ENERGY AND RELIABILITY IN FUTURE NOC INTERCONNECTED CMPS

A Dissertation

by

HYUNGJUN KIM

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2013

Major Subject: Computer Engineering

ABSTRACT

In this dissertation, I explore energy and reliability in future NoC (Network-on-Chip) interconnected CMPs(chip multiprocessors) as they have become a first-order constraint in future CMP design.

In the first part, we target the root cause of network energy consumption through techniques that reduce link and router-level *switching activity*. We specifically focus on memory subsystem traffic, as it comprises the bulk of NoC load in a CMP. By transmitting only the flits that contain words that we predicted would be useful using a novel spatial locality predictor, our scheme seeks to reduce network activity. We aim to further lower NoC energy consumption through microarchitectural mechanisms that inhibit datapath switching activity caused by unused words in individual flits. Using simulation-based performance studies and detailed energy models based on synthesized router designs and different link wire types, we show that (a) the prediction mechanism achieves very high accuracy, with an average rate of false-unused prediction of just 2.5%; (b) the combined NoC energy savings enabled by the predictor and microarchitectural support are 36% on average and up to 57% in the best case; and (c) there is no system performance penalty as a result of this technique.

In the second part, we present a method for dynamic voltage/frequency scaling of networks-on-chip and last level caches in CMP designs, where the shared resources form a single voltage/frequency domain. We develop a new technique for monitoring and control and validate it by running PARSEC benchmarks through full system simulations. These techniques reduce energy-delay product by 46% compared to a state-of-the-art prior work.

In the third part, we develop critical path models for HCI- and NBTI-induced wear assuming stress caused under realistic workload conditions, and apply them onto the interconnect microarchitecture. A key finding from this modeling is that, counter to prevailing wisdom, wearout in the CMP on-chip interconnect is correlated with a lack of load observed in the NoC routers, rather than high load. We then develop a novel wearout-decelerating scheme in which routers under low load have their wearout-sensitive components exercised without significantly impacting the router's cycle time, pipeline depth, area or power consumption. We subsequently show that the proposed design yields a $13.8 \sim 65\times$ increase in CMP lifetime.

DEDICATION

To my wife, Minyoung, and my loving parents

ACKNOWLEDGEMENTS

I would like to acknowledge many people without whom this work would not have been possible. First and foremost I would like to thank my advisor, Paul V. Gratz, for his patient support and endless guidance. He always waited for me to make progress by myself while teaching me invaluable knowledge and skills. He has been also patient with my communication skills. He always tried to understand what I was trying to say, and never gave up on any of my trivial idea. I was fortunate to have him as my advisor, and in fact, now, he is my role model as a researcher and as a co-worker.

I was also lucky to meet Eun Jung Kim in my first year in Texas A&M University. She actually was the one who introduced the research field that I ended up focusing on. She taught me the fundamental knowledge of interconnection networks and encouraged me to study in this area. Without her, this work wouldn't exist.

I would like to thank my other committee members, Daniel Jiménez, Jiang Hu, and Henry Pfister for their helpful feedback on my research. Special thanks to Prof. Jiménez for the idea of the spatial locality prediction which became a significant part of this work. I also really enjoyed the work with Prof. Hu and his students, Xi Chen and Zheng Xu, which is essential in this work, as well.

I would also like to thank co-authors, Boris Grot, Pritha Ghoshal, David Kadjo, Vassos Soteriou, and Arseniy Vitkovskiy. Also, I would like to thank the folks from Intel including Ümit Ogras, Raid Ayoub and Michael Kishinevsky who happened to be my mentors while I was interning at Intel. With their feedback and advice, I could expand the area of my research interests taking myself out of my comfort zone, little by little.

Many thanks to the members of CAMSIN group, a research group full of brilliant students and in which Prof. Gratz has fostered an excellent collaborative environment. Special thanks to Ehsan Fatehi, Mark Browning and Jinchun Kim for their dedicated effort, and time they took to setup the infrastructure of experiments for other CAMSIN members.

Thanks to Prof. Eun Jung Kim, again, I was also fortunate to join her group for awhile, and meet Yuho Jin and Lei Wang with whom I wrote my very first research paper. Special thanks to Yuho for teaching me many useful tips and skills required in this area of study, which became the foundation of what I have now in my tool box. I can't forget the days and nights with Lei struggling with our first paper, and the feeling of achievement at the moment it got accepted to be published.

On a more personal note, first, I would like to thank two office mates, Yongoh Lee and Yongtae Kim. They have been the first people with whom I could discuss concerns, and with whom I could share any good and bad news. In fact, they are very supportive not only mentally but also of my research. Also, KATC, Korean Aggies Tennis Club, has been inevitable in my life in College Station. It contributed many memorable moments and stress relief to my five years in College Station.

