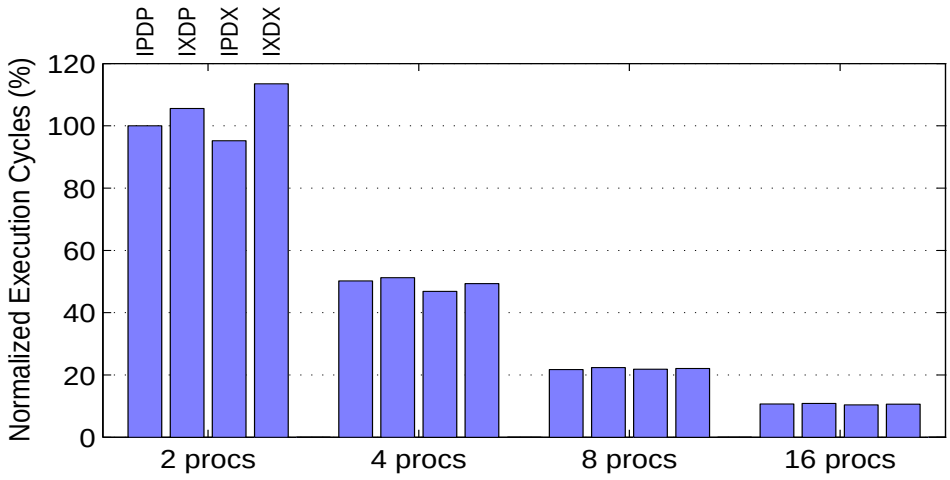

Fig. 2.28. Performance with different number of processors. The results are normalized with respect to IPDP on two processors.

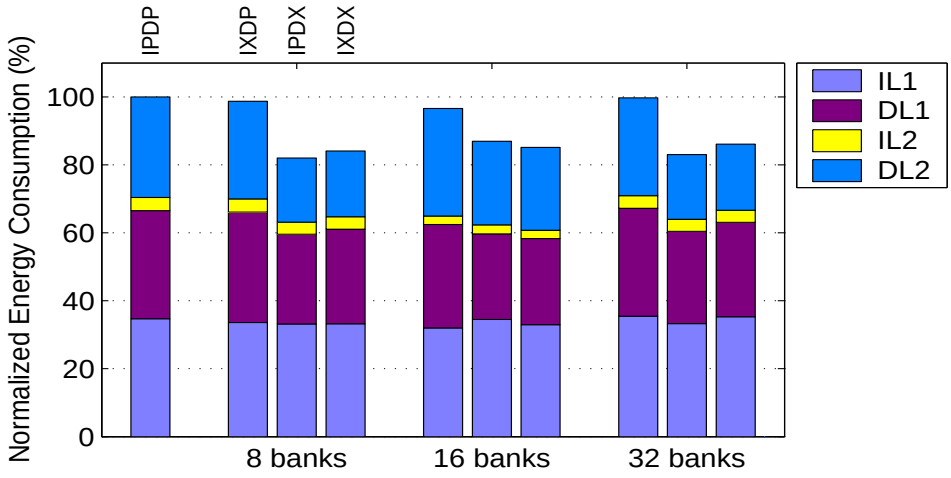

Fig. 2.29. Energy consumptions with different number of banks (8 processors). The results are normalized with respect to the first bar.

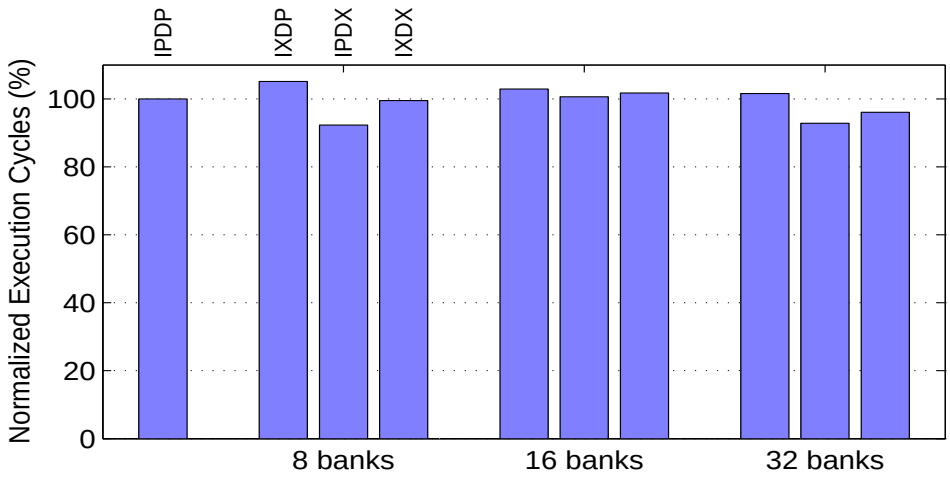

Fig. 2.30. Performance with different number of banks (8 processors). The results are normalized with respect to the first bar.

To study the impact of the number of on-chip processors, experiments are performed with 2, 4, 8, and 16 processors using the ocean1 benchmark. The energy and performance results are shown in Figures 2.27 and 2.28. In the CCC-based configuration, 2x banks is always used for x processors. When the number of the processors increases, the energy consumption of the private cache based system increases and that of the CCC based system remains the same. These results indicate that the effectiveness of proposed strategy increases with the increased number of processors. This is because of three major reasons. First, for the private cache based system, as increasing the number of processors, the dynamic energy spent in coherence activity increases. Second, increasing the number of caches (in the private cache-based system) tends to increase sharing, which in turn reduces the effective cache utilization. Third, increasing the number of banks (in CCC) reduces the number of bank conflicts. This helps both energy and performance.

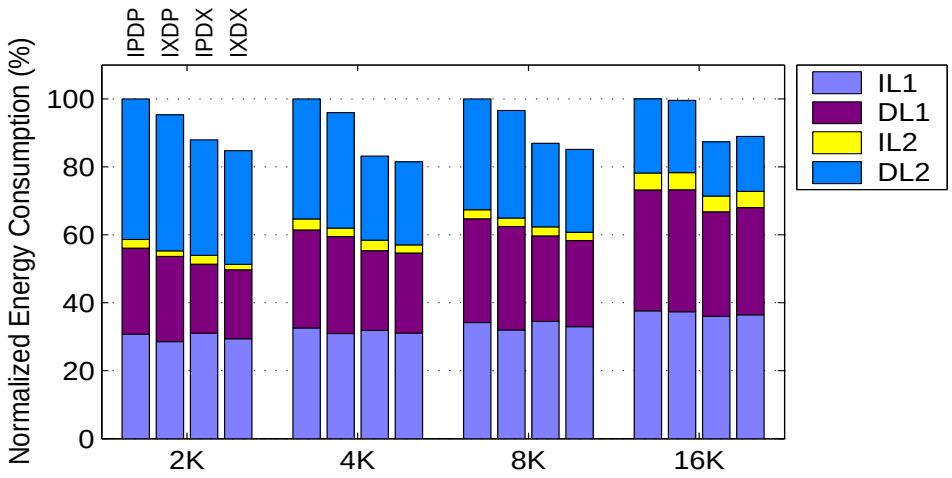

Fig. 2.31. Energy consumptions with different L1 cache size. The results are normalized with respect to IPDP.

The next set of experiments investigate the impact of the number of banks on energy consumption of the CCC based architecture. For this purpose, experiments are performed with 8 banks, 16 banks, and 32 banks using the ocean1 benchmark. The results given in Figures 2.29 and 2.30 clearly indicate that while there is not too much difference in energy behavior, CCC results in better execution time when using a larger number of banks. The main reason for this is the reduced number of bank conflicts. It should be mentioned that while increasing the number of banks seems to be beneficial from the performance perspective, it may not be a very scalable approach as large number of banks also poses an area and interconnection cost problem. This last issue is beyond the scope of this work.

In next set of experiments, L1 capacity (size) is modified. The values used in experiments are 2KB, 4KB, 8KB, and 16KB. All other parameters are the same as in the default configuration. The results are given in Figures 2.31 and 2.32. The observation is

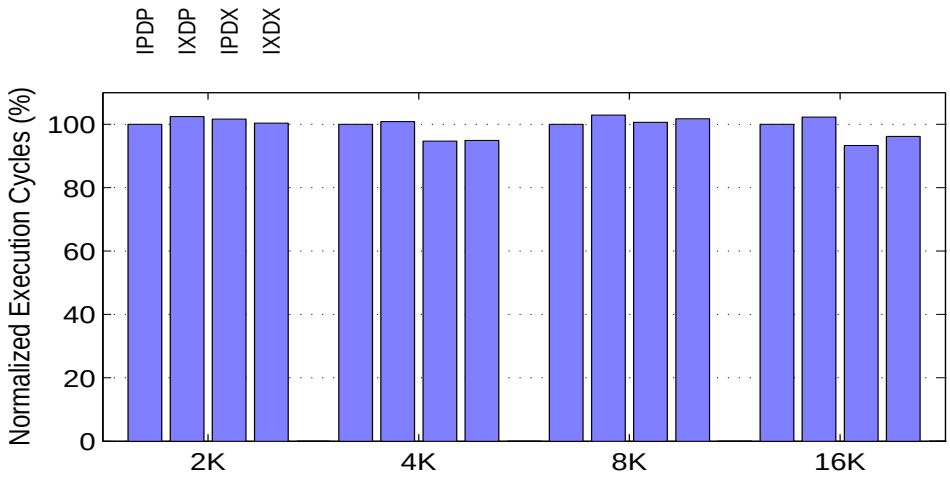

Fig. 2.32. Performance with different L1 cache size. The results are normalized with respect to IPDP.

that the CCC-based architecture still remains better than the private cache based system when L1 cache size is increased. However, if the L1 cache size is made large enough to capture the working set of the application, the difference between CCC and the private cache based system reduces. In experiments, this happens with a 16KB cache size.

In last set of experiments, the L1 cache line size is modified. The values in experiments include 8B, 16B, 32B, and 64B, and the energy and performance results are depicted in Figure 2.33 and Figure 2.34. It can be seen from these results that increasing the cache line size increases the chances for multiple accesses to the same cache line. Thanks to the optimization discussed in previous subsection, which reduce these types of conflicts significantly. As a result, the performance of the CCC-based architecture improves, and an improvement in energy behavior is also observed.

These experimental results indicate that proposed CCC-based cache architecture takes advantage of increased number of on-chip processors and of banks, increased cache

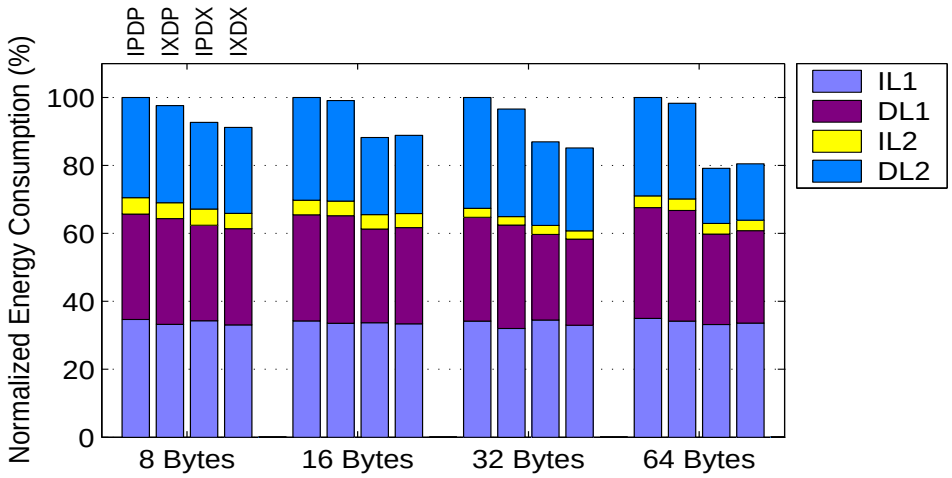

Fig. 2.33. Energy consumption for different L1 cache line size. The results are normalized with respect to IPDP.

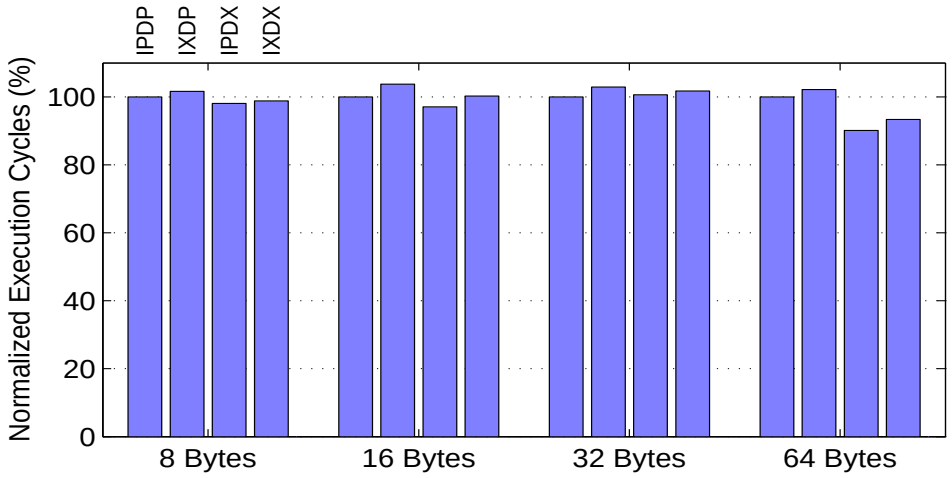

Fig. 2.34. Performance for different L1 cache line size. The results are normalized with respect to IPDP.

capacity and line size. Since these are also the trends that can be observed in the architecture area, the proposed cache architecture is expected to be even more successful in the future.

Chapter 3

Reducing Impacts of Crosstalk Noise

Traditional designs of on-chip interconnects determine the delay associated with data transmission based on the worst-case data transition pattern possible. However, the impact of crosstalk on the delay is data-dependent [76] and different combinations of transition directions of a specific wire and its adjacent wires induce different delays. Operating assuming worst-case delay can be overly pessimistic for a significant portion of the data transmissions. This pessimism is undesirable because it increases cycle time, area and power [42].

One approach that has been recently proposed to eliminate the pessimistic safety margins due to factors such as (but not limited to) data-dependent latencies is [27]. Here, the supply voltage is tuned below the normal operating voltage to meet the average case delay in order to save energy. Possible errors are monitored. When delay induced by a particular data pattern exceeds that supported by this supply voltage, error correction is required. The essence of their approach is to trade-off the energy gain from voltage scaling and the energy overhead incurred from dynamic error correction.

In contrast to this approach that adjusts the voltages reactively to correct errors when delays exceed that of the average case, a crosstalk-aware interconnect is designed that uses a faster clock and dynamically controls the number of cycles required for transmission based on the estimated delay of the data pattern to be transmitted.

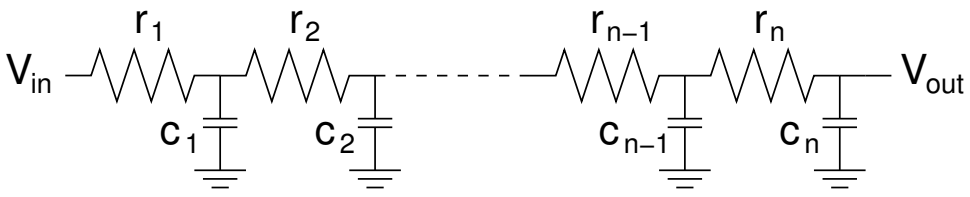

Fig. 3.1. Distributed RC model.

3.1 Model of Propagation Delay and Crosstalk

Due to the capacitance and resistance on the wires, signal switching incurs a propagation delay. Distributed RC model, shown in Figure 3.1, is one of the most frequently used model to analyze the delay behavior of a resistive wire [70]. Assuming that the length of the wire is L, the resistance of the wire is R per unit length, and the capacitance of the wire is C per unit length, Equation 3.2 can be used to calculate the first-order time-constant of propagation delay, the Elmore delay [70]. Equation 3.2 shows that the propagation delay of a wire is proportional to the capacitance induced by the wire.

$$r_1 = r_2 = \cdots = r_{n-1} = r_n = R \cdot \frac{L}{n}; \quad c_1 = c_2 = \cdots = c_{n-1} = c_n = C \cdot \frac{L}{n} \tag{3.1}$$

$$T_{delay} = \sum_{i=1}^{n} c_i \sum_{j=1}^{i} r_j = R \cdot \frac{L}{n} \cdot C \cdot \frac{L}{n} \cdot \sum_{i=1}^{n} i = RCL^2 \frac{n+1}{2n} \tag{3.2}$$

When looking at a group of wires in on-chip interconnects, the various capacitances associated with wires are depicted in Figure 3.2(a). Here, C_{couple} is the capacitance between two adjacent wires and C_{ground} is the capacitance between a wire and the

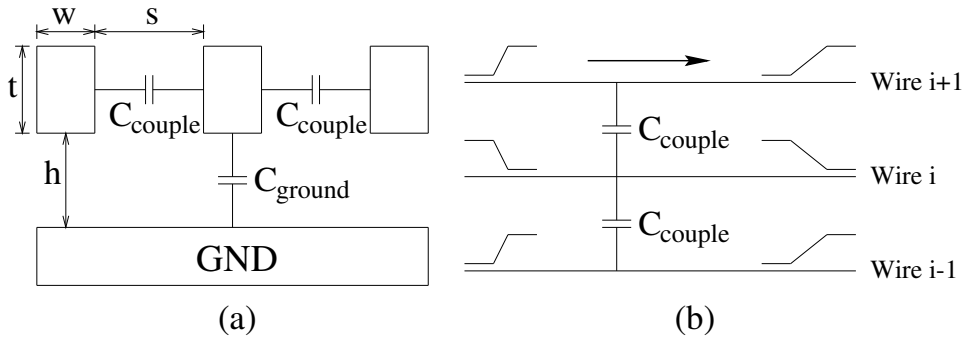

Fig. 3.2. (a) Wire capacitances (b) Worst-case propagation delay for wire i occurs when wire i transitions in the opposite direction of adjacent wires $i+1$ and $i-1$.

ground. The values of C_{couple} and C_{ground} are determined by the technology parameters such as wire width (w), spacing (s), wire thickness (t), height (h), and the length of the wires. As feature size scales down to deep sub-micron, the coupling capacitance contributes to a larger portion of the total capacitance.

The magnitude of coupling capacitance is dependent on the pattern of transition direction of the target wire and its two adjacent wires. The triplet (d_{i-1}, d_i, d_{i+1}) is used to represent the transition direction for adjacent wires $i-1$, i, and $i+1$. Here, the target wire for crosstalk analysis is the wire i. '↑', '↓', and '-' are used to represent transitions from 0 to 1, from 1 to 0, and no transition, respectively. The relation between C_{total} and transition patterns of these three wires are analyzed in [76] and are shown in Table 3.1 (capacitance for the boundary wires can be found in [76]).

$$T_{delay-i} = \begin{cases} 0, & i = 1 \\ RCL^2(n+1)/2n = (C_{ground} + (i-2) * C_{couple}) * R_{total}, & 2 \leq i \leq 6 \end{cases} \tag{3.3}$$

Table 3.1. Capacitance variations based on transition patterns.

Group	C_{total}	**Transition Patterns**			
1	0	$(-,-,-)$ $(-,-,\uparrow)$ $(\uparrow,-,\uparrow)$	$(-,-,\downarrow)$ $(\uparrow,-,\downarrow)$	$(\uparrow,-,-)$ $(\downarrow,-,\uparrow)$	$(\downarrow,-,-)$ $(\downarrow,-,\downarrow)$
2	C_{ground}	$(\uparrow,\uparrow,\uparrow)$	$(\downarrow,\downarrow,\downarrow)$		
3	$C_{ground}+C_{couple}$	$(-,\uparrow,\uparrow)$	$(-,\downarrow,\downarrow)$	$(\uparrow,\uparrow,-)$	$(\downarrow,\downarrow,-)$
4	$C_{ground}+2C_{couple}$	$(-,\uparrow,-)$ $(\uparrow,\uparrow,\downarrow)$	$(-,\downarrow,-)$ $(\uparrow,\downarrow,\downarrow)$	$(\downarrow,\uparrow,\uparrow)$	$(\downarrow,\downarrow,\uparrow)$
5	$C_{ground}+3C_{couple}$	$(-,\uparrow,\downarrow)$	$(-,\downarrow,\uparrow)$	$(\uparrow,\downarrow,-)$	$(\downarrow,\uparrow,-)$
6	$C_{ground}+4C_{couple}$	$(\uparrow,\downarrow,\uparrow)$	$(\downarrow,\uparrow,\downarrow)$		

Based on the total capacitance, the delay of the target wire can be calculated using Equation 3.3, in which $T_{delay-i}$ is the propagation delay for transition patterns in Group i and R_{total} is used to summarize remaining parameters except capacitance. The patterns in the same group incur the same delay for transmission. The delay increases from minimum to maximum as moving from Group 1 to Group 6. The worst-case delay is incurred by the transition pattern in Group 6, which happens when the direction of the transition of the target wire is opposite to those of its two adjacent wires (See Figure 3.2(b)). The worst-case delay is determined by $(C_{ground}+4C_{couple})*R_{total}$. Since transition patterns in Group 5 and Group 6 are the most problematic ones, they are the focus of the proposed approach. Based on the delay analysis of each wire, the delay of an interconnect with a group of wires is computed as the maximum delay among these wires.

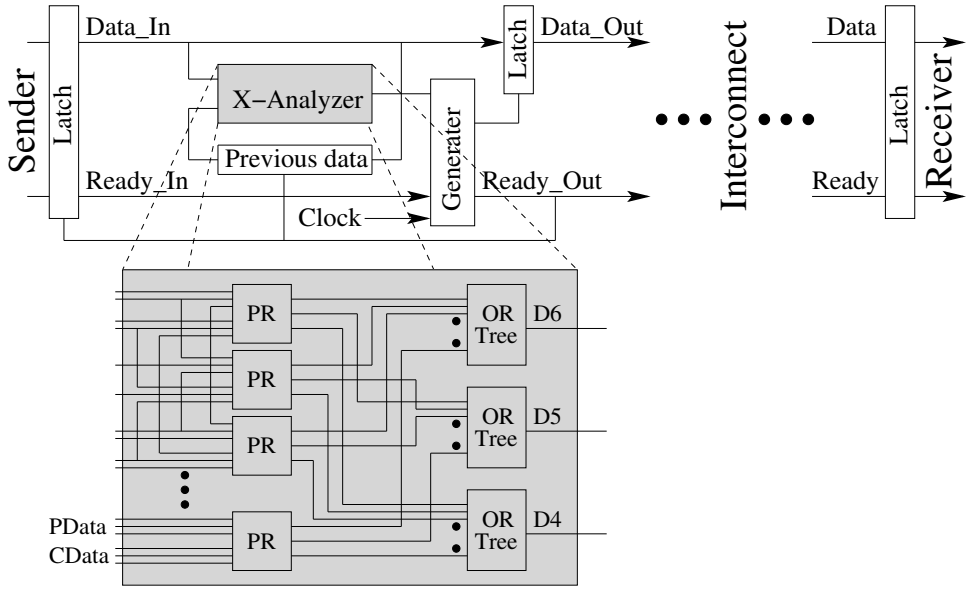

Fig. 3.3. Crosstalk analyzer implementation.

3.2 Crosstalk-Aware Interconnect

The original interconnect analyzed in this section is a 33-bit bus between the sender and the receiver. Signals transmitted on the interconnect include 32-bit data and 1-bit control signal, *Ready*, which indicates the availability of the data.

It is clear from Table 3.1 that the propagation delay of the wires is determined by the changes in value transmitted in the previous and current cycles. Therefore, an extra circuit, called Crosstalk-Analyzer (or *X-Analyzer*), is incorporated in the sender side of the bus to capture the transitions on the bus and identify the delay grouping, as shown in Figure 3.3. An extra register is needed to store the previous data at the sender portion of the bus. When the new data arrives, the *X-Analyzer* identifies the transition pattern between previous data and current data and determines the corresponding delay group

(See Table 3.1) for each wire. The wire that belongs to the largest group determines the delay of the entire bus. Therefore, the impact of crosstalk is captured by the sender before the transmission begins. This in turn provides the opportunity for the system to adjust for the different degrees of crosstalk impact.

Inside the *X-Analyzer*, *PR* stands for the Pattern Recognition unit, which determines whether the adjacent 3-bit transition pattern belongs to Group 4, 5 or 6.

Note that the number of distinct groups identified is limited to reduce the implementation complexity. There is one *PR* for each wire of the bus. Each *PR* has two group of 3 inputs, one from current data and one from previous data. Each 3 inputs are the signal of target wire and its two adjacent wires. Three outputs of each *PR* indicate whether the delay belongs to group 4, 5, or 6, respectively. In this approach, the *OR Trees* are used to combine the signals indicating the patterns recognized for each of the 32 data wires into final signals $G4$, $G5$ and $G6$, which indicate whether patterns in Group 4, 5 or 6 exist in the whole 32-bit data. Each *OR Tree* is implemented by hierarchy connecting five-level thirty-one 2-input OR gates together (Another possible solution is using dynamic logic implementation to improve the performance and reduce the area). Based on the signals $G4$, $G5$ and $G6$ from *X-Analyzer*, the *Generater* issues *Ready* signal with different cycles, which will be discussed in the next.

The bus clock cycle is normally designed for the worst-case scenario of crosstalk determined by $(C_{ground} + 4C_{couple}) * R_{total}$. But there might be many transmissions which incur a delay that is less than that of the worst case. Therefore, using a single long clock cycle (See Figure 3.4(a)) in the original bus method (denoted ORI) may not be performance efficient. Hence, the proposed approach uses a variable cycle transmission

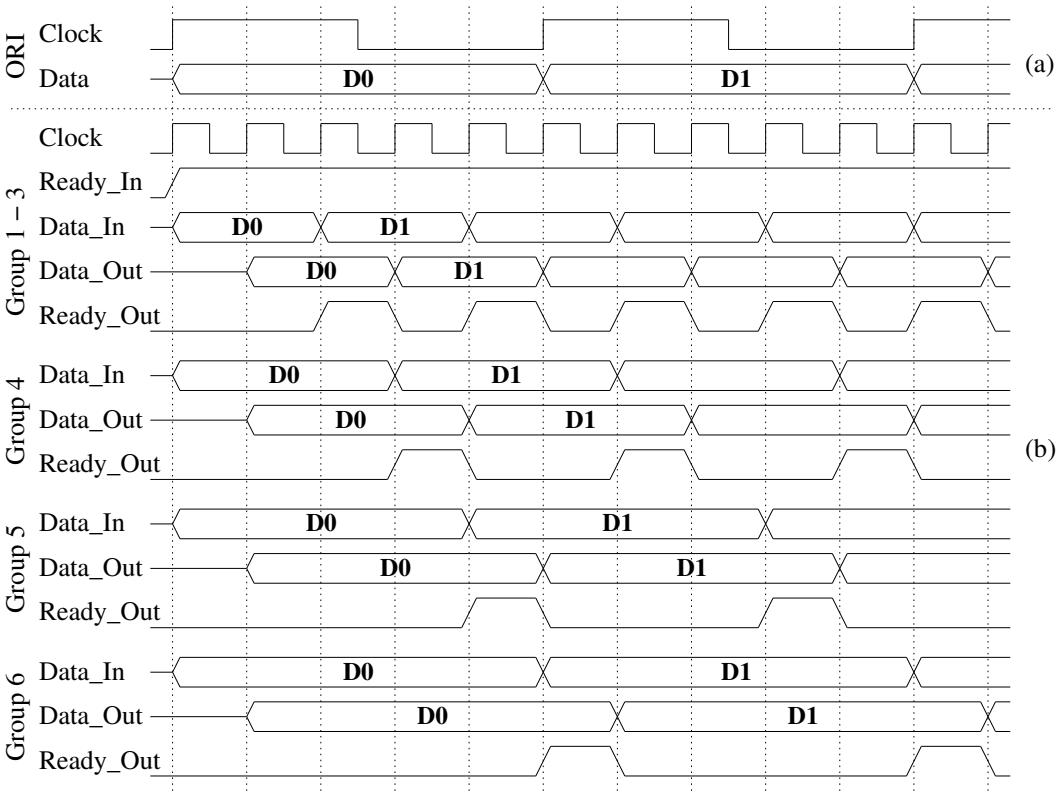

Fig. 3.4. Timing sequence for (a) original pessimism method & (b) Variable cycle transmission method.

whose basic idea is to use multiple short clock cycles instead of the original long clock cycle, where the number of clock cycles is determined by the *X-Analyzer* based on the delay group (See Figure 3.4(b)).

The sender of the original bus has 32 wires for data and 1 wire for the control signal *Ready*. When the *Ready* signal is high, the receiver of the bus knows that the data is ready. In the implementation shown in Figure 3.3, component *Generater* is inserted between *Ready_In* and *Ready_Out*. Based on the output of *X-Analyzer*, *Generater* can insert several idle cycles before issuing active-high *Ready_Out* signal. By controlling *Ready_Out* signal, the cycles used to transmit data can be changed dynamically. Therefore, the proposed approach is denoted as DYN. The timing sequences for the data transitions belonging to four different categories (for Group 1-3, Group 4, Group 5 and Group 6) are shown in Figure 3.4(b).

The clock cycle $T_{clk-ori}$ in the original method (ORI) is at least $(C_{ground} + 4C_{couple}) * R_{total}$, independent of the group to which the transition belongs. In contrast, a short clock cycle of $T_{clk-dyn}$ is used in the proposed approach. In [54], $T_{clk-dyn}$ is assigned as $(C_{ground} + C_{couple}) * R_{total}$ in a coarse granularity. For the transition in Group i, $(i-2)$ clock cycles are required for $3 \leq i \leq 6$, or 1 clock cycle is required for $i = 1$ and $i = 2$. In this assignment, although the transmission time used for transition patterns in Group 1 to 3 is less than or equal to that in original interconnect, the transmission time used for Group 4 to 6 is larger than that in original interconnect. This brings negative impacts on the performance of my scheme. Therefore, a new method is proposed to calculate the unit clock cycle in a fine granularity. Assuming that the

Table 3.2. Comparison of transmission time.

	Coarse granularity	Fine granularity
Clock cycle	$(C_{ground} + *C_{couple}) * R_{total}$	$(C_{ground} + 4 * C_{couple}) * R_{total}/5$
Group 1	$(C_{ground} + *C_{couple}) * R_{total}$	$2 * (C_{ground} + 4 * C_{couple}) * R_{total}/5$
Group 2	$(C_{ground} + *C_{couple}) * R_{total}$	$2 * (C_{ground} + 4 * C_{couple}) * R_{total}/5$
Group 3	$(C_{ground} + *C_{couple}) * R_{total}$	$2 * (C_{ground} + 4 * C_{couple}) * R_{total}/5$
Group 4	$2 * (C_{ground} + *C_{couple}) * R_{total}$	$3 * (C_{ground} + 4 * C_{couple}) * R_{total}/5$
Group 5	$3 * (C_{ground} + *C_{couple}) * R_{total}$	$4 * (C_{ground} + 4 * C_{couple}) * R_{total}/5$
Group 6	$4 * (C_{ground} + *C_{couple}) * R_{total}$	$(C_{ground} + 4 * C_{couple}) * R_{total}$

unit clock cycle is x, following conditions must be fulfilled for each group of transition patterns:

$$\begin{cases} x \geq C_{ground} * R_{total} \\ 2x \geq (C_{ground} + 1 * C_{couple}) * R_{total} \\ 3x \geq (C_{ground} + 2 * C_{couple}) * R_{total} \\ 4x \geq (C_{ground} + 3 * C_{couple}) * R_{total} \\ 5x \geq (C_{ground} + 4 * C_{couple}) * R_{total} \end{cases}$$

If C_{couple} is larger than C_{ground} per unit length, we can get $x \geq (C_{ground} + 4 * C_{couple}) * R_{total}/5$. Therefore, the transmission time for all groups of transition patterns are less than or equal to those in original interconnect and hence, it can improve the performance of the proposed approach significantly. In current technology generation and below, it is true that C_{couple} is larger than C_{ground}. The transmission time for each group of transition patterns in coarse granularity and fine granularity of the proposed approach are shown in Table 3.2.

When data and *Ready_In* signal are issued from the sender, the *X-Analyzer* determines the group to which the current transition belongs and correspondingly sets the

outputs G4, G5 and G6. In the next cycle, the data (Dx) is placed on the bus and the signals G4, G5 and G6 determine when *Ready_Out* is placed on the bus. The delay between when the data and *Ready_Out* are placed on the bus allows the necessary number of cycles for the data on the bus to settle at the receiver side. The *Ready_Out* signal is separated with other data wires by inserting a shielding wire. Therefore, *Ready_Out* signal reaches the receiver side in a single bus cycle immaterial of the data transmitted. For data patterns from Group 4, Group 5, and Group 6, the *Ready_Out* signal is placed three, four, and five cycles after the data is placed on the bus. Otherwise, *Ready_Out* is placed on the bus two cycles after (shown in Figure 3.4(b)). Once the *Ready_Out* is placed on the bus, the next data to be transmitted is fed into the *X-Analyzer* and the process repeats. Although the *X-Analyzer* incurs a one cycle latency to identify the transition pattern, this latency is overlapped with the cycle that it takes for the *Ready_Out* signal to traverse the bus.

Note that the data-dependent transmission makes the data transfer rate variable. In order to control the next input being latched before the current transfer is complete, the output from the *Generater* is fed to gate the latch at the sender side until data transfer is complete. The receiver side remains similar and only needs to recognize the *Ready_Out* signal.

3.3 Experiments Configuration

The crosstalk analyzer is designed in VHDL and synthesized using the Synopsys Design Compiler with a 0.16um technology library. The parameters of *width*, *spacing*,

Table 3.3. Parameters of different metal layers and corresponding capacitances.

Layer	Height (um)	Width (um)	Spacing (um)	Thickness (um)	C_{ground} (fF/mm)	C_{couple} (fF/mm)
MET 1	0.52	0.26	0.26	0.5	34.671	103.871
MET 2-3	0.52	0.26	0.26	0.5	34.671	80.379
MET 4	1.04	0.52	0.52	0.7	34.317	60.990
MET 5	1.04	0.52	0.52	0.7	34.317	80.478

thickness, and *height* of each metal layer are shown in Table 3.3. The Berkeley interconnect model [1] is used to calculate the ground and coupling capacitance of the interconnects that is shown in the last two columns of Table 3.3. It is clearly shown that coupling capacitance are larger than ground capacitance significantly for every layer in 0.16um technology. Later, the parameters of layer 5 is used in experiments.

SimpleScalar 3.0 [16] and the SPEC2000 CINT [5] benchmark suite are used to simulate the performance of different on-chip buses between the processor datapath and L1 I-cache/D-cache and between the L1 and L2 caches. The results of two typical buses, address bus of IL1 cache and data bus of DL1 bus are shown in following subsections. All benchmarks have been executed for 100 million instructions (after skipping 300 million instructions). The parameters of the base configuration are the same as those shown in Table 2.2 in Section 2.1.

In addition to the original transmission method (ORI) and DYN method, three other methods are also implemented, which are Crosstalk Prevention Coding (CPC), Double Spacing (DBS), and Shielding method (SHD), for comparison. These three methods are also used to reduce the impact of crosstalk and are shown in Figure 3.5 along with ORI.

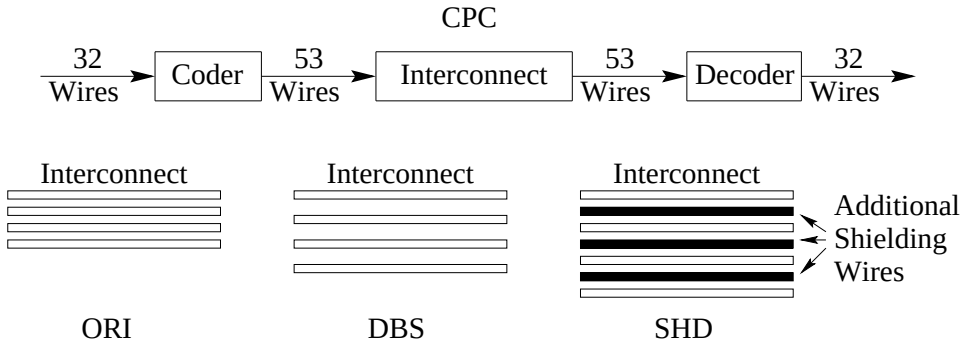

Fig. 3.5. Original interconnect, Crosstalk Prevention Coding, Double Spacing, and Shielding.

Crosstalk Prevention Coding method (CPC) [81] codes the source data into transmission data via extra circuitry in the form of encoder/decoder on the sender/receiver, respectively. There are more wires for transmitting coded data than that for source data. In the transmission data, there are no adjacent transitions in the opposite direction. Hence, all the transition patterns in Groups 5 and 6 are eliminated. For a 32-bit source data, 46 wires are required for data transmission based on the methodology proposed in [81]. For implementation purposes, the wires are divided into multiple smaller groups and build the encoder and decoder for these smaller groups. The different groups of the wires are separated by shielding wires. Specifically, every 3 bits are coded into 4 bits. Hence, a total of 53 wires are required for transmitting the original 32-bit data. The mapping between source data and code data is shown in Table 3.4.