Finally and most importantly, I would like to thank my family. I don't know how to thank my parents enough for the sacrifices they have made for me. Without your love and support, this journey would not have been possible. I would also like to thank the love of my life, Minyoung. Minyoung's constant love and sense of humor make every day special. I am always excited to be with you for the rest of our lives.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

While process technology scaling continues to provide more transistors, the transistor performance and power gains that accompany process scaling have largely ceased [39]. Chip-multiprocessor (CMP) designs achieve greater efficiency than traditional uniprocessors through concurrent parallel execution of multiple programs or threads. Meanwhile, the size of on-chip cache has been increased to overcome the memory wall caused by off-chip memory transactions. The on-chip cache, in general, is shared by multiple cores and divided into multiple slices each of which resides at each core. The increases in size and slice count of this shared cache result in the increases of on-chip communication bandwidth and power consumption. Indeed, the energy consumptions of on-chip communication fabrics and shared, last-level caches (LLCs) have grown to occupy a significant portion of the overall chip power, as much as 30% of total power in recent Intel's single-chip cloud computer [36].

As the core count in chip-multiprocessor (CMP) systems increases, networks-on-chip (NoCs) present a scalable alternative to traditional, bus-based designs for interconnection between processor cores [21]. As in most current VLSI designs, power efficiency has also become a first-order constraint in NoC design. In fact, it is becoming increasingly difficult to ignore the power consumption of NoC. The energy consumed by the NoC itself is as significant as 28% of the per-tile power in the Intel Teraflop chip [35] and 36% of the total chip power in MIT RAW chip [78].

At the same time, with the continuous down-scaling of process technologies, reliability has become an important concern in NoC design [26, 63, 9]. Deep sub-micron CMOS process technology is marred by increasing susceptibility to wearout. Prolonged operational stress gives rise to accelerated wearout and failure, due to several physical failure mechanisms, including Hot Carrier Injection (HCI) and Negative Bias Temperature Instability (NBTI). Each failure mechanism correlates with different usage-based stresses, all of which can eventually generate permanent faults. While the wearout of an individual core in many core CMPs may not necessarily be catastrophic for the entire parallel-processing system, a single fault in the interprocessor Network-on-Chip (NoC) fabric could render the entire chip useless, as it could lead to protocol-level deadlocks, or even partition away vital components such as the memory controller or other critical I/O.

1.1 Thesis Statement

In this dissertation I propose that microarchitectural techniques can be used to address the twin problems of energy-efficiency and reliability in future process technologies. This dissertation describes two designs to improve energy efficiency in the NoCs and LLCs, and a solution to improve reliability of the NoCs.

It has been observed that cache lines fetched to level-1 caches are not fully utilized, and merely 40% of words in the cache lines are actually referenced by processors. A significant portion of NoC dynamic power is consumed by those unused words, which turn out to be predictable. This dissertation proposes a novel spatial locality predictor which speculates those unused words, and a novel packet composition scheme which encodes those words to eliminate bit transition along the NoC datapath.

This dissertation also explores a new Dynamic Voltage and Frequency Scaling (DVFS) scheme to reduce NoC and LLC power consumption. It is necessary to choose a metric on which to base a DVFS control. The metric should be controlled by the frequency level that we adjust and it should reflect the overall system performance. This dissertation proposes to use the throughput of *uncore*, which refers to NoC and LLC together, as the metric of the DVFS policy for uncore system, and proves that such control results in improved energy efficiency compared to previous schemes.

It also addresses reliability issues in NoC design. Reliability of NoC becomes more important as the current and future chip-multiprocessors rely on it more. The scaling transistor size even accelerates the failure mechanisms, which will impact the critical path of the routers. This dissertation discusses the cause of such failure mechanisms in NoC and introduces a novel wear-resistant NoC router microarchitecture.

1.2 Dissertation Organization

This document is organized as follows. Chapter 2 introduces a novel energy reduction technique for the NoCs and LLCs through spatial locality speculation. Chapter 3 includes a new DVFS policy to achieve energy efficient NoCs and LLCs. Chapter 4 describes a new design method for reliable NoCs which mitigates the dominating failure mechanisms efficiently.

2. ENERGY EFFICIENCY THROUGH SPATIAL LOCALITY PREDICTION[*]

2.1 Introduction

In this work we present a novel technique to reduce energy consumption for CMP core interconnect leveraging spatial locality speculation to identify unused cache block words. In particular, we propose to predict which words in each cache block fetch will be used and leverage that prediction to reduce dynamic energy consumption in the NoC channels and routers through diminished switching activity.

2.1.1 Motivation

Current CMPs employ cache hierarchies of multiple levels prior to main memory [58, 3]. Caches organize data into *blocks* containing multiple contiguous words in an effort to capture spatial locality and reduce the likelihood of subsequent misses. Unfortunately, applications often do not fully utilize all the words fetched for a given cache block, as recently noted by Pujara et al. [66].