Double Spacing method (DBS) is explored in [11]. Increasing the space between adjacent wires can reduce the coupling capacitance and the total capacitance. In experiments, the space between the adjacent wires was doubled, which means parameter *spacing* is changed to 1.04um for metal layer 5. When the spacing between two adjacent

Table 3.4. Coding for CPC scheme.

Source Data	000	001	010	011	100	101	110	111
Code Data	0000	0001	0100	0101	0111	1100	1101	1111

Table 3.5. Characteristics of different data transmission methods.

	Number of Wires	**Normalized Cycle Time**	**Normalized Area**			**Extra Energy**
			2mm	**5mm**	**10mm**	
ORI	33	1.00	100	100	100	-
CPC	55	0.75	172	170	169	0.91 mW
DYN	34	0.20	119	109	106	3.35 mW
DBS	33	0.62	149	149	149	-
SHD	65	0.55	198	198	198	-

wires increases, ground capacitance increases a little due to fringe effects and coupling capacitance decreases significantly. Consequently, total capacitance and transmission delay also decrease significantly.

Shielding method (SHD) [11] inserts one V_{dd} or *Ground* wire between every two wires. Since there is no transition on the shielding wires, transition patterns in Group 5 and Group 6 are eliminated. Therefore, the worst-case delay reduces to $(C_{ground} + 2C_{couple}) * R_{total}$.

3.4 Experimental Results

The main characteristics of the different data transmission methods are shown in Table 3.5. The number of the wires required for each method is listed in the second column. Original interconnect (ORI) includes 32 data wires and 1 control wire. Since CPC converts every 3 bits of source data into 4 bits code data and one shielding wire

is for every 4 wires, there is 55 wires totally in CPC. The DYN scheme utilizes one more shielding wire to separate *Ready_Out* signal with data wires. DBS remains the same number of wires as ORI, but the space between every two wires is doubled. SHD requires 32 shielding wires for original 33 wires. Clock cycle time of each scheme is normalized to the cycle time of ORI scheme and is shown in the third column.

Three different wire lengths of 2mm, 5mm and 10mm are modeled to capture respectively short, medium and long wires in the target technology. Short wires are representative of the longest connection within a module; medium wires can model on-chip buses such as the data bus and instruction bus from the datapath to L1 caches; and long wires are representative of on-chip multiprocessor buses. The overall bus area for the different schemes are normalized with respect to the area occupied by the bus in ORI scheme of the same length and are shown in columns 4 to 6 in Table 3.5.

It can be observed that DYN incur area overhead 19%, 9%, and 6% for 2mm, 5mm, and 10mm bus, respectively, since DYN needs one more shielding wire and extra circuitry for the crosstalk analyzer. Shielding of the ready signal in DYN scheme incurs a fixed percentage area overhead immaterial of the wire length. As the crosstalk analyzer circuit in DYN remains the same when interconnect length (and consequently interconnect area) increases, the relative area overhead incurred by DYN decreases with the length of the interconnects. On the other hand, in the schemes that use more wires or wire spacing, the relative area overhead remains the same with the change in length. This means that the area overhead of the DYN method would decrease for the longer interconnection. In comparison, the SHD scheme incurs the maximum area overhead (98%) among the schemes considered due to the most extra shielding wire used. CPC incurs area overhead

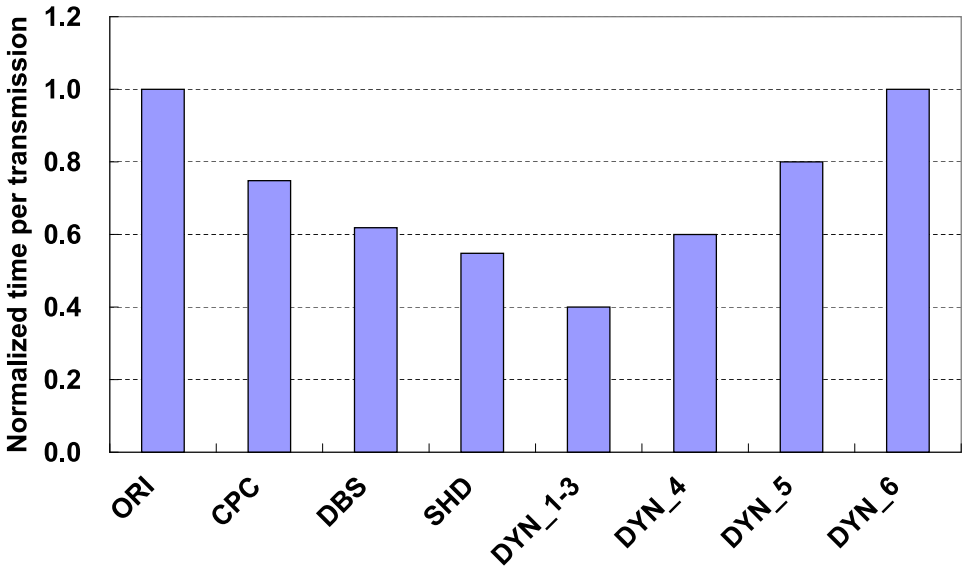

Fig. 3.6. Transmission time.

72%, 70%, and 69% for 2mm, 5mm, and 10mm bus, respectively, due to more extra wires for coded data and extra circuits for encoding and decoding. DBS have fix area overhead 49% due to increasing space between wires. Energy overheads for methods of CPC and DYN are also listed in the last column.

Figure 3.6 shows the normalized time required for one transmission in different schemes. Since the cycle times of ORI, CPC, DBS, and SHD are designed for the worst-case scenario, it is different for transmission in different groups. Although both CPC and SHD eliminate the transitions in Group 5 and Group 6, CPC requires extra time for encoding and decoding and, hence, incurs longer time for transmissions compared to SHD. Shorter transmission time in DBS is due to the smaller total capacitance in the worst-case scenario. The DYN scheme uses different number of cycles for transmissions in different groups; specifically, from two to five cycles for transmissions in Groups 1-3,

4, 5, and 6. The transmission times required for Group 1-4 are significantly smaller than the original transmission time. Therefore, the overall benefits of the DYN approach result from the fact that a large portion of the transmissions belong to Groups 1 to 4. Moreover, techniques that reduce the percentage of transmissions in Group 5 and Group 6 can enhance the effectiveness of proposed variable cycle transmission scheme.

Since the DYN scheme exploits the difference in the groups to which the transition patterns belong, the distributions of different transition patterns is analyzed on different on-chip buses. The results of L1 instruction cache (IL1) address bus and L1 data cache (DL1) data bus are shown in Figures 3.7 and 3.8, respectively. The reason to pick up these two busses is that the data on the IL1 address bus are high-sequential and data on DL1 data bus are more random.

Distribution of transition patterns among different buses varies due to the intrinsic characteristics of different buses. For IL1 address bus, since the program counter increases by one most of the time, a large portion of the transmission patterns belong to Groups 4 and 5 (averaging at 42.87% and 44.45%, respectively) and the distribution of Group 6 is only 1.01%. Since there are only two possible data transitions that belong to Group 2: from $(000\cdots000)$ to $(111\cdots111)$ or from $(111\cdots111)$ to $(000\cdots000)$, the distribution of Group 2 is very rare in all the on-chip buses. In IL1 address bus, the distributions of different transition groups are very similar across different benchmarks.

For the DL1 data bus, the data accessed is more random than other on-chip buses, and is mostly decided by the intrinsic characteristics of the benchmarks. Therefore, diversity is observed among the distributions across the different benchmarks. The

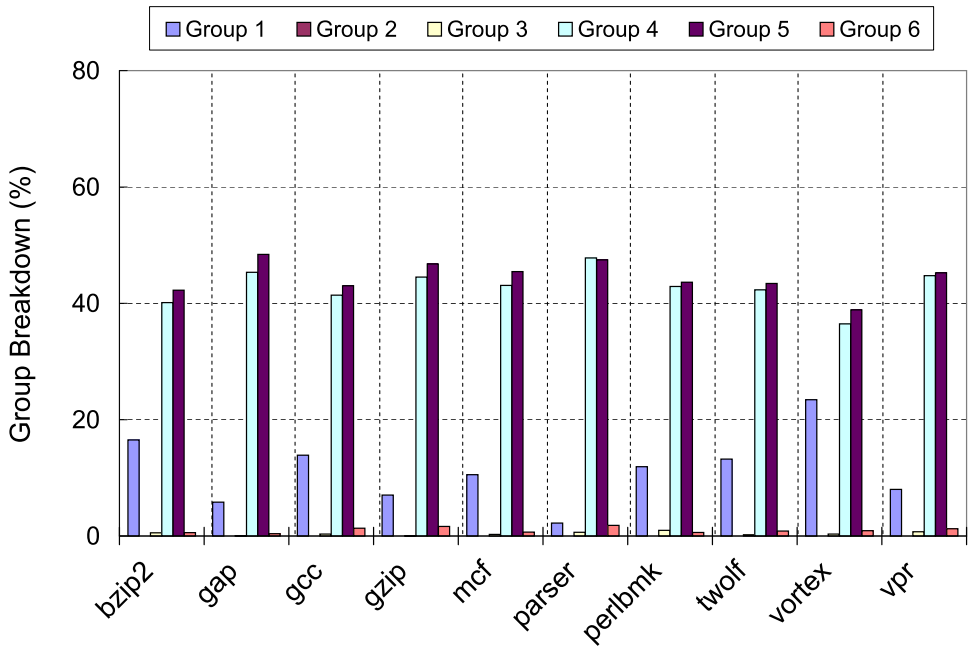

Fig. 3.7. Distribution of transition patterns in original interconnect (IL1 address bus).

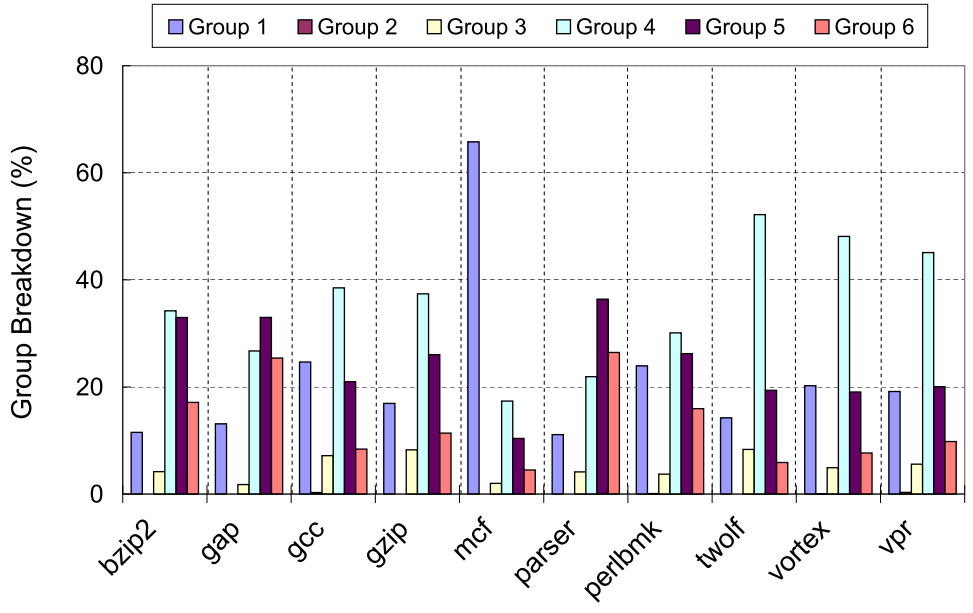

Fig. 3.8. Distribution of transition patterns in original interconnect (DL1 data bus).

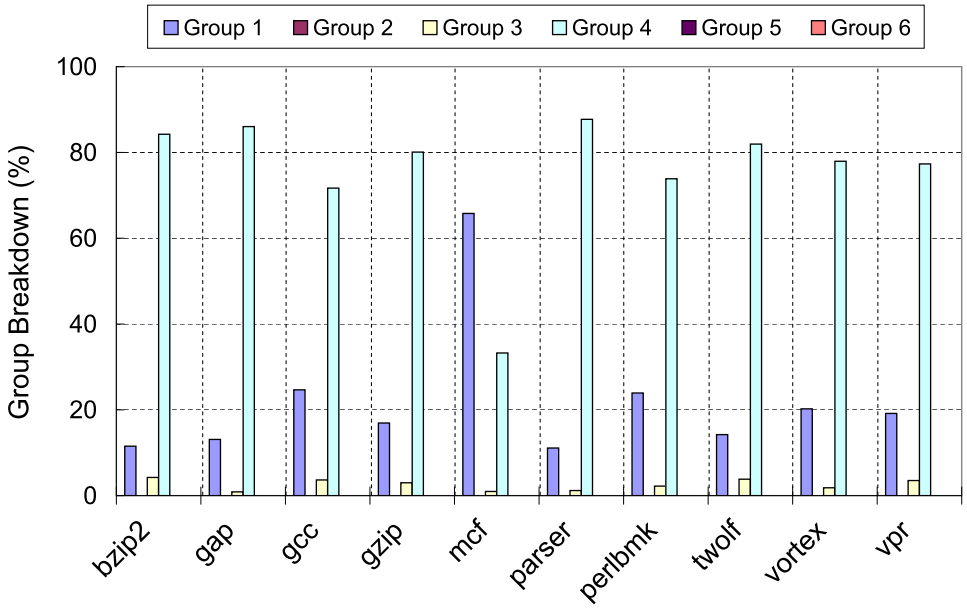

Fig. 3.9. Distribution of transition patterns in CPC scheme (DL1 data bus).

average distributions are 22.06%, 0.07%, 5.00%, 35.16%, 24.45%, and 13.25% for Group 1 through Group 6.

Based on the distribution of above two different on-chip buses, it is observed that in most cases, the worst case (Group 6) is not the dominant one. Instead, the distributions of Groups 4 and 5 are dominant. Groups 2 and 3 are quite rare. The diversity of the distributions across different groups indicates an opportunity for improvement using proposed variable cycle transmission scheme.

Since CPC transmits coded data rather than original data, the distribution of transition patterns in CPC scheme is significantly different from that in ORI scheme, which is shown in Figure 3.9. CPC removes all the transitions in Group 5 and Group 6. Hence the transitions in Group 4 are dominant. The transitions in Groups 1 to 4 are 22.06%, 0%, 2.53%, and 75.41%, respectively.

Figures 3.10 give the results of bus performance for different transmission schemes on DL1 data bus (same result is observed on IL1 address bus). The performance is measured by the number of cycles spent on the target buses. Since the cycle times of different schemes are different (see Table 3.5), the number of cycles in different schemes is normalized with respect to the cycle time in proposed DYN scheme.

The performance improvements due to CPC, DYN, DBS, and SHD methods are 26.2%, 35.2%, 41.0%, and 46.2%, respectively, for DL1 data bus (as compared to ORI). Since SHD places a V_{dd} wire between every two adjacent wires, the impact of crosstalk on the delay is reduced significantly. Therefore, the performance of SHD is the best among all the methods in experiments. In comparison, the DYN scheme behaves very differently across the benchmarks. DYN is better than all other methods for mcf and worse than CPC for $parser$. From Figure 3.8, we can find that the reason is that Group 5 and 6 transition patterns are dominant in $parser$, but Group 1 patterns are dominant in mcf. This means that the performance improvement depends on the distribution of the transition patterns.

Recalling that the area overhead of different schemes from Table 3.5, it is observed that, for a 10mm long bus, the CPC improves performance by 26.2% with 69% area overhead, DBS improves performance by 41.0% with 49% area overhead, SHD improves performance by 46.2% with 98% area overhead, and DYN improves performance by 35.2% with 6% area overhead. Therefore, the DYN scheme is better than the CPC scheme from both performance and area overhead perspectives for the benchmarks used. Compared with DBS and SHD, the DYN scheme provides a more efficient way to improve

performance with smaller area overhead. In other words, DYN is the best considering the performance/area metric.

3.5 Combining with Other Schemes

Bus coding schemes have been widely used in interconnect design for energy-efficiency [21]. Bus-invert coding scheme [78] is one candidate due to its simplicity and efficiency. In this section, bus-invert coding scheme is applied to achieve further performance improvements.

Based on the experimental results in the last section, the improvement of interconnect performance for the DYN method depends on the distribution of transition patterns. The more the transitions in Groups 1 to 4, the larger the improvement in performance that can be achieved. In proposed DYN method, the transmitted data is the same as the source data. Therefore, the distribution of the transition pattern is determined by the intrinsic characteristics of the benchmarks.

In order to transform this distribution to increase the transitions that belong to the lower groups (1-4), the bus-invert coding scheme [78] is employed with proposed DYN method (See in Figure 3.11). Two crosstalk analyzers are used for determining whether the original data or the inverted data will result in smaller crosstalk delay. Accordingly, either the original or inverted data is selected for transmission. One more wire is required to indicate whether original or inverted data is on the bus, called *Indicator* in Figure 3.11. Bus-invert coding can apply to different numbers of wires. For example, for interconnect with 32 data wires, bus-invert scheme can be applied to 1 group of 32 wires, 2 groups of 16 wires, 4 groups of 8 wires, or 8 groups of 4 wires. Each group has one more wire

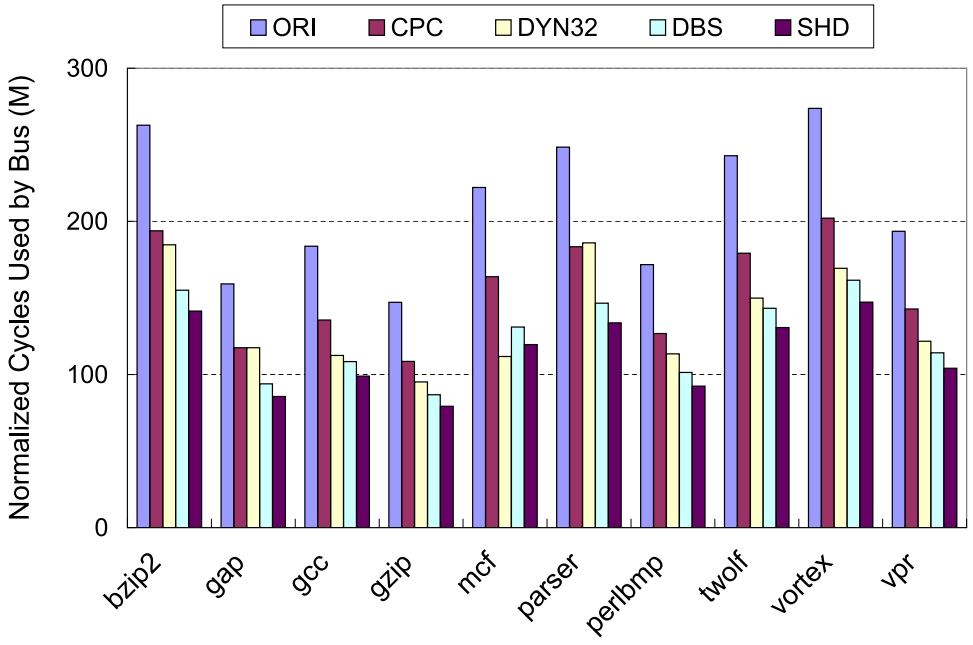

Fig. 3.10. Performance comparison for different schemes (DL1 data bus).

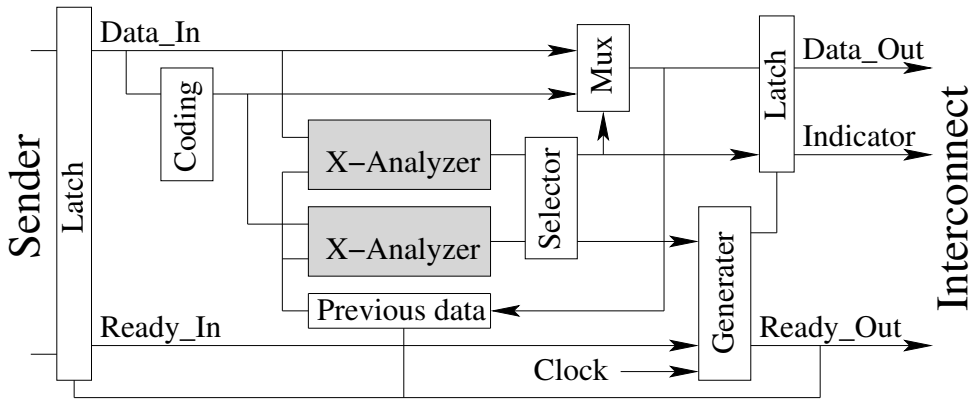

Fig. 3.11. DYN with bus-Invert scheme.

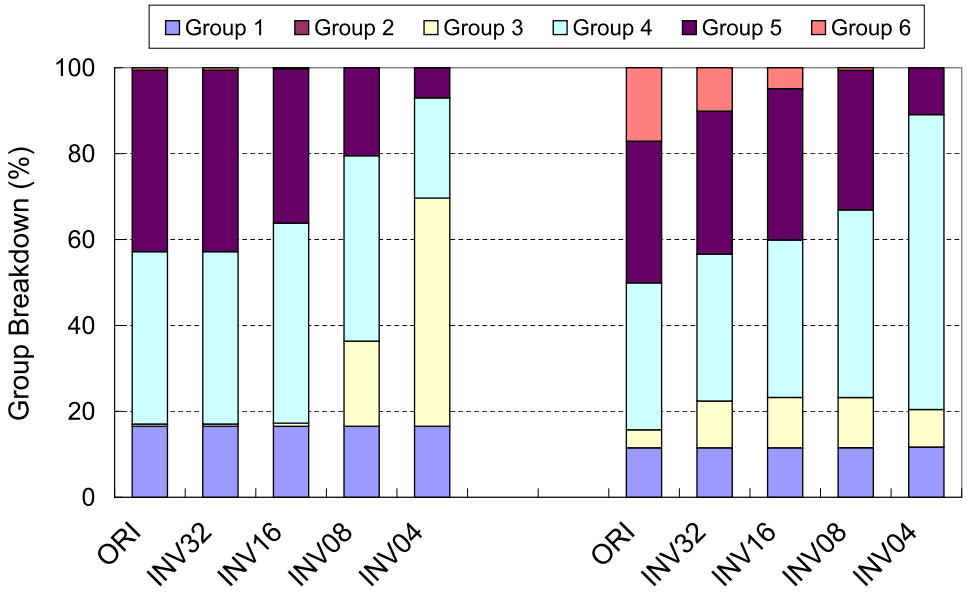

Fig. 3.12. Distribution of transition patterns of benchmark *bzip*2 for different INV schemes. (Left is for IL1 address bus and right is for DL1 data bus)

and groups are separated by shielding wires. The smaller the group of wires is, the more benefits we can get from bus-invert scheme.

The distribution of transition patterns of benchmark *bzip*2 are shown in Figure 3.12 with applying bus-invert coding scheme on different size of wire groups for both IL1 address bus (left) and DL1 data bus (right). All other benchmarks have the same results. The name of $INVnn$ is used to refer the scheme that combines both DYN and bus-invert scheme with group of n wires. Based on Figure 3.12, when moving from INV32 to INV04, the transition in Group 5 and 6 are reduced significantly. Transition in Group 5 change from 42.26% to 7.02% for IL1 address bus and from 32.97% to 10.93% for DL1 data bus. Transition in Group 6 change from both 0.57% and 17.12% to 0% for IL1 address bus and DL1 data bus.

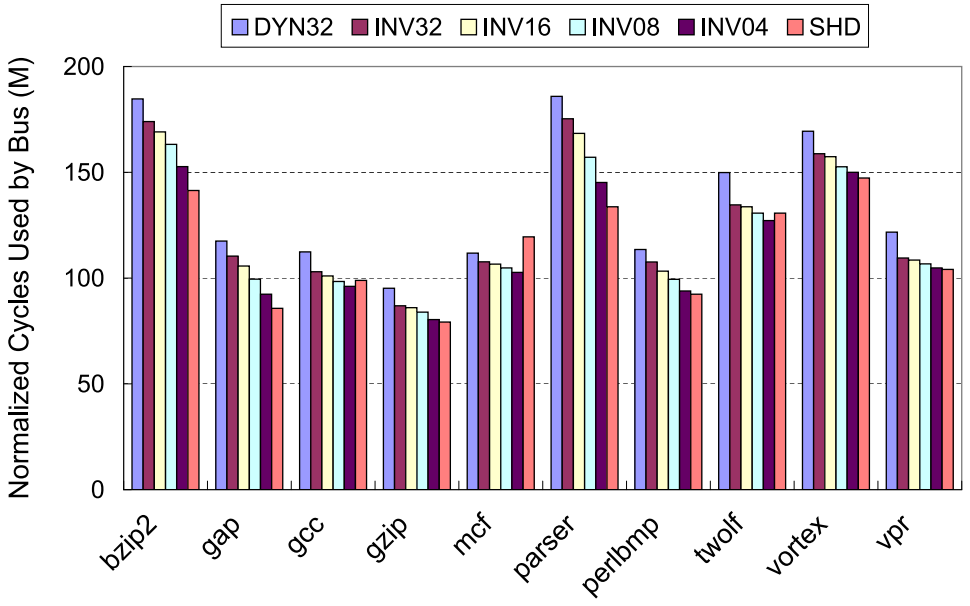

Fig. 3.13. Performance comparison for different INV schemes (DL1 data bus).

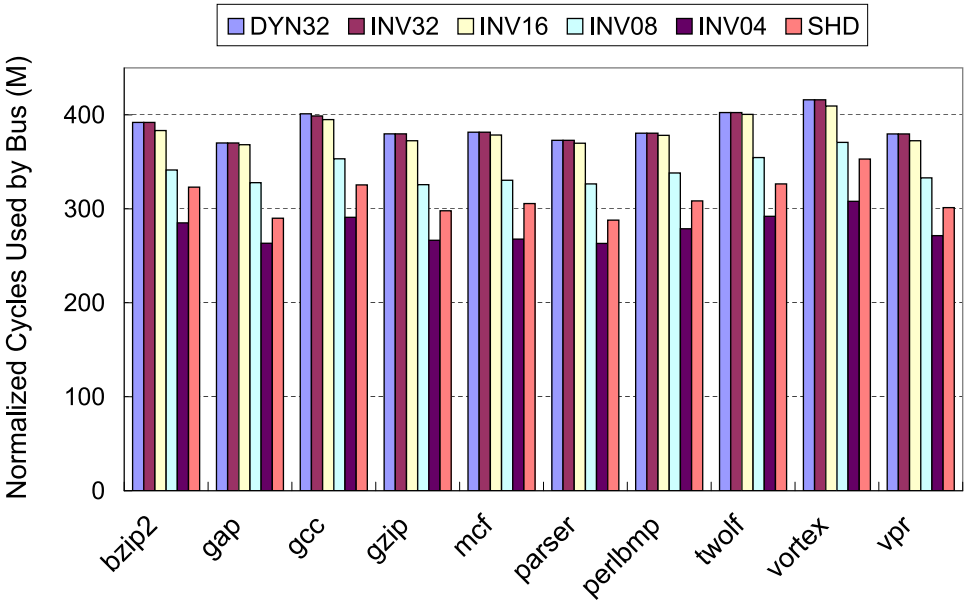

Fig. 3.14. Performance comparison for different INV schemes (IL1 address bus).

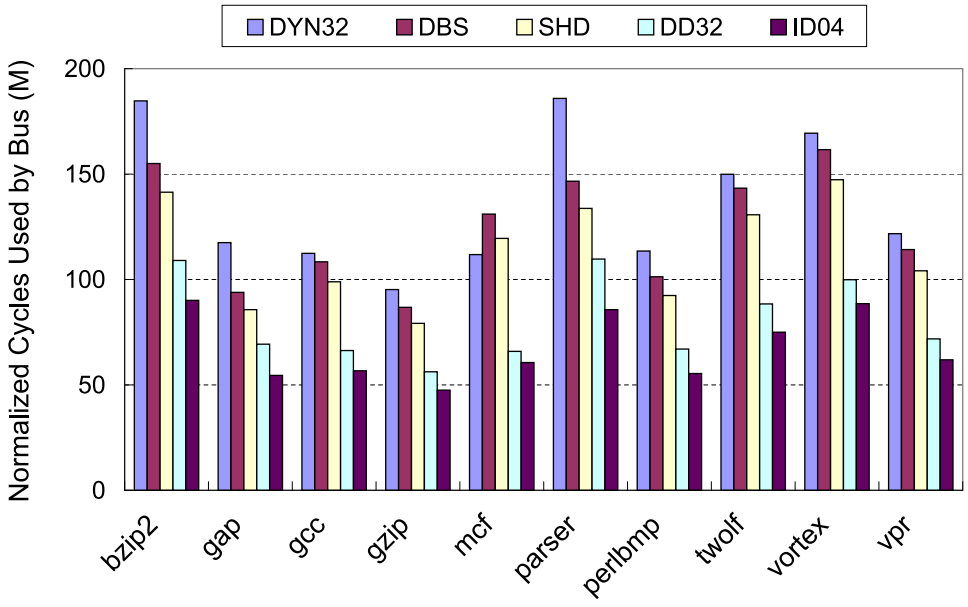

Fig. 3.15. Performance comparison for DD32 and ID04 (DL1 data bus).

The performance improvement achieved by combining DYN and bus-invert scheme with different size of wire groups are shown in Figure 3.13 for DL1 data bus and Figure 3.14 for IL1 address bus. For DL1 data bus, the average performance improvement (over original DYN) of INV32, INV16, INV08, and INV04 are 6.9%, 9.1%, 12.3%, and 16.0%, respectively. At the same time, SHD have 17.0% improvement over DYN. For IL1 address bus, the average performance improvement (over original DYN) of INV32, INV16, INV08, and INV04 are 0.1%, 1.2%, 12.3%, and 28.2%, respectively. At the same time, SHD have 19.7% improvement over DYN. Therefore, combining proposed DYN scheme with bus-invert coding scheme can achieve the same or better performance than the SHD scheme.

Table 3.6. Summary of normalized execution time of different schemes (average among benchmarks).

	IL1 address bus	DL1 data bus	Average	Meaning
ORI	100.0	100.0	100	original bus scheme
CPC	73.8	73.8	73.8	crosstalk prevention coding
DYN	67.0	64.8	65.9	variable cycle transmission
DBS	59.0	59.0	59.0	double spacing
SHD	53.8	53.8	53.8	shielding method
INV04	48.1	54.4	51.2	DYN + Bus-Invert
DD32	39.5	38.2	38.9	DYN + DBS
ID04	28.4	32.1	30.2	DYN + DBS + Bus-Invert

DYN exploits the differences in the temporal pattern of data transmitted in the buses to adjust the number of cycles based on estimated delay. However, the DYN approach does not affect the topology of the bus and maintains the same spacing as the original bus. In contrast, the DBS scheme only changes the topology and increases the spacing to reduce the coupling capacitance and does not exploit temporal correlations. Consequently, both these schemes are orthogonal to each other and can be combined together. First, the original DYN scheme is combined only with DBS technique, called DD32. Then all three schemes, DYN, DBS, and bus-invert scheme are combined together, and bus-invert coding is applied on group of 4 wires, which is called ID04. The performance results for DL1 data bus are shown in Figure 3.15 and results of original DYN, DBS, and SHD are used for reference. It is obvious that both DD32 and ID04 are better than SHD, which is the best in original five schemes. DD32 improves the performance with 41.0% over DYN and 29.0% over SHD. ID04 achieves the best performance improvement in this work, 50.5% over DYN and 40.3% over SHD. Finally, the normalized execution time of different schemes over ORI is summarized in Table 3.6.

The results are shown as the average value among benchmarks for IL1 address bus and DL1 data bus.

Chapter 4

Interaction between Energy Consumption and Reliability

4.1 Soft Error and Cache Energy Consumption

The focus of this section is on understanding the interactions between soft errors and the energy consumption behavior of the data caches. First, the impact of leakage energy optimizations on the soft errors is investigated. Consequently, the influence of soft errors is studied on a cache with no leakage energy optimizations and on caches employing the drowsy cache and cache decay technique for energy savings.

Next, how to protect the cache against soft errors in an energy efficient fashion is investigated. An adaptive scheme is proposed that protects dirty and clean data using different complexities of error protection mechanisms. This adaptive scheme is in contrast to current commercial caches that use a uniform error protection scheme for all data. This is also different from other schemes provided for energy-efficient data protection through duplication of data [89], or protecting only the frequently used cache lines [45]. The approach involving cache line duplication is orthogonal to the proposed approach and has been investigated using a cache using both ECC and parity. In contrast, this work focuses on providing different strength of protection for different cache blocks.

Table 4.1. Soft error rate (per cycle).

	Normal	Low Voltage	R/W
Single-bit Error	1E-7	1E-6	5E-7
Double-bit Error	1E-9	1E-8	5E-9
Multi-bit Error	1E-11	1E-10	5E-11

4.1.1 Soft Error Injection

The SERs used in experiments are shown in Table 4.1. The second column gives the SERs of the cache under the normal supply voltage, 1.0V in this work. The third column gives the SERs of the cache under the lower supply voltage used in drowsy cache mode, 0.3V in this work. The last column gives the SERs when a read/write operation on the target block, which reflects the fact that the cache blocks in read or write incur higher soft error rates.

Usually soft error rates are thought of as single bit upsets. But as the technology scales, the nodal cross section decreases, hence multiple bit upsets (MBU) are also probable. In [86], the magnitude difference between single and double error rates was found to differ from three orders to one order of magnitude based on the operating conditions such as supply voltage. To reflect this observation, two orders of magnitude difference is used between the error rates of single, double and multiple bit errors. Also as shown by Equation 1.5 in Chapter 1, error rates for low power mode was fixed as one order of magnitude higher than the corresponding high power mode, as the error rate is exponential dependent on the supply voltage [26].

During read and write operations, the error can manifest in either the SRAM cell or the peripheral circuits like sense amps. Also, when the wordline in SRAM is asserted,

charge sharing occurs and hence reduces the $Q_{critical}$ of the cell [71]. To account for these facts the error rate during read and write is increased by 5X from the corresponding steady state error rate. Since it is very time consuming to experimentally simulate for multiple hour executions, when current soft error rates will manifest in actual errors, the base probability of 1E-7 errors is used. While significantly higher than the error rate observed in current technology, it is still effective to mimic soft error problems that would occur in long running applications. And accurate relative numbers of the different soft error cases are used to provide a realistic evaluation.

In this work, a random soft error injection is implemented in the cache memories using the SimpleScalar [16] simulator. A random variable generated by the simulator along with the SERs provided in Table 4.1 is used to determine whether a soft error happens and the number of bits influenced by the error. In addition, three independent random variables are used to decide the set, way, and bit of block in the cache at which the external radiation hits. All these random variables are generated based on pseudo-random numbers using the linear congruential algorithm. These three random variables guarantee an even distribution of soft errors in a cache. Whether a soft error happens and how many bits error happens depends on not only the strength of external radiation, but also the state of the cache block. Cache blocks in different state have different susceptibility to external radiations.

4.1.2 Influence of Leakage Optimizations on Soft Error Rates

Drowsy cache is based on controlling the leakage by using dynamic voltage scaling (DVS) while using the standard 6T SRAM cell structure [28]. This method makes use

of the fact that to retain a value in the SRAM cell, the source voltage of the cell can be just about 1.5 times of V_t. Thus, for a 70-nanometer technology based SRAM cell that normally operates at 1.0V, the voltage can be reduced to up to 0.3V when accesses are not required while still retaining the data. Substantial leakage energy saving can be gained when placing the cache line in this low voltage drowsy state [28]. However, one cycle penalty is incurred when accessing a drowsy cache line, as the supply rails have to be restored to 1.0V before a read or write operation. For voltages less than 0.3V it was noted that the cache line lost its values. In the drowsy cache scheme, all the cache lines are periodically switched to a lower voltage assuming a periodic change in the working set of the application.

When in the drowsy state, the cache cells due to the reduced supply voltage are more susceptible to soft errors compared to the normal state. This observation can be deduced from Equation 1.5. The parameters influencing SER, except $Q_{critical}$, shown in the Equation 1.5 are the same since the underlying circuit structure remains the same. $Q_{critical}$ reduces superlinearly with the supply voltage (since capacitance is also a function of the supply voltage). Thus, the use of DVS provides an interesting opportunity for trade off between leakage reduction and soft error immunity.

In [41], Kaxiras et al. present a leakage energy reduction technique, called Cache Decay, for cache memories. Cache Decay exploits the temporal behavior of cache blocks to reduce leakage consumption. This technique is based on the idea that a cache block that is not used for a sufficiently long period of time can be considered dead. More specifically, with each cache block, they associate a small 4-state FSM (finite state machine). The FSM steps through these states as long as the cache block is not accessed.

Table 4.2. Benchmarks and execution cycles.