Figure 2.1 shows the percentage of words utilized in the PARSEC multithreaded benchmark suite [10]. On average, 61% of cache block words in the PARSEC suite benchmarks will never be referenced and represent energy wasted in transference through the memory hierarchy[1]. In this work, we focus on the waste associated with traditional approaches to spatial locality, in particular the wasted energy and power caused by large cache blocks containing unused data.

[*]©2012 IEEE. Reprinted, with permission, from Hyungjun Kim, Boris Grot, Paul V Gratz, Daniel Jimenez, NOCS Special Section: Spatial Locality Speculation to Reduce Energy in Chip-Multiprocessor Networks-on-Chip, IEEE Transactions on Computers, 2012

[1]Versus 64-byte lines, 32-byte lines reduces unused words to 45%, however it increases AMAT 11% (See Section 2.5.3).

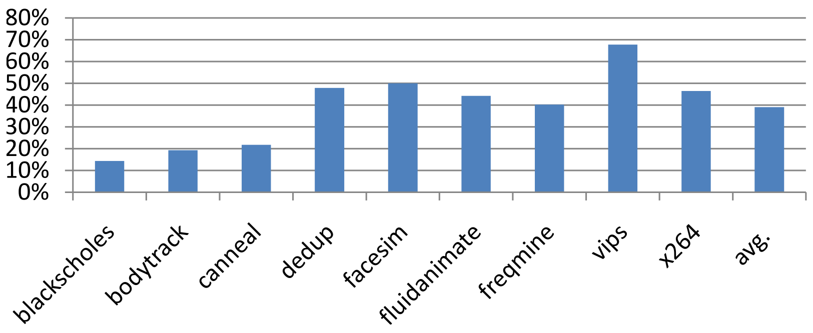


Figure 2.1: Percentage of 64-byte block, cache words utilized per block in the PARSEC multithreaded benchmarks.

2.1.2 CMP Interconnect

Networks-on-chip (NoCs) purport to be a scalable interconnect to meet the increasing bandwidth demands of future CMPs [21]. NoCs must be carefully designed to meet many constraints. Energy efficiency, in particular, is a challenge in future NoCs as the energy consumed by the NoC itself is a significant fraction of the total chip power [35, 78]. The NoC packet datapath, consisting of the link, crossbar and FIFOs, consumes a significant portion of interconnect power, 55% of network power in the Intel Teraflop chip [35].

Existing NoCs implement channels with relatively large link bit-widths (>=128 bits) [28, 29], a trend expected to continue as more wire density becomes available in future process technologies. These high-bandwidth link wires reduce the latency of cache block transmission by allowing more words to be transferred in each cycle, minimizing serialization latency for large packets. In some cases, however, not all the words in a flit are useful to the processor. In particular, unused cache block words represent wasted power and energy. We propose to use spatial locality speculation to leverage unused words of the block transfers between the lower and upper cache levels to save energy.

2.1.3 Proposed Technique

The goal of the proposed technique is to reduce dynamic energy in CMP interconnect by leveraging spatial locality speculation on the expected used words in fetched cache blocks in CMP processor memory systems.

The work makes the following contributions:

- A novel intra-cache-block spatial locality predictor, to identify words unlikely to be used before the block is evicted.

- A static packet encoding technique which leverages spatial locality prediction to reduce the network activity factor, and hence dynamic energy, in the NoC routers and links. The static encoding requires no modification to the NoC and minimal additions to the processor caches to achieve significant energy savings with negligible performance overhead.

- A complementary dynamic packet encoding technique which facilitates additional energy savings in NoC links and routers via light-weight microarchitectural enhancements.

In a 16-core CMP implemented in a 45-nm process technology, the proposed technique achieves an average of $\sim$35% savings in total dynamic interconnect energy at the cost of less than 1% increase in memory system latency.

The rest of this chapter is organized as follows: Section 4.7 discusses the related work and background in caches, NoCs and power efficiency on-chip to provide the intuition behind our power saving flit-encoding technique. Section 2.3 discusses our proposed technique in detail, including the proposed spatial locality predictor and the proposed packet encoding schemes. Section 4.6 and 2.5 present the experimental setup and the results. Finally, we conclude in Section 4.8.