	Execution Cycle		Execution Cycle
bzip	793,768,890	parser	992,353,260
gap	731,340,328	perlbmk	1,455,161,111
gcc	1,521,913,064	twolf	1,661,660,355
gzip	769,888,583	vortex	1,758,572,035
mcf	2,147,483,647	vpr	945,542,126

When the last state is reached, the cache block is turned off completely after committing any changes back to higher levels of memory hierarchy. This can be implemented using the same circuit fabric as in drowsy cache but by changing the sleep mode voltage to 0V instead of 0.3V. Since the data is completely destroyed, soft errors in the idle state is not a concern for this leakage control mode. Furthermore, since cache lines are evicted from the cache close to their anticipated dead times, the time for which they are exposed to soft errors in the normal mode is also smaller as compared to the original cache. In an original cache, the entries will be evicted only when replaced by another cache line when employing the commonly used write-back approach.

The soft error injection, drowsy cache, and cache decay are implemented in the simulator SimpleScalar 3.0 [16]. The benchmarks are from SPEC CINT2000 [5]. Each benchmark is first fast forwarded 300 million instructions and then simulated 1 billion instructions. Table 4.2 gives the total number of cycles taken by each benchmarks. In drowsy cache, the entire data cache was put into a low voltage mode for every 2000 cycles. In cache decay scheme, the decay interval of L1 data cache is set as 10K. The default configuration of the simulator with a 16KB, 32 byte, 1-way L1 instruction cache,

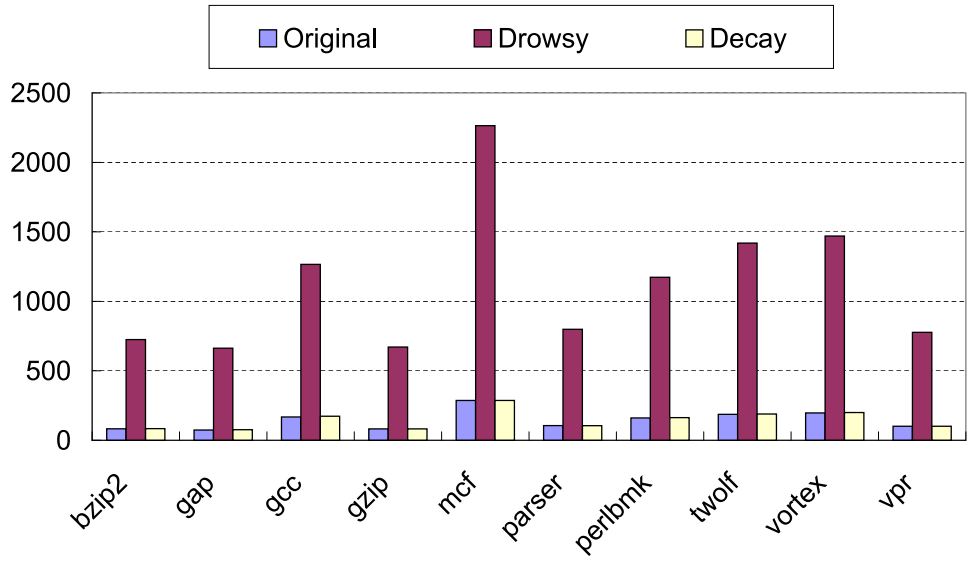

Fig. 4.1. Number of soft errors when using the original cache with no leakage optimizations, drowsy caches and cache decay.

a 16KB, 32 byte, 4-way L1 data cache and a unified 256KB, 64 byte, 4-way L2 cache was used in the experiments.

Figure 4.1 shows the total number of soft errors injected in the L1 data cache in original cache, drowsy cache, and decay cache for each benchmark. Since the original cache and decay cache use the same memory cells, they have the same SER and, therefore, incur the same number of soft errors. But in the drowsy cache, a large portion of cache blocks that operate at a lower voltage when in drowsy mode are more susceptible to soft errors (See table 1). Therefore, the total number of soft errors induced in the drowsy cache is significantly more than that in the normal cache.

When a soft error happens, it happens either on a valid cache block or an invalid cache block. Later, the valid cache block with soft error can be read into datapath, overwritten by new data, replaced by new cache block if it is clean, or written back to

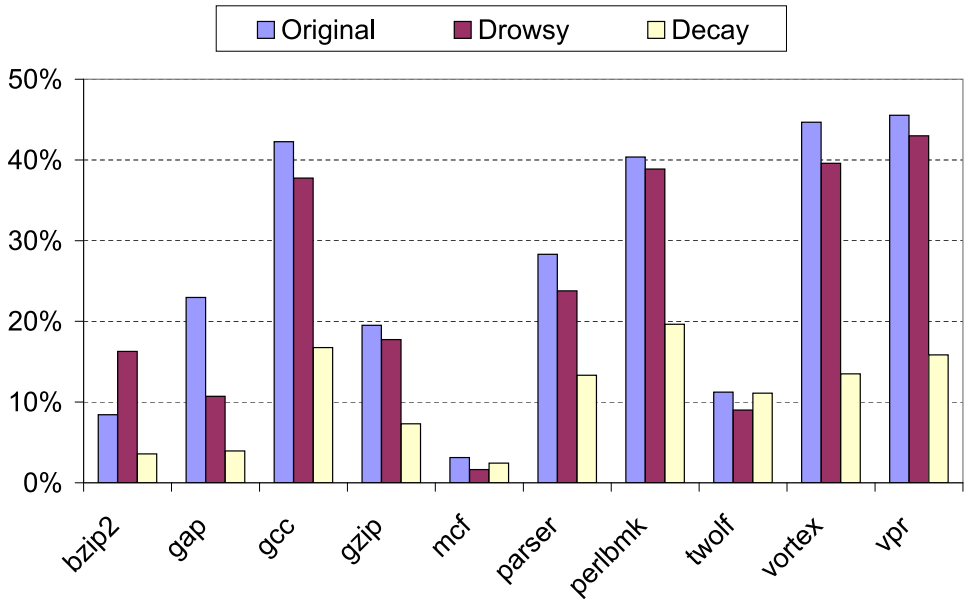

Fig. 4.2. Percentage of soft errors read into datapath.

L2 cache if it is dirty. It is obvious that not every injected soft error will impact the correctness of the operation performed by the system. Therefore, the soft errors being read into datapath and written back to the L2 cache is denoted the effective soft errors as they propagate the error beyond the L1 cache.

Figure 4.2 shows the percentage of injected soft errors being read into datapath. It can be observed that the percentage of errors that propagate to the datapath is significantly less when employing the decay cache. This happens because the decay cache increases the number of invalid cache blocks by applying cache block shut down. All errors that occur in invalid blocks do not propagate to the datapath. In contrast, the original and drowsy modes do not perform any additional tasks to increase the number of invalid cache blocks. Also, note that the absolute number of errors that propagate to

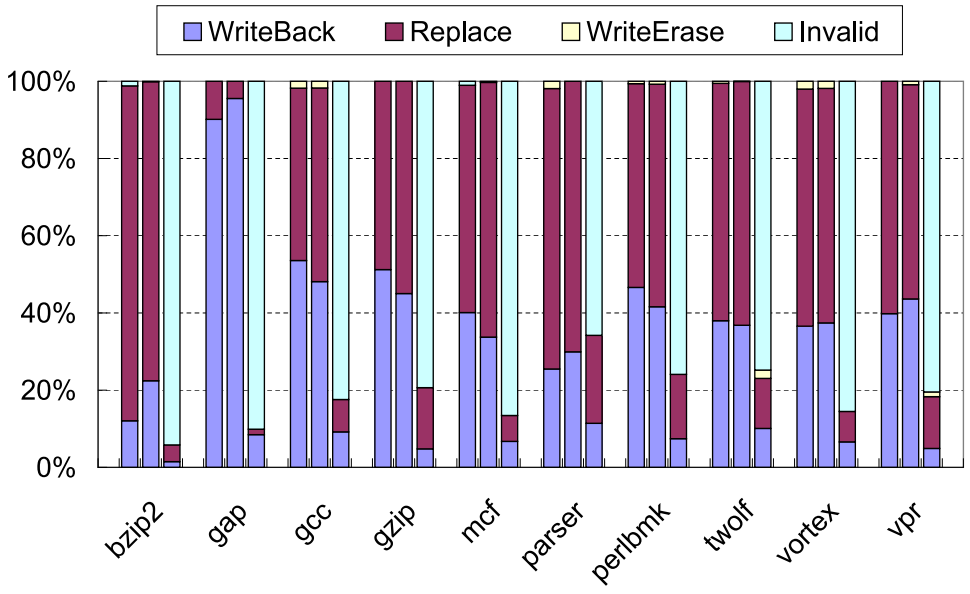

Fig. 4.3. Distribution of how L1 cache blocks with soft errors propagate in the memory hierarchy (from left to right columns are original, drowsy, and decay cache).

the datapath for the drowsy cache is much higher (on the average 6.7 times more) than that of the original cache.

Figure 4.3 shows how the error in the L1 cache block affected by soft errors will propagate in the memory hierarchy. The problematic case occurs when the dirty data is written back to the L2 cache. It is observed that this portion is 43.3%, 43.4% and 7.1% on the average for the original, drowsy and decay caches, respectively. Consequently, decay cache not only reduces the leakage energy consumption for the cache block which is not accessed anymore, but also reduces the chance of those blocks being exposed to external radiation. Specifically, when doing shutdown, if there is a dirty block it is written back early, reducing the amount of time it will be exposed to a soft error.

Table 4.3. Drowsy cache with bit interleaving.

	without Interleaving			**with Interleaving**		
	1-bit error	2-bit error	Multi-bit error	1-bit error	2-bit error	Multi-bit error
bzip2	716	9	0	734	0	0
gap	656	7	0	670	0	0
gcc	1246	20	0	1286	0	0
gzip	663	8	0	679	0	0
mcf	2232	32	0	2296	0	0
parser	789	10	0	809	0	0
perlbmk	1159	14	0	1187	0	0
twolf	1400	19	0	1438	0	0
vortex	1450	20	0	1490	0	0
vpr	767	10	0	787	0	0

It is also observed that double-bit errors also occur, especially in the case of drowsy cache lines that operate at a lower voltage. These double bit errors are those caused by particle strikes with an impact area spanning more than a single memory cell and not due to two independent single events upsets that is rare. Handling multi-bit errors will necessitate more powerful error correction schemes than single bit correction or parity-based recovery for clean data.

A more effective optimization for such cases is by employing bit interleaving [57] that interleaves bits from different cache block in a row. This technology distributes the impact of a multi-bit soft error on a single cache block into multiple single bit soft errors on multiple cache blocks. The result of the number of soft errors that occur in the drowsy cache with and without bit interleaving scheme are shown in the Table 4.3. Based on the results in Table 4.3, most double-bit errors are converted into single bit errors. Therefore, the double bit errors and multi-bit errors are reduced significantly

with the increase in single bit errors. This alleviates the burden of more powerful error detection and error correction circuits.

4.1.3 Energy-Efficient Soft Error Protection

Error detection or correction codes are a way of introducing redundant information in the form a codeword. Error protection codes are widely used to improve the memory reliability. Most simple form of coding involves adding a single bit to store the parity (odd or even) of each data word. But this can only detect the errors and not correct them. Another commonly used scheme is SEC-DED. (38, 32) Hamming code and odd-weight code belong to this class of code. The most important feature of this code is its fast encoding and error detection in the decoding phase. This code can correct the single bit errors and detect double bit errors with extra bits overhead. Therefore, the more powerful the error protection coding scheme is, the higher reliability it guarantees and the more energy consumption and area overhead it incurs.

Compared to clean blocks, dirty blocks in L1 data cache have no corresponding duplicate copies in the L2 cache (or memory, if no L2 cache is present) when using the commonly used write-back approach. This means that data in the dirty blocks may be damaged permanently, in case one of the bits flip due to soft errors. In contrast, data with soft errors in the clean blocks can be recovered from the duplicate copy in the L2 cache. Consequently, dirty blocks need to be protected more strongly than clean data.

Error protection schemes are widely used in the cache memories, such as parity and ECC coding. However, they employ the same protection for the cache lines. This can be wasteful as clean and dirty blocks can operate with different strength of error

Table 4.4. Power consumption and delay of different coding schemes.

	Power (mW)		**Delay (ns)**	
	Coding	Decoding	Coding	Decoding
Parity	6.9232	7.2239	1.41	1.41
SEC	14.4871	26.2962	1.45	2.66

protection schemes while providing the desired protection. Proposed adaptive protection scheme treats these cache lines differently. In order to eliminate 1-bit soft error, dirty blocks need Single Error Correction (SEC) coding at least. But the clean blocks may only use Single Error Detection (SED) coding and then get the correct data from the L2 cache. Since error cases are rare, the performance and energy overhead of additional L2 accesses are insignificant.

Since the energy consumption of SEC and SED are quite different, the proposed scheme provides significant savings over a scheme that uses single error correction approach for all cache lines. Table 4.4 gives the power value and delay of coding and decoding for SED (implemented by parity) and SEC (implemented by Hamming (38,32) code) custom implemented using 250 nm libraries. In order to support the adaptive error protection, the dirty bit is used to determine whether SED or SEC is used. An additional 6 bits are associated with each cache line to support the more powerful SEC. In the case where only SED is required (dirty bit not set), the additional 5-bits are supply gated to reduce leakage and only the single bit is used for parity.

Next, an optimization is explored to decrease the amount of time that single error correction needs to be employed. This can be achieved by reducing the duration for which cache lines remain in the dirty state. Traditional cache designs employ the

write-back policy, which keep dirty blocks in L1 data cache as long as possible and write the data in this dirty block back to L2 cache until this block is replaced by another block. Write-back policy significantly reduces the number of writes to L2 cache. At the same time, it makes the dirty blocks vulnerable to soft errors for a long time. Intel Pentium® M processor uses write-back policy in L1 data cache.

Another common write policy is write-through, which writes the data back to L2 cache whenever it is changed in L1 cache. Intel Itanium® 2 processor uses write-through policy in L1 data cache. This method keeps the blocks in L1 cache in clean mode, but it significantly increases the number of L2 cache accesses. Therefore, a coalescing write-buffer [75] is always employed with write-through cache to merge the writebacks to L2 cache. But the total number of L2 cache accesses of write-through cache is still significantly larger than that of write-back cache.

A new write policy, early-write-back, is proposed which writes the data back to L2 cache after a fixed time gap from last write operation. In experiments, the 10K cycles is used as the time period. This means when 10K cycles pass after the last write operation, the dirty block is written back to L2 cache, if not evicted earlier. This method keeps the blocks in clean mode longer and does not increase the number of write to L2 significantly. The implementation of the time period monitoring is achieved in a fashion similar to that used in the decay cache.

Three different error protection schemes is implemented for evaluation. Scheme 1 uses a L1 data cache with SEC protection for all cache blocks and uses a write-back policy. Scheme 2 uses a write-through L1 data cache with a coalescing write-buffer. The coalescing write-buffer contains four entries and each entry is the same size as the

cache line. SED protection is used for all cache lines and SEC protection is used for coalescing write-buffer. The last one is proposed approach that combines the adaptive error protection scheme and the early-write-back policy.

Figure 4.4 shows the energy consumption of these three different schemes that provide equal protection to soft errors. All these schemes have equivalent error behavior but exhibit different energy behavior for providing the protection. They differ in the energy required for the error encoding and decoding with cache line accesses, any additional L2 accesses incurred because of the differences in the write policies, and error encoding and decoding with coalescing write-buffer accesses in Scheme 2. Energy consumption for accesses to coalescing write-buffer is very small and is omitted in simulation.

It is observed that Scheme 2 focuses on minimizing the energy consumption for error coding protection since there are no dirty blocks. However, it incurs a significant number of additional L2 accesses. In seven cases, the total energy consumption of Scheme 2 is larger than those of Scheme 1 and 3. The average energy overhead of Scheme 2 is 14% over Scheme 1. In contrast, Scheme 1 minimizes the energy required for additional L2 accesses while requiring significant energy for using SEC for all cache blocks. Proposed approach balances between previous two schemes and achieves the minimum overall total energy consumption. The average of energy reduction of Scheme 3 is 11% as compared to Scheme 1.

Figure 4.5 shows the time distribution of blocks in clean mode and dirty mode. It is obvious that all blocks are in clean mode under write-through policy. It also can be observed that dirty blocks are around 56% under write-back policy for most benchmarks. In comparison, the early-write-back significantly increases blocks in the clean mode,

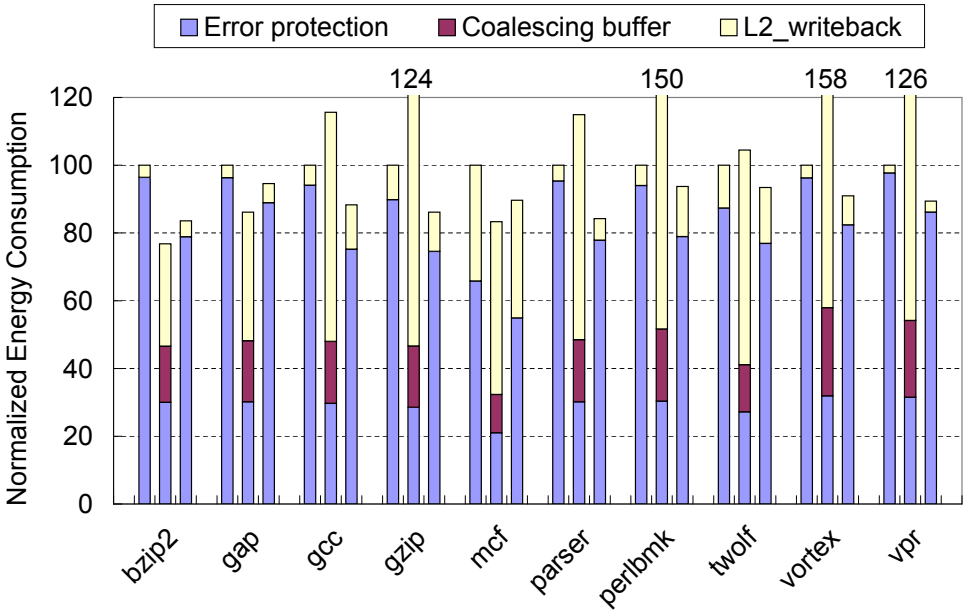

Fig. 4.4. Energy consumption of different schemes (from left to right columns are Scheme 1, 2, and 3).