2.2 Background and Related Work

2.2.1 Dynamic Power Consumption

When a bit is transmitted over interconnect wire or stored in an SRAM cell, dynamic power is consumed as a result of a capacitive load being charged up and also due to transient currents during the momentary short from Vdd to Gnd while transistors are switching. Dynamic power is not consumed in the absence of switching activity. Equation 2.1 shows the dynamic and short-circuit components of power consumption in a CMOS circuit.

$$P = \alpha \cdot C \cdot V^2 \cdot f + t \cdot \alpha \cdot V \cdot I_{short} \cdot f \qquad (2.1)$$

In the equation, P is the power consumed, C is the switched capacitance, V is the supplied voltage, and F is the clock frequency. α represents the activity factor, which is the probability that the capacitive load C is charged in a given cycle. C, V, and F are a function of technology and design parameters. In systems that support dynamic voltage-frequency scaling (DVFS), V and F might be tunable at run time; however, dynamic voltage and frequency adjustments typically cannot be done at a fine spatial or temporal granularity [75]. In this work, we target the activity factor, α, as it enables dynamic energy reduction at a very fine granularity.

2.2.2 NoC Power and Energy

Researchers have recently begun focusing on the energy and power in NoCs, which have been shown to be significant contributors to overall chip power and energy consumption [35, 78, 44, 6].

One effective way to reduce NoC power consumption is to reduce the amount of data sent over the network. To that extent, recent work has focused on compression at the cache and network levels [23, 42] as an effective power-reduction technique. In general, however, compression techniques have overheads in terms of latency for compression and decompression. The technique we present is orthogonal to, and could potentially be used in conjunction with, these loss-less compression techniques to further reduce power. Our work seeks to reduce the amount of data transmitted through identification and removal of useless words; traditional compression could be used to more densely pack the remaining data.

Researchers have also proposed a variety of techniques to reduce interconnect energy consumption through reduced voltage swing [83]. Schinkel et al. propose a scheme which uses a capacitative transmitter to lower the signal swing to 125 mV without the use of an additional low-voltage power supply [72]. In this work we evaluate our prediction and packet encoding techniques for links composed of both full-signal swing as well as low-signal swing wires.

NoC router microarchitectures for low power have also been explored to reduce power in the transmission of data which is much smaller than a flit. Das et al. propose a novel crossbar and arbiter design that supports concurrent transfers of multiple flits on a single link to improve bandwidth utilization. [22].

Finally, static power consumption due to leakage currents is also a significant contributor to total system power. However, researchers have shown that power-gating techniques can be comprehensively applied at the NoC level and are highly effective at reducing leakage power at periods of low network activity [32].

2.2.3 Spatial Locality and Cache Block Utilization

Spatial and temporal locality have been studied extensively since caches came into wide use in the early 1980's [33]. Several works in the 1990's and early 2000's focused on indirectly improving spatial locality through compile and run-time program and data transformations which improve the utilization of cache lines [16, 15, 20, 19]. While these techniques are promising, they either require compiler transformations or program changes and cannot be retrofitted onto existing code. Our proposed approach relies on low-overhead hardware mechanisms and is completely transparent to software.

Hardware techniques to minimize data transfer among caches and main memory have also been explored in the literature. Sectored caches were proposed to reduce the data transmitted for large cache block sizes while keeping overhead minimal [54]. With the sectored cache, only a portion of the block (a sector) is fetched, significantly reducing both the miss time and the bus traffic. The proposed technique builds upon this idea by speculatively fetching, not just the missing sector but sectors (words in this case) which have predicted spatial locality with the miss.

Prefetching is a technique where cache lines expected to be used in the future are fetched prior to their demand request, to improve performance by reducing misses. This may come at the cost of more power spent in the interconnect between caches when inaccurate prefetches lead to unused cache block fetches. Our technique is complementary and can be used to compensate prefetch energy overhead by gating unused words.

Pujara et al. examined the utilization of cache lines and showed that only 57% of words are actually used by the processor and the usage pattern is quite predictable [66]. They leverage this information to lower power in the cache itself by reducing the number of words read from the lower level cache and written to the upper level cache. This mechanism is orthogonal and potentially complementary to our technique, as we focus primarily on achieving lower energy consumption in the interconnect. Yoon et al. proposed an architecture that adaptively chooses memory system granularity based on spatial locality and error-tolerance tradeoffs [81]. While this work focuses on contention for off-chip memory bandwidth, our work targets on-chip interconnect energy consumption by observing spatial locality. Qureshi et al. suggested a method to pack the used words in a part of the cache after evicting it from the normal cache, thus increasing performance and reducing misses [67]. Their work thus focuses on performance rather than energy-efficiency and targets the effectiveness of the second-level cache.

Spatial locality prediction is similar to dead block prediction [50]. A dead block predictor predicts whether a cache block will be used again before it is evicted. The spatial locality predictor introduced in this work can be thought of as a similar device at a finer granularity. The spatial locality predictor, however, takes into account locality relative to the critical word offset, unlike dead block predictors. Chen et al. predicted a spatial pattern using a pattern history table which can be referenced by the pc appended with the critical offset [17]. The number of entries in the pattern history table, as well as the number of indexes increase the memory requirement of the technique. Unlike these schemes, our predictor uses a different mechanism for managing prediction thresholds in the face of mispredictions. Kim et al. proposed spatial locality speculation to reduce energy in the interconnect [47], we present here an extended journal version of this earlier work.

2.3 Description

Our goal is to save dynamic energy in the memory system interconnect by eliminating switching activity associated with unused words in cache blocks transferred between the different levels of the on-chip cache hierarchy. To this end we developed a simple, low complexity, spatial locality predictor, which identifies the words expected to be used in each cache block. A used word prediction is made on a L1 cache miss, before generating a request to the L2. This prediction is used to generate the response packet eliding the unused words with the proposed flit encoding schemes described below.

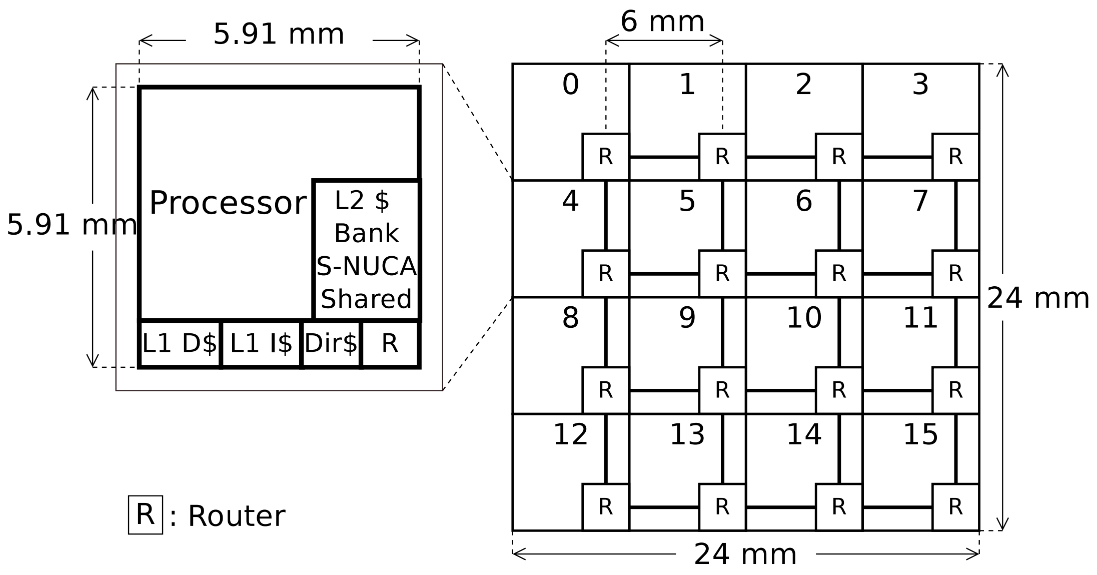


Figure 2.2: General CMP architecture

Figure 2.2 depicts the general, baseline architecture, representing a 16-node NoC-connected CMP. A tile consists of a processor, a portion of the cache hierarchy and a Network Interface Controller (NIC), and is bound to a router in the interconnection