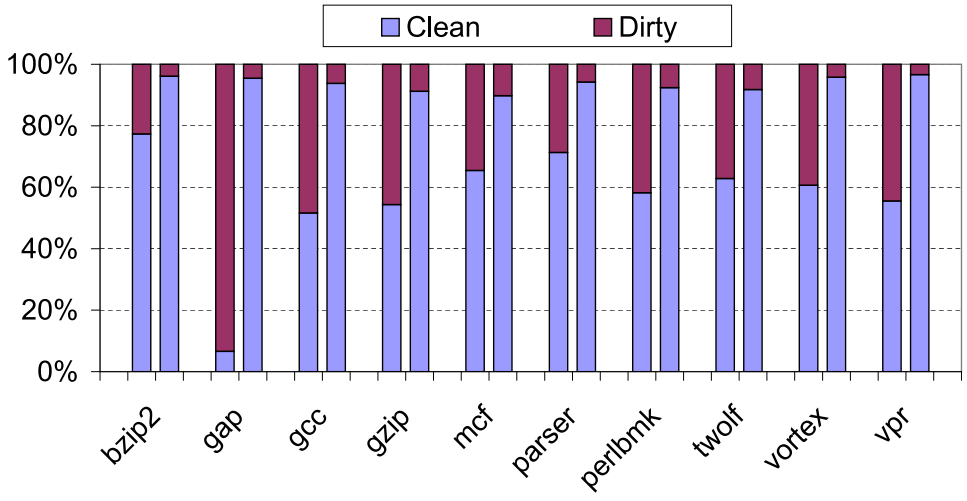

Fig. 4.5. Distribution of clean and dirty blocks for Scheme 1 (left) and Scheme 3(right).

around 94%. Increasing the overall percentage of clean blocks results in more read operations accessing clean blocks. Consequently, many of the SEC operations translate to the less power consuming SED operations when employing early-write-back with the adaptive protection scheme. The results also show that proposed technique incurs little additional performance penalty (less than 0.1%) as compared to the write-back scheme used in Scheme 1.

4.2 Adaptive Error Protection for On-Chip Interconnects

To guarantee reliability, many error correction codes and error detection codes such as parity, Hamming code, and Berger code are increasingly used to protect the data stored in memories and data transmitted in the buses. Different coding methods have different capabilities to detect or correct errors induced by different noise sources. While the coding scheme adopted can be designed for the worst-case noise scenario, such an approach will be inefficient in terms of both energy and performance.

It must be observed that the induced noise fluctuates due to various environmental factors (such as altitude for soft error rates) and operational conditions (e.g., supply voltage variations due to changes in switching activity). Hence, it is necessary to design a system with self-embedded intelligence to provide the minimum amount of protection to meet the desired reliability levels. This section focuses on ensuring the reliability of communications in on-chip multiprocessors and concentrates on the data buses connecting the private L1 caches and the shared L2 cache. Specifically, an adaptive error protection scheme is designed and evaluated that varies the coding technique based on the dynamic variations in error rates. This scheme helps to reduce the energy

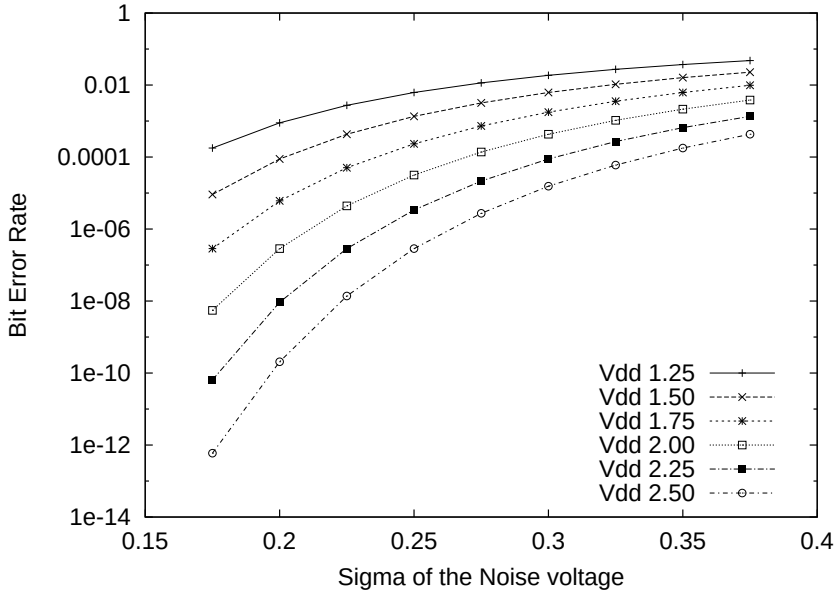

Fig. 4.6. BER with different V_{dd} and σ.

consumption by employing a less complex protection mechanism whenever appropriate. The objective is to maintain the same degree of system reliability provided by the most powerful protection mechanism without always incurring its high energy cost.

4.2.1 Noise and Error Protection Schemes

Noise model introduced in Equation 1.3 of Chapter 1 is used in this work, which models the different noise sources impacting the bus line as a single Gaussian noise source. Note that the probability of error is sensitive to both the supply voltage (V_{dd}) and the variance. Specifically, a lower voltage or a larger variance increases the possibility of an error as illustrated in Figure 4.6, which shows the BER with different V_{dd} and σ.

In order to model the impact of temporal variations in the noise, σ is varied over the execution of the application. This variation causes a change in distribution of the

magnitude of noise voltages generated. The objective in changing σ is to model physical effects such as variations of flux distribution of alpha particles with altitude/latitude or changes in crosstalk patterns based on data activity in adjacent buses.

Based on this one bit error model, the error in each bit line of the target 32-bit bus is captured as independent and identically distributed random variables (all bit lines use the same σ at any given time). Note that multiple errors occur when two or more individual bit lines generate error at the same cycle.

In this work, error-detection schemes is used with re-transmission on identified errors as the means for error protection. The choice was motivated by the observation in [15] that error-detection schemes with retransmission are less costly in terms of energy consumption than error-correction schemes at low error rates typical in on-chip environments. Three error detection schemes of different strengths is considered in the evaluation.

- Parity (PAR) uses one extra bit to detect all odd number of bit errors and is a well known representative of single-bit error detection schemes.

- Double Error Detection (DED) uses a (38,32) Hamming code that can detect all single and double bit errors and a subset of multiple bit errors. This scheme uses 6 additional bit lines to carry protection information for the 32-bit bus.

- Triple Error Detection (TED) employs an extra parity bit in addition to the (38,32) Hamming code in order to detect all single, double, and triple bit errors and a subset of higher bit errors [83].

Table 4.5. Power consumption and normalized area (with respect to Encoder for PAR) for different error coding schemes.

	Energy (mW)				**Area**	
	Encoder	**Decoder**	**Total**		**Encoder**	**Decoder**
PAR	4.74	4.92	9.66	100%	1.00	1.03
DED	9.96	11.91	21.88	226%	1.91	2.32
TED	12.58	17.41	29.99	310%	2.43	3.45

In order to implement these codes, an encoder is required at the sender and a decoder is required at the receiver of the bus. Consequently, the encoders and decoders are modeled for PAR, DED, and TED in VHDL and synthesized using a target 0.25um library to obtain their power consumption and layout area characteristics shown in Table 4.5. While the TED coding scheme provides the best error detection ability, its encoding plus decoding consumes 3.1 (1.4) times the energy required by PAR (DED).

Also, there is additional energy consumed in the extra bus lines used in TED and DED as mentioned earlier. The energy consumed in the bus is evaluated using the following equation:

$$E_{bus} = s * C_L * V_{dd}^2 * N_{lines} * N_{trans}, \tag{4.1}$$

where s is the average switching activity, C_L is 5 pF in evaluation to model an on-chip bus, V_{dd} is 2.5V, N_{lines} are 33, 38, 39 for PAR, DED, and TED, respectively. Here, N_{trans} is the number of transactions (including the retransmissions when errors are detected).

In order to illustrate the differences in the error protection offered by the three schemes considered, Figure 4.7 shows the resulting word error rate for different bit error rates, and Figure 4.8 shows the word error rate as a function of varying σ. It is clear that

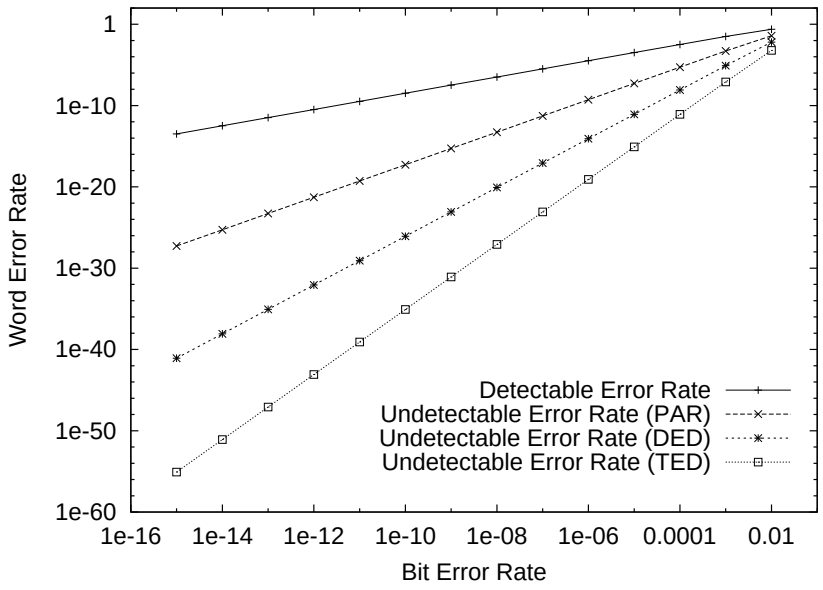

Fig. 4.7. Error rate of different EDC methods.

the TED is superior in offering much lower undetectable error rates as compared to DED and PAR. It must be observed that undetectable errors are of serious concern as they can lead to faulty outputs or system crashes. In this figure, the detectable error rates of the different schemes are difficult to distinguish in the scale used and are represented by a single curve (named detectable error rate).

This graph also indicates that minor changes in detectable errors magnify to huge variations in undetected errors (The plot for victim line will be explained later). Note that the target reliability metric of undetectable word error rate cannot be observed directly. In order to impose the required bounds on this unmeasurable metric, its relationship is utilized with the detectable error rates in the implementation. From the above discussion, it is clear that there are inherent tradeoffs between error protection

capability and energy consumption. The adaptive strategy to be presented in the next section exploits this tradeoff.

4.2.2 Adaptive Error Protection

The objective is to adapt the strength of error detection scheme dynamically based on the noise behavior observed. Using only the simplest error detection coding method (PAR), while being most energy-efficient, will lead to high number of undetectable errors. At the other extreme, adopting the worst case approach and employing the most powerful scheme (TED) catches most of the errors but incurs unnecessary energy consumption most of the time due to its conservative nature. Therefore, if the change in noise behavior can be monitored and error detection scheme is switched to the least powerful one that can maintain the undetected error rates below specified levels, the energy consumption can be minimized while maintaining required protection levels.

The main area overhead of adaptive method is the size of the encoder and decoder due to providing the functions of PAR, DED, and TED. Synthesis result shows that the total area of encoder and decoder in proposed method is 29% more than that in TED method.

There are two important aspects of the proposed approach: (a) detecting the variation in noise behavior and (b) identifying the protection scheme to employ for the observed noise behavior. An indicator of the variation in the noise behavior is the detected error rates observed at the decoders of the bus. When detectable error rate is small, it requires a long period before the number of detectable errors change and this makes it difficult to use the counted errors to effect changes in the protection scheme

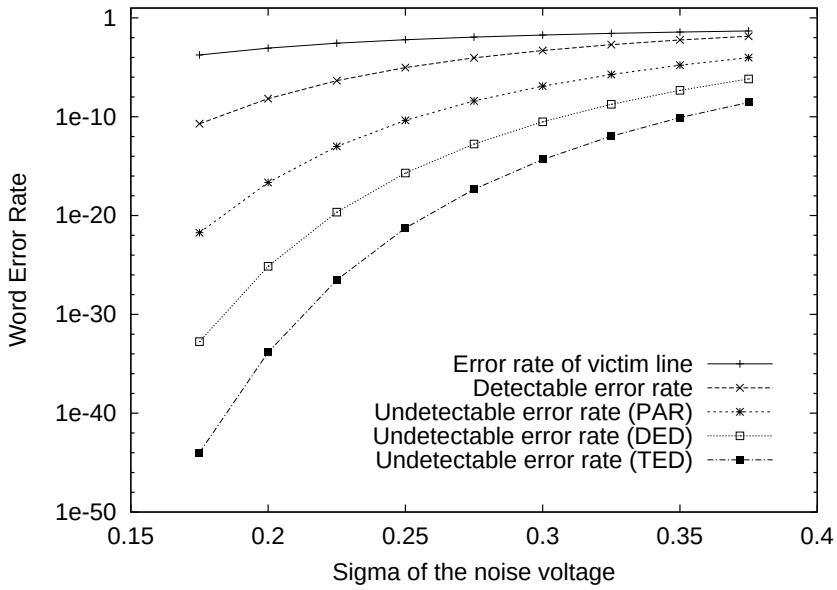

Fig. 4.8. Variation in word error rate as a function of σ.

promptly. For example, when σ is 0.3, the detectable error rates of PAR, DED, and TED are 5.09E-4, 5.87E-4, and 6.02E-4, respectively, and they result in only five or six errors in 10,000 transmissions.

Coupled with the observation that even small variations in detectable error rates indicate huge variations in undetectable error rates, it becomes important to devise an alternate monitoring scheme. For this purpose a *victim bus line* is utilized that amplifies the number of detectable errors in order to detect changes in the noise behavior quickly. This victim line uses half the voltage swing as the normal bus lines and is more susceptible to variations in noise. Consequently, it can be observed from Figures 4.8 that the detectable error rate of the victim line is much higher than those of PAR, DED, and TED schemes. In the implementation, the victim line transmits a repeated sequence of 0-1 with each data transmission and one per 10 cycles when the bus is in idle

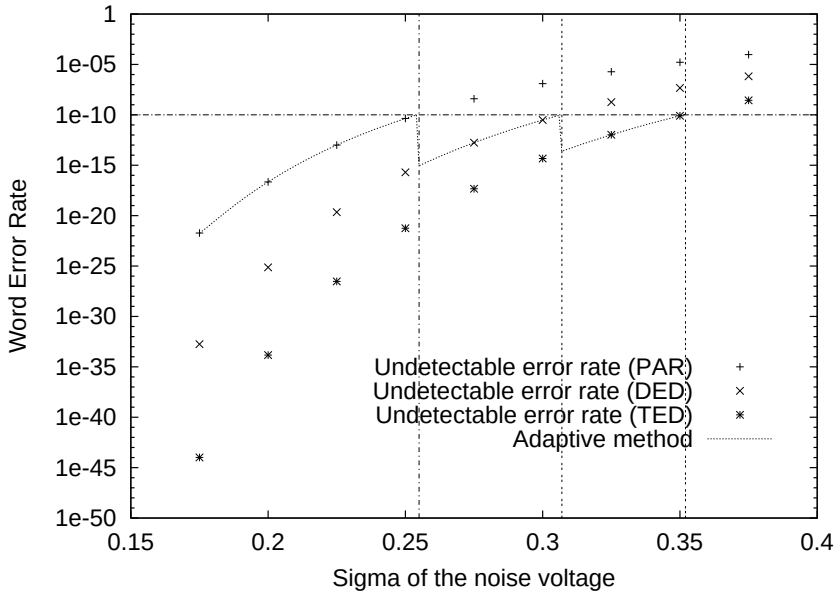

Fig. 4.9. Word error rate in adaptive method as a function of σ.

state. Whenever, the decoder of this line receives the same bit value in two successive transmissions it indicates a detected error.

The main goal of adaptive approach is to keep the undetectable word error rate below a preset value (threshold) when the noise variable σ changes. For purpose of evaluation, the target undetectable word error rate is set to 1E-10. When more than one coding scheme can achieve the target undetectable word error rate, the scheme that has the lowest energy consumption is selected.

Figure 4.9 shows the behavior of adaptive strategy as σ changes. Specifically, when σ increases, the simple PAR method is used at first. When the undetectable word error rate of PAR exceeds the predefined value 1E-10, the proposed method switches to DED, and later to TED when DED can no longer provide the required reliability. As a result, one can identify three thresholds, $\sigma = 0.255$, 0.307, and 0.352 for switching

Table 4.6. Influence of sampling window size on the number of errors detected.

window size	$\sigma = 0.255$		$\sigma = 0.307$		$\sigma = 0.352$	
	Min	Max	Min	Max	Min	Max
1,000	0	21	4	42	15	68
10,000	41	106	153	266	312	461
100,000	620	822	1939	2241	3595	4011
1,000,000	6858	7360	20499	21292	37441	38303

between the different schemes from this figure. It must be noted that the strength of coding can also be reduced using the same thresholds. Further, when σ is over 0.352, the implementation cannot guarantee the required reliability level. In order to identify these switching points of σ, two counters are utilized, a saturating counter to track a sampling window and another counter to maintain the total number of detected errors on the victim line during this window. At the end of every sampling window, the error counter is reset.

In order to choose an appropriate sampling window size, 10 million transmissions are simulated with different σ values associated with the three thresholds mentioned above and experiment with different window sizes to count the number of errors on the victim line. Table 4.6 shows the minimum and maximum values of errors detected in the simulation across all the sample window sizes considered. If a small window size is selected, the system can react faster to changes in noise. However, a very small window size such as 1,000 transmissions cannot distinguish between different noise levels due to the overlap between the maximum and minimum errors that can be observed. In contrast, a large value for sampling window size can clearly distinguish between the different noise levels but will react too slowly to the changes in noise behavior. Based

on the results in Table 4.6, a sampling window size of 10,000 transmissions is selected for experiments.

Having selected the sampling window size, the thresholds is determined with which to compare the error counter values for switching between different coding schemes. Using equation 1.3 and the three σ values corresponding to the thresholds, the BER is calculated. Then utilizing the BER and assuming the maximum number of possible transmissions that can occur within given sample window size, the number of errors is computed corresponding to the three cases as 71, 208 and 379. In order to avoid thrashing between the choices of the schemes when noise behavior straddles across the boundaries, the following rules is set for switching: switch from PAR to DED when σ exceeds 0.255 (counter value 71) and is below 0.307 (counter value 208); switch from PAR/DED to TED when σ exceeds 0.307 (counter value 208); switch from TED/DED to PAR when σ is below 0.235 (counter value 39); from TED to DED when it is below 0.287 (counter value 147) and higher than 0.235 (counter value 39).

It is essential for both the encoders and decoders to be notified of the change required in the coding scheme. In the implementation, this task is performed by transmitting a special code to all encoders and decoders using the data bus. This transmission occurs quite infrequently and is observed to cause a negligible impact on performance and energy.