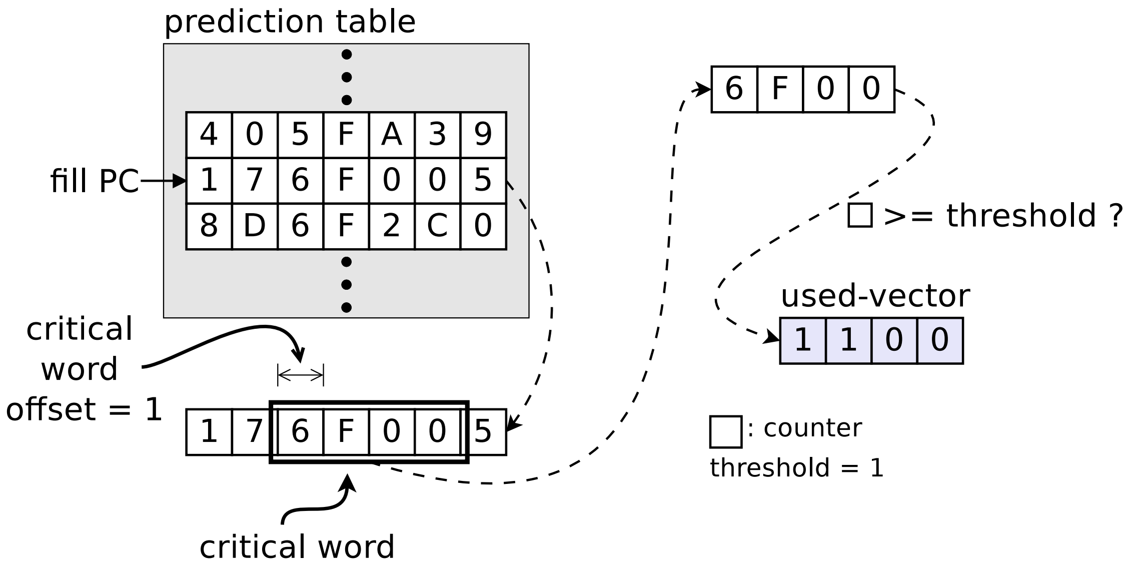


Figure 2.3: Prediction example for 4 words/block cache model

network. Each processor tile contains private L1 instruction and data caches. We assume the L2 is organized as a shared S-NUCA cache [46], each tile containing one bank of the L2. The chip integrates two memory controllers, accessed via the east port of node 7 and west port of node 8. Caches have a 64-byte block size. The NoC link width is 16 bytes, discounting flow-control overheads. Thus, cache-block-bearing packets are five flits long, with one header flit and four data flits. Each data flit contains four 32-bit words, as shown in Figure 2.5(b).

2.3.1 Spatial Locality Prediction

2.3.1.1 Prediction Overview

Our predictor leverages the history of use patterns within cache blocks brought by a certain instruction has been accessed. The intuition behind our predictor is that a given set of instructions may access multiple different memory address regions in a similar manner. In fact, we have observed that patterns of spatial locality are highly correlated to the address of the instruction responsible for filling the cache (the *fill PC*). The literature also shows that a small number of instructions cause the most cache misses [2]. Moreover, a given sequence of memory instructions accesses the same fields of data structures throughout memory [17]. Data structure instances, are unfortunately not aligned to the cache block, this misalignment can be adjusted by using the offset of the word which causes the cache miss (the *critical word offset*) while accessing the prediction table as in [66].

2.3.1.2 Predictor Implementation

Our prediction table is composed of rows of four-bit saturating counters where each counter corresponds to a word in the cache block. The table is accessed such that the *fill PC* picks the row of the table, and then n consecutive counters starting from the *critical word offset* are chosen where n is the number of words in a cache block. (Thus, there are $2n - 1$ counters per row to account for all possible $n - 1$ offsets.) These counters represent the history of word usage in cache blocks brought by a certain memory instruction.

The value of the saturating counter relates to the probability that the corresponding word is used. The lower the counter is, the higher confidence the word will not be used. Initially, all counters are set to their max value, representing a prediction where all words are used. As cache lines are evicted with unused words, counters associated with those unused words are decremented while counters associated with used words are incremented. If a given word counter is equal to or greater than a fixed *threshold* (configured at design time), then the word is predicted to be used; otherwise, it is predicted not used. We define *used-vector* as a bit vector which identifies the words predicted used by the predictor in the cache block to be filled. A used-vector of 0xFFFF represents a prediction that all sixteen words will be used while a used-vector of 0xFF00 signifies that only the first eight words will be used.

Figure 2.3 shows the steps to the prediction. In this example, the number of words in a block is assumed to be 4 and the threshold is 1, for simplicity. In the figure a cache miss occurs on an instruction accessing the second word in a given block (*critical word offset = 1*). The lower-order bits of the fill PC select a row in the prediction table. Among the counters in the row, the selection window of 4 counters, which initially includes the four rightmost counters, moves to the left by the number of the critical word offset. Those selected counters are translated into a predicted used-vector based on the threshold value. The used-vector, `1100`, indicates that the first and the second words in this block will be used.

The L1 cache keeps track of the actual used vector while the block is live, as well as the lower-order bits of the fill PC, and the critical word offset. When the block is evicted from the L1, the predictor is updated with the actual used vector; if a word was used, then the corresponding counter is incremented; otherwise, it is decremented. While updating, it finds the corresponding counters with the fill-PC and the critical word offset as it does for prediction. In the event a word is falsely predicted "unused", the counters for the entire row are reset to 0xF to reduce the likelihood of future mispredictions. This form of resetting have been shown to improve confidence over up/down counters for branch predictors [40]; in initial development of the predictor we found a considerable improvement in accuracy using this technique as well. Resetting the counters allows the predictor to quickly react in the event of destructive interference and/or new program phases.

2.3.1.3 Impact on Energy

We model a cache with 64B blocks and 4B words. Each row of the predictor is composed of 31 four-bit saturating counters where all counters are initialized to

0xF. The predictor table has 256 rows of 31×4 bits each, thus requiring ∼4KB of storage. We note, although the "word" size here is 4B, this represents the prediction granularity, it does not preclude use in a 64b (8B) word architecture.

In addition to the 4KB prediction table, our scheme requires extended metadata in each cache tag. In L1, 16 bits (one per word) are necessary to determine which words have been accessed so that we can update the predictor. 8 bits for *fill-PC* and 4 bits for *critical word offset* are required to access the prediction table, as well. We also replace the single valid bit with a vector of 16 bits in L1 and L2 caches. Although, this metadata increases the power per access of the L1 and L2 caches by 0.35% and 0.72%, respectively, we also reduce the number of words read from the lower level caches and written to the upper level cache by ∼30% and also the number of words written back into the lower level cache by ∼40%. Altogether this results an average ∼20% reduction in dynamic energy consumption per cache access. The dynamic energy consumed by the prediction tables is discussed in Section 4.6.

Although not the focus of this work, leakage energy dissipation can be also optimized with the help of spatial locality speculation. Chen et al. achieved 41% of leakage energy reduction with their proposed spatial locality predictor and a circuit level selective sub-blocking technique [17]. We expect a better reduction could be achieved with our technique (as our predictor accuracy is higher), we plan to explore this in a future work on used word prediction for cache power reduction.

2.3.1.4 Impact on Performance

When an L1 cache miss is discovered, the predictor supplies a prediction to inform flit composition. The prediction will take two cycles: one for the table access and one for thresholding and shifting. The fastest approach would be to speculatively assume that every cache access will result in a miss and begin the prediction simultaneously with address translation; thus, the latency can be completely hidden. A more energy efficient approach is to begin the prediction as soon as the tag mismatch is discovered and simultaneously with victim selection in the L1 cache. While this approach would add a cycle to the L1 miss time, no time would be added to the more performance critical L1 hit time. The latter approach was used in our experiments. If a word predicted unused actually is used, it is treated as a miss and all words initially predicted as unused are brought into the upper-level cache in order to correct this misprediction. This performance impact of these extra misses is discussed in Section 2.5.1.

On eviction of a L1 cache block, the used-vector and fill PC collected for that block are used to update the predictor. This process is less latency sensitive than prediction since the predictor does not need to be updated immediately to provide good accuracy.

2.3.2 Packet Composition

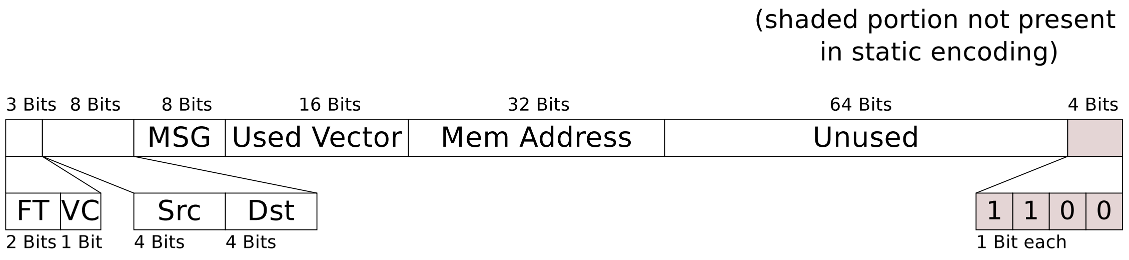
Once we have predicted the application's expected spatial locality to determine the unused words in a missing cache block, we employ a flit encoding technique which leverages unused words to reduce dynamic link and router energy in the interconnect between the L1, directory, L2 cache banks and memory controllers. We propose two complementary means to leverage spatial locality prediction to reduce α, the activity factor, in the NoC, thereby directly reducing dynamic energy: 1) Remove flits from NoC packets (*flit-drop*); 2) Keep unused interconnect wires at fixed polarity during packet traversal (*word-repeat*). For example, if two flits must be transmitted and all the words in the second flit are predicted unused, our *flit-drop* scheme would discard the unused flit to reduce the number of flits transmitted over the wire. In contrast, our *word-repeat* scheme would re-transmit the first flit, keeping the wires at fixed polarity to reduce gate switching. These encoding schemes are also used for writeback packets to include dirty words only.

The packet compositioning may be implemented either "statically", whereby packet encoding occurs at packet generation time, or "dynamically", in which the unused words in each flit are gated within the router FIFOs, crossbars and links to avoid causing bit transitions regardless of the flits which proceed or follow it. We will first discuss the "static" packet compositioning techniques including *flit-drop*, *static-word-repeat* and their combination. We then discuss the "dynamic" packet composition techniques which allow greater reductions in activity factor, at the cost of a small increase in logical complexity in the routers and a slight increase in link bit-width.