4.2.3 Experimental Evaluation

The target system is an on-chip multiprocessor where each processor has private L1 data and instruction caches and the processors share an on-chip L2 cache. To model

Table 4.7. The number of transactions and execution cycles for the Splash2 benchmarks.

	transactions	cycles
barnes	10098956	490339742
ocean1	35087162	1719408062
ocean2	22634451	2039085568
radix	7776529	722890419
raytrace	22299175	1108642808
water1	5468576	442982664
water2	9083072	361669709

this system, MP_simplesim [58], a multiprocessor version of the SimpleScalar simulator, is used. The default configuration contains four processors and has L1 (8K) and L2 (256K) access latencies of 1 cycle and 10 cycles, respectively. The main memory latency is 100 cycles. The focus of the adaptive strategy is the data bus between the L1 caches and the L2 cache, which is the longest on-chip interconnect in the design other than the clocking network. It should be noted that the proposed approach can also be applied to other buses and architectures.

The proposed strategy is evaluated using the benchmarks from Splash2, a widely used parallel benchmark suite [84]. Each benchmark is simulated for 500 million instructions and the corresponding number of transactions and execution cycles are given in Table 4.7. These values correspond to the case when no error protection is implemented.

In order to simulate the noise activity, two different types of noise profiles are used. The first profile mimics a scenario where there is a repeated pattern of a slow increase in the severity of the noise problem followed by a phase of decrease. To model this, σ is changed from 0.150 to 0.389 continuously in the first part and from 0.389 to 0.150 in the second part of a period of 100 million cycles (this profile repeats itself

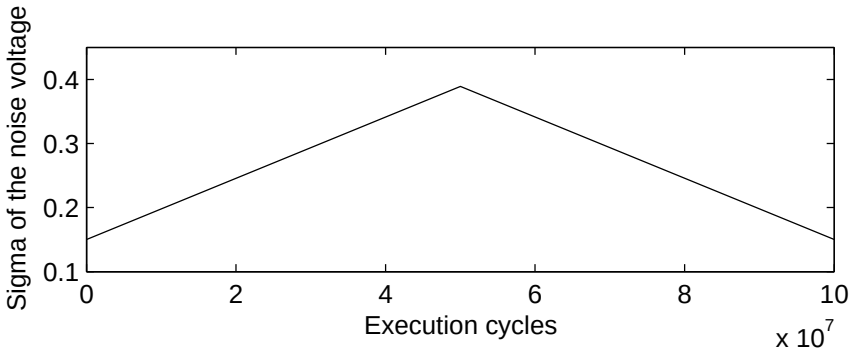

Fig. 4.10. Noise profile 1.

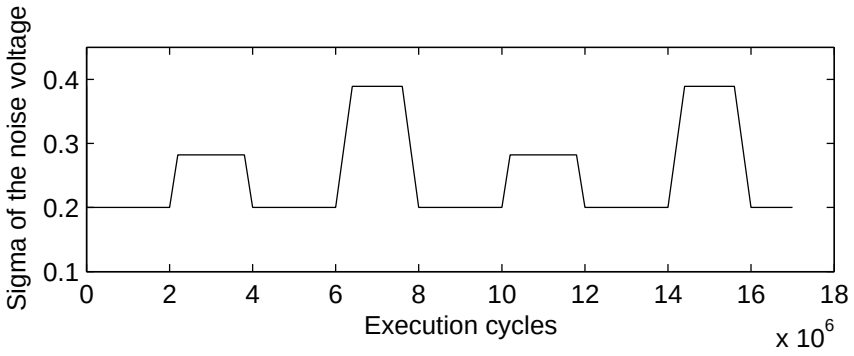

Fig. 4.11. Noise profile 2.

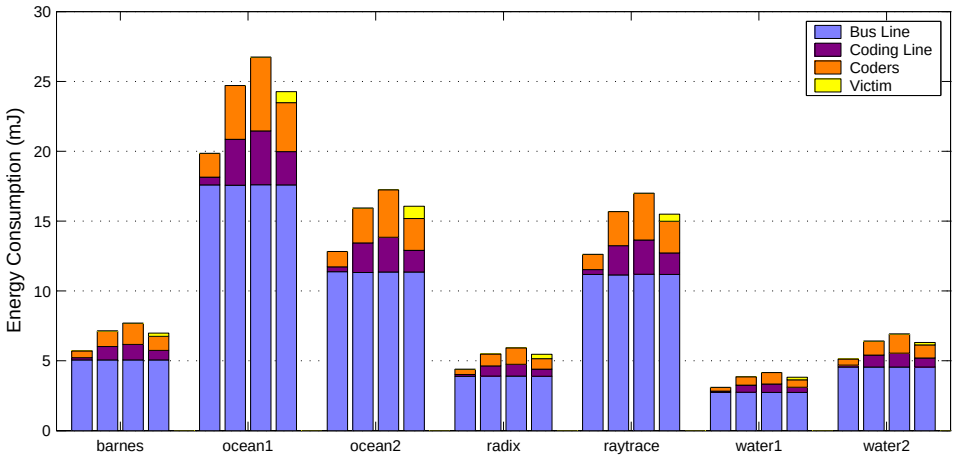

Fig. 4.12. Energy behavior with noise profile 1. The four bars for each benchmark from left to right correspond to using PAR, DED, TED and adaptive approach.

throughout execution) as shown in Figure 4.10. The second profile corresponds to the case where there are more frequent and random changes in the noise behavior and is modeled as illustrated in Figure 4.11. The noise in this profile repeats every 8,000,000 cycles. Here, σ is 0.2 by default and varies to 0.282 from cycle 2,000,000 to 4,000,000 and to 0.389 from cycle 6,000,000 to 8,000,000.

4.2.4 Results

In this subsection, the energy and reliability impact of the proposed adaptive error protection strategy is presented. First, Figures 4.12 and 4.13 show the energy consumption for noise profiles 1 and 2, respectively. In these figures, the energy consumption is broken down (from bottom to top in each bar) as that consumed by the 32-bit data transfer (bus line), the energy consumed in the additional bit lines for supporting the coding scheme (code lines), the energy consumed by the encoders and decoders (coder)

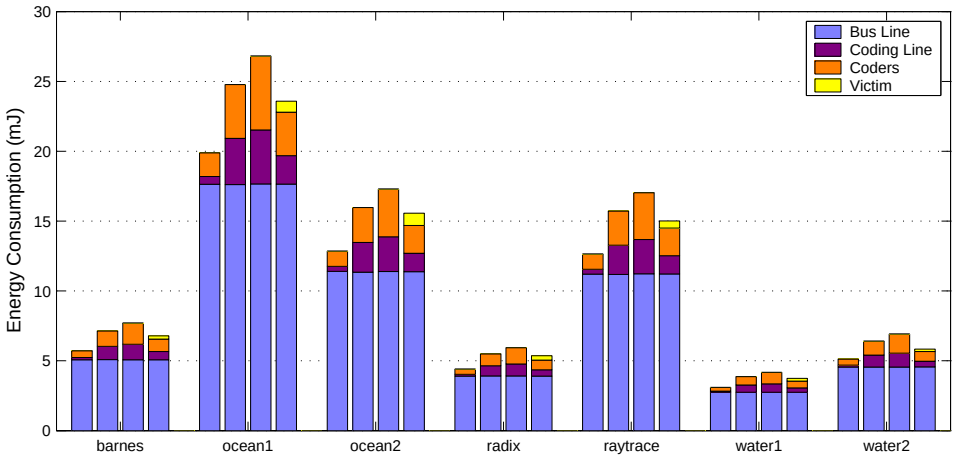

Fig. 4.13. Energy behavior with noise profile 2. The four bars for each benchmark from left to right correspond to using PAR, DED, TED and adaptive approach.

and the energy consumed by the victim line in the case of the proposed adaptive approach (victim). The energy for retransmissions is included.

It can be observed that in all the coding strategies the bulk of the energy is consumed by the data transfers and that the additional overhead for error protection ranges from 12.7% for PAR to 51.8% for TED.

The adaptive approach provides an average energy saving of 8% (12%) over the TED approach for noise profile 1 (noise profile 2) when both of them provide target undetectable error rate. In addition, the adaptive approach consumes 1% (5%) less energy using noise profile 1 (noise profile 2) even as compared to the DED scheme while providing better resilience to errors.

It should be emphasized that the relative gains of proposed scheme are based on the relative energy overhead of the coding schemes as compared to the energy consumed by the actual data transfers. For example, when error protection is required for a bus with

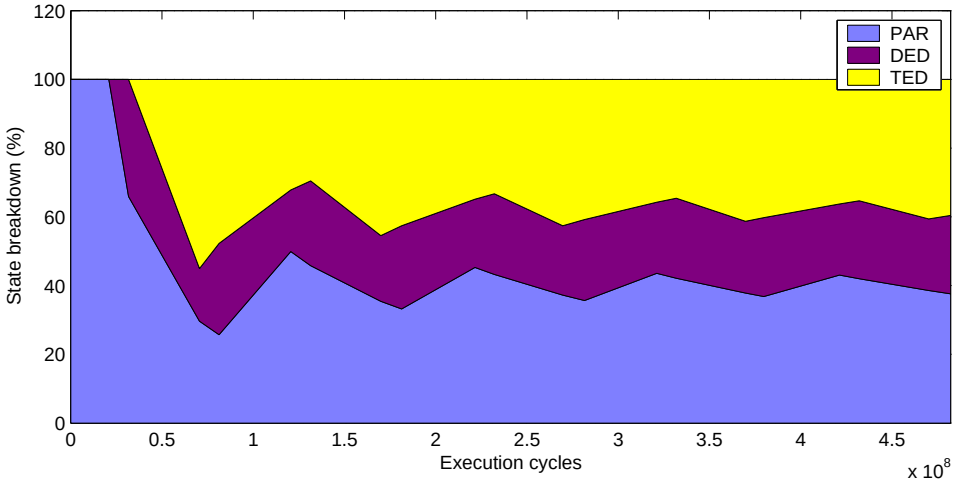

Fig. 4.14. State breakdown based on coding schemes used (noise profile 1).

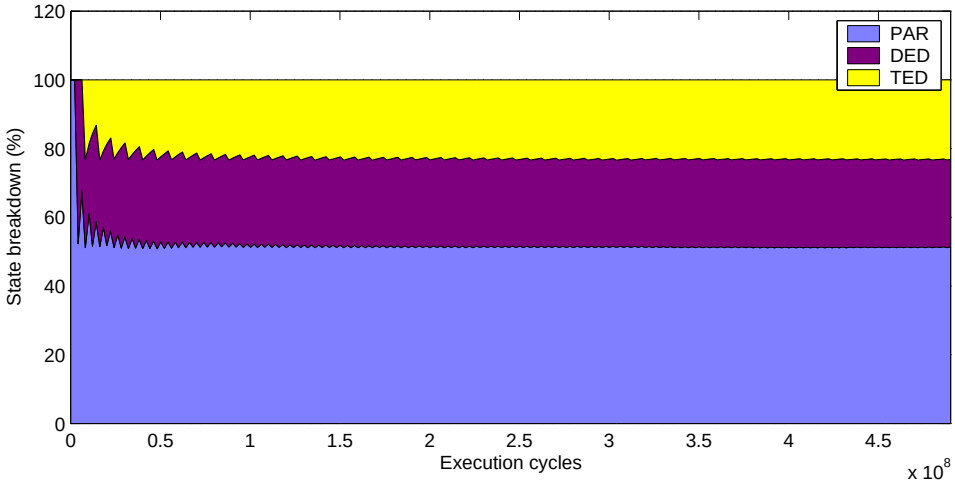

Fig. 4.15. State breakdown based on coding schemes used (noise profile 2).

Table 4.8. Error behavior when using noise profile 1. (D: Detected Error, U: Undetected Error)

	PAR		DED		TED		ADAPTIVE	
	D	U	D	U	D	U	D	U
barnes	25652	124	29590	1	30446	0	30435	0
ocean1	83346	461	96359	1	98876	0	98845	0
ocean2	57422	346	66183	0	67956	0	67927	0
radix	17293	97	19927	0	20446	0	20442	0
raytrace	57551	331	66547	1	68344	0	68318	0
water1	11346	53	13105	0	13435	0	13428	0
water2	20090	108	23175	0	23798	0	23785	0

half the capacitive load (2.5pF per bit line) considered in results shown in Figure 4.12, energy saving of the adaptive approach over TED increases to 13% (from 8%). In order to better understand the source of energy savings over TED, Figures 4.14 and 4.15 show the cumulative cycles spent in each error protection scheme (PAR, DED and TED) when the adaptive strategy is used. It can be observed that the adaptive strategy makes use of the PAR and DED schemes that have a smaller energy overhead around 60% (80%) of the time when using noise profile 1 (noise profile 2).

While in all experiments the adaptive strategy generated a better energy behavior than TED, some anomalous cases can result in proposed strategy consuming more energy than TED. For example, if the noise level is continuously severe requiring TED always, the overhead of the victim line makes the adaptive strategy more energy consuming than TED. As another example, if the number of transmissions is very low, the energy overhead of the victim line (that transitions every 10 cycles) can become a bottleneck. However, such cases did not happen in the realistic workloads in experiments.

Next, the error behavior of the different encoding strategies is studied. Tables 4.8 and 4.9 show the number of detected and undetected errors for noise profiles 1 and 2,

Table 4.9. Error behavior when using noise profile 2. (D: Detected Error, U: Undetected Error)

	PAR		**DED**		**TED**		**ADAPTIVE**	
	D	U	D	U	D	U	D	U
barnes	48768	476	56597	0	58066	0	57908	0
ocean1	171724	1732	199050	1	204323	0	203754	0
ocean2	109998	1056	127404	0	130759	0	130358	0
radix	37302	356	43219	1	44410	0	44273	0
raytrace	108992	1057	126442	2	129807	0	129488	0
water1	26163	250	30283	2	31108	0	31012	0
water2	30901	303	35681	0	36692	0	36686	0

respectively. It must be observed that proposed adaptive scheme is as powerful as the TED approach and leaves no errors undetected. In contrast, the DED approach leads to 3 (6) undetected errors using noise profile 1 (noise profile 2). Furthermore, PAR is clearly insufficient for the noise profiles used in this experimentation. Also, it must be observed that the number of actual errors that happen during transmission is different for the different schemes as DED and TED use additional bus lines for coding as compared to PAR. As a consequence of the additional lines, the number of errors increases.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The rapid scaling of VLSI technology has resulted in two important trends in digital circuits and systems design: a dramatic increase in energy consumption and a deterioration of reliability. This thesis consists of design methodologies and techniques for these two important issues at the micro-architecture level and makes the following contributions:

- Duplication of data and instructions at different levels of memory hierarchy is costly from the leakage energy perspective. Therefore, five different strategies are proposed to put L2 subblocks that hold duplicate copies of L1 blocks to low leakage consumption states. These strategies differ from each other with respect to the circuit type that they employ (state-destroying versus state-preserving), whether they conservatively or speculatively turn off L2 subblocks, and the time that the L2 subblocks are reactivated. Experimental results indicate that the best strategy (S-SP-Lazy) in terms of energy and energy-delay product is to place the L2 subblock into a state-preserving leakage control mode as soon as its contents are moved to L1 and to reactivate it only when it is accessed again. In addition, this strategy is integrated with a previously proposed optimization scheme, cache decay, and shows that the integrated strategy generates better energy results.

- Based on the observation that both private cache based systems and multi-ported shared cache architecture have advantages and disadvantages, an alternative cache organization is proposed to combine their advantages. The idea is to divide the L1 cache into multiple banks and then to connect the processors to banks using a crossbar interconnect. The results obtained using cycle-accurate simulation and SPLASH-2 benchmarks indicate that the energy benefits of proposed cache architecture range from 9% to 26% (when default simulation parameters are used) with respect to the private cache option. These savings come at the expense of a small degradation in performance in some benchmarks (whereas the proposed approach improves performance in some others).

- Crosstalk-induced delays are transition dependent. Designing bus cycle times based on worst-case crosstalk is overly pessimistic. Therefore, a crosstalk-aware interconnect is proposed that uses a faster clock and dynamically controls the number of cycles required for transmission based on the estimated delay of the data pattern to be transmitted. The experimental results show that the proposed approach improves performance by 34.1% as compared to the original pessimistic approach. The proposed scheme can also be combined with bus-invert coding scheme and spacing technique to further enhance the performance benefits.

- The influence of two low-power cache designs, drowsy cache and cache decay, on the soft error rate has been studied. Results show that the drowsy cache, while saving significant leakage energy, also incurs a significant increase in soft errors as compared to the original cache configuration. In contrast, the cache decay can

reduce both the leakage energy and the amount of effective soft errors. In addition, an adaptive error protection scheme with early-write-back policy in L1 data cache is proposed for the purpose of energy efficient error protection. Experimental results show that the proposed scheme can reduce dynamic energy of error protection components in L1 data cache by 11% on average without impacting the performance.

- It is observed that adopting the most comprehensive protection mechanism may not be very desirable from energy and performance perspectives. Therefore, an adaptive error protection scheme is proposed for on-chip interconnects, where the strength of the error coding scheme is varied to trade off between energy consumption and error protection. Experimental results indicate that the proposed strategy achieves the same level of reliability as the triple error detection scheme while reducing the energy consumption of the latter. In addition, it outperforms the double error detection scheme from both energy and reliability angles.

5.2 Future Work

The work of this thesis brings attention to the following open issues that need to be investigated in the future:

- *More complete and accurate energy/power modeling for microprocessors:* This thesis focus on energy consumption issues of cache memories and on-chip interconnects. Besides that, pipeline stages, register files, function units, and branch units

are also important components in micro-processor cores and consume a large portion of the total chip power budget. Investigating the behavior of cooperation and synchronization among different components in processors will provide more accurate modeling and calculation of total chip energy consumption. It will also provide potential chances to achieve more energy gain via chip-level energy management. Another feature that affects the accuracy of the energy/power model is the temperature of the chip. Thermal hotspots increase power consumption as leakage currents increase exponentially with temperature. They also degrade reliability by accelerating many failure mechanisms. Therefore, including thermal feedback into the energy/power model is a necessity.

- *Multicore and multiprocessor architecture:* Along with the evolution of the current technology generation, more and more transistors are being integrated into a single chip. At the same time, the propagation delay of long on-chip interconnects is becoming more significant than the delay of gates in deep sub-micron technology. Therefore, traditional complex and single processor architecture will be the bottleneck for performance improvement. Decentralized architectures, such as on-chip multicores and multiprocessors, provide promising solutions for scalable improvement of performance, simplification of complex design, and reduction of validation cost. They also bring potential opportunities at the processor level for energy saving and reliability improvement. For example, different processors may operate at different clock frequencies with different supply voltages based on the information of workload at running time. Thus, tasks can be scheduled and migrated

among processors for energy saving, and data and operations can be duplicated on different processors to achieve high reliability.

- *Adaptive micro-architecture design:* With the scaling of VLSI technology, design metrics is no longer one-dimensional. As a result, not only performance, but energy consumption and reliability should also be considered early in the system design cycle. In previous work [50], we saw that one technique may benefit one metric while hurting another one. Also, the traditional design method is aimed at the worst-case scenario. Although this method can guarantee the functionality of systems, it is inefficient in terms of performance, energy, and reliability. Therefore, adaptive micro-architecture design may either provide a holistic optimization for the requirements of performance, energy-efficiency, and reliability or a smart balance among these three requirements under different environments and operation conditions.

References

[1] Berkeley Predictive Technology Model.
http://www-device.eecs.berkeley.edu/~ptm/interconnect.html.

[2] IBM Power4 Project.
http://www-1.ibm.com/servers/eserver/pseries/hardware/whitepapers/power4.html.

[3] Intel Coroporation. Intel® Itanium® 2 Processor Datasheet.

[4] Intel Coroporation. Intel® Pentium® III Processor for the SC242 at 450 MHz to 1.0 GHz Datasheet.

[5] SPEC CPU2000 Benchmark. http://www.spec.org/.

[6] Stanford Hydra Project. http://www-hydra.stanford.edu.

[7] Sun MAJC Project. http://www.sun.com/products/processors/MAJC/.

[8] A. Agarwal, H. Li, and K. Roy. DRG-cache: a data retention gated-ground cache for low power. In *Proceedings of Conference on Design Automation (DAC'02)*, pages 473–478, 2002.

[9] D. Albonesi. Selective cache ways: on-demand cache resource allocation. In *Proceedings of International Symposium on Microarchitecture (MICRO'99)*, pages 248–259, 1999.

[10] C. Alpert, A. Devgan, and S. Quay. Buffer insertion for noise and delay optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(11):1633–1645, 1999.

[11] R. Arunachalam, E. Acar, and S. Nassif. Optimal shielding/spacing metrics for low power design. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pages 167–172, 2003.

[12] T. Austin. DIVA: a reliable substrate for deep submicron microarchitecture design. In *Proceedings of International Symposium on Microarchitecture (MICRO'99)*, pages 196–207, 1999.

[13] N. Azizi, A. Moshovos, and F. Najm. Low-leakage asymmetric-cell SRAM. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED'02)*, pages 48–51, 2002.

[14] R. Baumann. The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction. In *Technical Digest of International Electron Devices Meeting (IEDM'02)*, pages 329–332, 2002.

[15] D. Bertozzi, L. Benini, and G. de Micheli. Low power error resilient encoding for on-chip data buses. In *Proceedings of Conference on Design, Automation and Test in Europe (DATE'02)*, pages 102–109, 2002.

[16] D. Burger and T. Austin. The SimpleScalar tool set, version 2.0. *SIGARCH Comput. Archit. News*, 25(3):13–25, 1997.

[17] J. Butts and G. Sohi. A static power model for architects. In *Proceedings of International Symposium on Microarchitecture (MICRO'00)*, pages 191–201, 2000.

[18] A. Chandrakasan, W. Bowhill, and F. Fox. *Design of High-Performance Microprocessor Circuits*. IEEE Press, 2001.

[19] G. Chen, R. Shetty, M. Kandemir, N. Vijaykrishnan, M. Irwin, and M. Wolczko. Tuning garbage collection in an embedded java environment. In *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA'02)*, pages 92–103, 2002.

[20] H. Chen and D. Ling. Power supply noise analysis methodology for deep-submicron VLSI chip design. In *Proceedings of Conference on Design Automation (DAC'97)*, pages 638–643, 1997.

[21] W. Cheng and M. Pedram. Memory bus encoding for low power: a tutorial. In *Proceedings of International Symposium on Quality Electronic Design (ISQED'01)*, pages 199–204, 2001.

[22] R. Cooksey and D. Grunwald. Characterization of the SPEC2000 benchmark suite. Technical report, http://www.cs.colorado.edu/ rcooksey/pubs.html, 2001.

[23] M. Cuviello, S. Dey, X. Bai, and Y. Zhao. Fault modeling and simulation for crosstalk in system-on-chip interconnects. In *Proceedings of International Conference on Computer-Aided Design (ICCAD'99)*, pages 297–303, 1999.

[24] W. Dally and J. Poulton. *Digital Systems Engineering*. Cambridge University Press, 1998.

[25] V. Degalahal, Lin Li, N. Vijaykrishnan, M. Irwin, and M. Kandemir. Soft errors issues in low power caches. *Accepted by IEEE Transactions on Very Large Scale Integration Systems*.

[26] V. Degalahal, N. Vijaykrishnan, and M. Irwin. Analyzing soft errors in leakage optimized SRAM design. In *Proceedings of International Conference on VLSI Design*, pages 227–233, 2003.

[27] D. Ernst, N. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: A low-power pipeline based on circuit-level timing speculation. In *Proceedings of International Symposium on Microarchitecture (MICRO'03)*, pages 7–18, 2003.

[28] K. Flautner, N. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: simple techniques for reducing leakage power. In *Proceedings of International Symposium on Computer Architecture (ISCA'02)*, pages 148–157, 2002.

[29] E. Geethanjali, V. Narayanan, and M. Irwin. An analytical power estimation model for crossbar interconnects. In *Proceedings of IEEE International ASIC/SOC Conference*, pages 119–123, 2002.

[30] P. Gupta and A. Kahng. Wire swizzling to reduce delay uncertainty due to capacitive coupling. In *Proceedings of International Conference on VLSI Design*, pages 431–436, 2004.

[31] P. Hazucha and C. Svensson. Impact of CMOS technology scaling on the atmospheric neutron soft error rate. *IEEE Transactions on Nuclear Science*, 47(6):2586–2594, 2000.

[32] R. Hegde and N. Shanbhag. Toward achieving energy efficiency in presence of deep submicron noise. *IEEE Transactions on Very Large Scale Integration Systems*, 8(4):379–391, 2000.

[33] R. Hegde and N. Shanbhag. Soft digital signal processing. *IEEE Transactions on Very Large Scale Integration Systems*, 9(6):813–823, 2001.

[34] S. Heo, K. Barr, M. Hampton, and K. Asanovic. Dynamic fine-grain leakage reduction using leakage-biased bitlines. In *Proceedings of International Symposium on Computer Architecture (ISCA'02)*, pages 137–147, 2002.

[35] Z. Hu, P. Juang, P. Diodato, S. Kaxiras, K. Skadron, M. Martonosi, and D. Clark. Managing leakage for transient data: decay and quasi-static 4T memory cells. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED'02)*, pages 52–55, 2002.

[36] K. Inoue, T. Ishihara, and K. Murakami. Way-predicting set-associative cache for high performance and low energy consumption. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED'99)*, pages 273–275, 1999.

[37] K. Itoh, K. Sasaki, and Y. Nakagome. Trends in low-power RAM circuit technologies. *Proceedings of the IEEE*, 83(4):524–543, 1995.

[38] N. Jouppi and S. Wilton. Tradeoffs in two-level on-chip caching. In *Proceedings of International Symposium on Computer Architecture (ISCA'94)*, pages 34–45, 1994.

[39] T. Karnik, B. Bloechel, K. Soumyanath, V. De, and S. Borkar. Scaling trends of cosmic ray induced soft errors in static latches beyond 0.18? In *Digest of Technical Papers of Symposium on VLSI Circuits*, pages 61–62, 2001.

[40] H. Kaul, D. Sylvester, and D. Blaauw. Active shields: a new approach to shielding global wires. In *Proceedings of Great Lakes Symposium on VLSI (GLSVLSI'02)*, pages 112–117, 2002.

[41] S. Kaxiras, Z. Hu, and M. Martonosi. Cache decay: exploiting generational behavior to reduce cache leakage power. In *Proceedings of International Symposium on Computer Architecture (ISCA'01)*, pages 240–251, 2001.

[42] B. Kiani. Static crosstalk analysis assures silicon success. *EE Times*, June 5, 2002.

[43] K. Kim, S. Jung, U. Narayanan, C. Liu, and S. Kang. Noise-aware power optimization for on-chip interconnect. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED'00)*, pages 108–113, 2000.

[44] N. Kim, K. Flautner, D. Blaauw, and T. Mudge. Drowsy instruction caches: leakage power reduction using dynamic voltage scaling and cache sub-bank prediction. In *Proceedings of International Symposium on Microarchitecture (MICRO'02)*, pages 219–230, 2002.

[45] S. Kim and A. Somani. Area efficient architectures for information integrity in cache memories. In *Proceedings of International Symposium on Computer Architecture (ISCA'99)*, pages 246–255, 1999.

[46] S. Kim, N. Vijaykrishnan, M. Kandemir, A. Sivasubramaniam, M. Irwin, and E. Geethanjali. Power-aware partitioned cache architectures. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED'01)*, pages 64–67, 2001.

[47] J. Kin, M. Gupta, and W. Mangione-Smith. The filter cache: an energy efficient memory structure. In *Proceedings of International Symposium on Microarchitecture (MICRO'97)*, pages 184–193, 1997.

[48] T. Koyama, K. Inoue, H. Hanaki, M. Yasue, and E. Iwata. A 250-MHz single-chip multiprocessor for audio and video signal processing. *IEEE Journal of Solid-State Circuits*, 36(11):1768–1774, 2001.

[49] C. Lee, M. Potkonjak, and W. Mangione-Smith. MediaBench: a tool for evaluating and synthesizing multimedia and communicatons systems. In *Proceedings of International Symposium on Microarchitecture (MICRO'97)*, pages 330–335, 1997.

[50] Lin Li, V. Degalahal, N. Vijaykrishnan, M. Kandemir, and M. Irwin. Soft error and energy consumption interactions: A data cache perspective. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED'04)*, pages 132–137, 2004.

[51] Lin Li, I. Kadayif, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. Irwin, and A. Sivasubramaniam. Leakage energy management in cache hierarchies. In *Proceedings of International Conference on Parallel Architectures and Compilation Techniques (PACT'02)*, pages 131–140, 2002.

[52] Lin Li, I. Kadayif, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. Irwin, and A. Sivasubramaniam. Managing leakage energy in cache hierarchies. *Journal of Instruction-Level Parallelism*, 5, 2003.

[53] Lin Li, N. Vijaykrishnan, M. Kandemir, and M. Irwin. Adapative error protection for energy efficiency. In *Proceedings of International Conference on Computer-Aided Design (ICCAD'03)*, pages 2–7, 2003.

[54] Lin Li, N. Vijaykrishnan, M. Kandemir, and M. Irwin. A crosstalk aware interconnect with variable cycle transmission. In *Proceedings of Conference on Design, Automation and Test in Europe (DATE'04)*, pages 102–107, 2004.

[55] Lin Li, N. Vijaykrishnan, M. Kandemir, M. Irwin, and I. Kadayif. CCC: Crossbar connected caches for reducing energy consumption of on-chip multiprocessors. In *Proceedings of Euromicro Symposium on Digital System Design (DSD'03)*, pages 41–48, 2003.

[56] J. Ma and L. He. Formulae and applications of interconnect estimation considering shield insertion and net ordering. In *Proceedings of International Conference on Computer-Aided Design (ICCAD'01)*, pages 327–332, 2001.

[57] J. Maiz, S. Hareland, K. Zhang, and P. Armstrong. Characterization of multi-bit soft error events in advanced SRAMs. In *Technical Digest of International Electron Devices Meeting (IEDM'03)*, pages 21.4.1–21.4.4, 2003.

[58] N. Manjikian. Multiprocessor enhancements of the SimpleScalar tool set. *SIGARCH Comput. Archit. News*, 29(1):8–15, 2001.

[59] D. Mavis and P. Eaton. Soft error rate mitigation techniques for modern microcircuits. In *Proceedings of Reliability Physics Symposium*, pages 216–225, 2002.

[60] T. May and M. Woods. Alpha-particled-induced soft errors in dynamic memories. *IEEE Transactions on Electron Devices*, ED-26(1), 1979.

[61] C. McNairy and D. Soltis. Itanium 2 processor microarchitecture. *IEEE Micro*, 23(2):44–55, 2003.

[62] K. Mohanram and N. Touba. Cost-effective approach for reducing soft error failure rate in logic circuits. In *Proceedings of International Test Conference (ITC'03)*, pages 893–901, 2003.

[63] G. Moore. No exponential is forever: but "forever" can be delayed! In *Digest of Technical Papers of International Solid-State Circuits Conference (ISSCC'03)*, pages 20–23, 2003.

[64] S. Mukherjee, M. Kontz, and S. Reinhardt. Detailed design and evaluation of redundant multithreading alternatives. In *Proceedings of International Symposium on Computer Architecture (ISCA'02)*, pages 99–110, 2002.

[65] B. Nikolic. State-preserving leakage control mechanisms. *Gigascale Silicon Research Center Annual Report*, Sept. 2001.

[66] K. Olukotun, B. Nayfeh, L. Hammond, K. Wilson, and K. Chang. The case for a single-chip multiprocessor. In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'96)*, pages 2–11, 1996.

[67] A. Orailoglu and R. Karri. A design methodology for the high-level synthesis of fault-tolerant asics. In *Proceedings of Workshop on VLSI Signal Processing*, pages 417–426, 1992.

[68] M. Powell, A. Agarwal, T. Vijaykumar, B. Falsafi, and K. Roy. Reducing set-associative cache energy via way-prediction and selective direct-mapping. In *Proceedings of International Symposium on Microarchitecture (MICRO'01)*, pages 54–65, 2001.

[69] H. Qin and J. Rabaey. Leakage suppression of embedded memories. *Gigascale Silicon Research Center Annual Review*, 2002.

[70] J. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits: A Design Perspective*. Prentice-Hall, 2002.

[71] N. Seifert, D. Moyer, N. Leland, and R. Hokinson. Historical trend in alpha-particle induced soft error rates of the alpha microprocessor. In *Proceedings of International Reliability Physics Symposium*, pages 259–265, 2001.

[72] N. Shanbhag, K. Soumyanath, and S. Martin. Reliable low-power design in the presence of deep submicron noise (embedded tutorial session). In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED'00)*, pages 295–302, 2000.

[73] K. Shepard and V. Narayanan. Noise in deep submicron digital design. In *Proceedings of International Conference on Computer-Aided Design (ICCAD'96)*, pages 524–531, 1996.

[74] P. Shivakumar and N. Jouppi. CACTI 3.0: An integrated cache timing, power and area model. Technical report, Western Research Lab (WRL), Feb 2001.

[75] K. Skadron and D. Clark. Design issues and tradeoffs for write buffers. In *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA'97)*, pages 144–155, 1997.

[76] P. Sotiriadis and A. Chandrakasan. Reducing bus delay in submicron technology using coding. In *Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC'01)*, pages 109–114, 2001.

[77] S. Srinivasan, Lin Li, and N. Vijaykrishnan. Simultaneous partitioning and frequency assignment for on-chip bus architectures. In *Proceedings of Conference on Design, Automation and Test in Europe (DATE'05)*, pages 218–223, 2005.

[78] M. Stan and W. Burleson. Bus-invert coding for low-power I/O. *IEEE Transactions on Very Large Scale Integration Systems*, 3(1):49–58, 1995.

[79] M. Takahashi, H. Takano, E. Kaneko, and S. Suzuki. A shared-bus control mechanism and a cache coherence protocol for a high-performance on-chip multiprocessor. In *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA'96)*, pages 314–322, 1996.

[80] P. van der Meer, A. van Staveren, and A. van Roermund. Ultra-low standby-currents for deep sub-micron VLSI CMOS circuits: Smart series switch. In *Proceedings of International Symposium on Circuits and Systems (ISCAS'00), Volume 4*, pages 1–4, 2000.

[81] B. Victor and K. Keutzer. Bus encoding to prevent crosstalk delay. In *Proceedings of International Conference on Computer-Aided Design (ICCAD'01)*, pages 57–63, 2001.

[82] L. Villa, M. Zhang, and K. Asanovic. Dynamic zero compression for cache energy reduction. In *Proceedings of International Symposium on Microarchitecture (MICRO'00)*, pages 214–220, 2000.

[83] J. Wakerly. *Digital Design: Principles and Practices*. Prentice-Hall, 2000.

[84] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The SPLASH-2 programs: characterization and methodological considerations. In *Proceedings of International Symposium on Computer Architecture (ISCA'95)*, pages 24–36, 1995.

[85] F. Worm, P. Ienne, P. Thiran, and G. de Micheli. An adaptive low-power transmission scheme for on-chip networks. In *Proceedings of International Symposium on System Synthesis (ISSS'02)*, pages 92–100, 2002.

[86] F. Wrobel, J. Palau, M. Calvet, O. Bersillon, and H. Duarte. Simulation of nucleon-induced nuclear reactions in a simplified SRAM structure: Scaling effects on SEU and MBU cross sections. *IEEE Transactions on Nuclear Science*, 48(6):1946–1952, 2001.

[87] Y. Xie, Lin Li, M. Kandemir, N. Vijaykrishnan, and M. Irwin. Reliability-aware co-synthesis for embedded systems. In *Proceedings of International Conference on Application-Specific Systems, Architectures and Processors (ASAP'04)*, pages 41–50, 2004.

[88] S. Yang, M. Powell, B. Falsafi, K. Roy, and T. Vijaykumar. An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance I-caches. In *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA'01)*, pages 147–157, 2001.

[89] W. Zhang, S. Gurumurthi, M. Kandemir, and A. Sivasubramaniam. ICR: In-cache replication for enhancing data cache reliability. In *Proceedings of International Conference on Dependable Systems and Networks (DSN'03)*, pages 291–300, 2003.

[90] G. Zhong, C. Koh, and K. Roy. A twisted-bundle layout structure for minimizing inductive coupling noise. In *Proceedings of International Conference on Computer-Aided Design (ICCAD'00)*, pages 406–411, 2000.

[91] H. Zhou, M. Toburen, E. Rotenberg, and T. Conte. Adaptive mode control: a static-power-efficient cache design. In *Proceedings of International Conference on Parallel Architectures and Compilation Techniques (PACT'01)*, pages 61–70, 2001.

[92] J. Ziegler. Terrestrial cosmic ray intensities. *IBM Journal of Research and Development*, 40(1):19–39, 1996.

Vita

Lin Li received his B.S. and M.S. degrees in computer science from Peking University, China, in 1997 and 2000, respectively. He joined Microsystems Design Lab (MDL), part of the Embedded and Mobile Computing Design Center (EMC^2), in the Department of Computer Science and Engineering at the Pennsylvania State University in 2000 to pursue his Ph.D. degree under the guidance of Dr. Mary Jane Irwin and Dr. Vijaykrishnan Narayanan. His research interests include computer architecture and micro-architecture, system simulation, performance analysis, digital circuit design, low-power design, and reliable system design. He is a student member of IEEE and ACM.