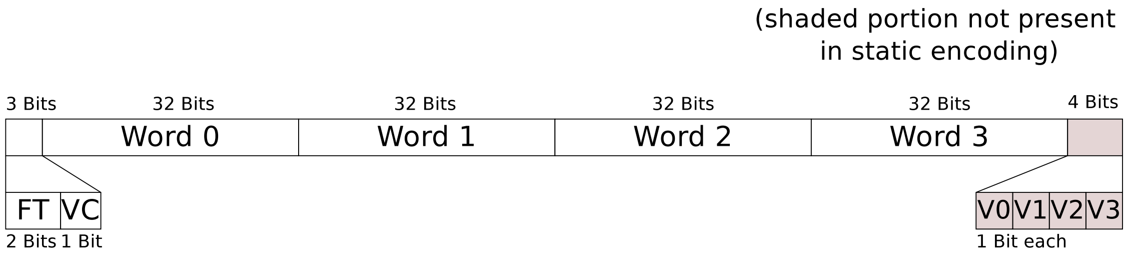
2.3.2.1 Static Packet Composition

Figure 2.4 depicts the format of cache request and reply packet flits in our design. A packet is composed either of a head flit and a number of body flits (when the packet contains a cache block) or it consists of one atomic flit, as in the case of a request packet or a coherence protocol message. The head/atomic flit contains a used-vector. The head flit also contains source and destination node identifiers, and the physical memory address of the cache block. The remaining bytes in the head/atomic flit are unused. We assume a flow-control overhead of three bits, 1 bit for virtual channel id (VC) and 2 bits for flit type (FT). As each of body/tail flit contains data of four words (16 bytes), a flit is 16 bytes and 3 bits wide including flow control overheads.

Figure 2.5(a) depicts an example of read request (L1 fill). In this example, tile #1 requests a block at address 0x00001200 which resides in the S-NUCA L2 cache bank in tile #8. The used-vector is `1111 1100 0000 1010`, indicating the words $word_0$ - $word_5$, $word_{12}$ and $word_{14}$ are predicted used. The corresponding response



(a) head/atomic



(b) body/tail

Figure 2.4: Flit format for static and dynamic encoding. (Shaded portion not present in static encoding.)

packet must contain at least those words. Since the baseline architecture sends the whole block as it is, the packet contains all of the words from $word_0$ to $word_{15}$, as shown in figure 2.5(b).

Flit-drop: In the *flit-drop* technique, flits which are predicted to contain only unused words are dropped from the packet and only those flits which contain one or more used words are transmitted. The reduction in the number of flits per packet, reduces the number of bit transitions over interconnect wires and therefore the energy consumed. Latency due to packet serialization and NoC bandwidth will also be reduced as well. Although a read request packet may have an arbitrary used-vector, the response packet must contain all flits which have any words predicted used leading to some lost opportunity for packets which have used and unused words intermingled throughout.

Figure 2.5(c) depicts the response packet to the request shown in Figure 2.5(a) for the *flit-drop* scheme. The first body flit, containing $word_0$ - $word_3$, therefore must be in the packet as all of these words are used. The second body flit, with $word_4$ - $word_7$, also contains all valid words, despite the prediction $word_6$ and $word_7$ would not be used. These extra words are overhead in the *flit-drop* scheme because they are not predicted used but must be sent nevertheless. Although these words waste dynamic power when the prediction is correct, they may reduce the miss-prediction probability.

Static-word-repeat: The *static-word-repeat* scheme, reduces the activity factor of flits containing unused words by repeating the contents in previous flit in the place of unused words. Flits with fewer used words consume less power because there are fewer bit transitions between flits. Words marked as "used" in the used-vector contain real, valid data. Words marked as "unused" in the used-vector contain repeats of the word in the same location in the previous flit. For instance, if $word_{4x+1}$ is predicted unused, the NIC places $word_{4(x-1)+1}$ in its place. As the bit-lines repeat the same bits, there are no transitions on those wires and no dynamic energy consumption. A buffer retaining four words previously fetched by the NIC is placed between the cache and the NIC and helps the NIC in repeating words. An extra mux and logic gates are also necessary in the NIC to encode repeated words.

Figure 2.5(d) depicts the response packet for the request in Figure 2.5(a) using the *static-word-repeat* scheme. In $body_1$, $word_6$ and $word_7$ are unused and, thus, replaced with $word_2$ and $word_3$ which are at the same location in the previous flit. All of the words in $body_2$ are repeated by the words in $body_1$, thus it carries virtually nothing but flow-control overhead. We also encode the unused header words, if possible.

FT	SRC	DST	MSG	Used Vector	Mem Address	Unused
ATOM	1	8	REQ	0xFC0A	0x1200	X

0XFC0A = 1111 1100 0000 1010

(a) Request

HEAD	8	1	RESP	0xFFFF	0x1200	UNUSED 0	UNUSED 1
BODY0	WORD 0				WORD 1	WORD 2	WORD 3
BODY1	WORD 4				WORD 5	WORD 6	WORD 7
BODY2	WORD 8				WORD 9	WORD 10	WORD 11
TAIL	WORD 12				WORD 13	WORD 14	WORD 15

0xFFFF = 1111 1111 1111 1111

(b) Baseline Response

HEAD	8	1	RESP	0xFF0F	0x1200	UNUSED 0	UNUSED 1
BODY0	WORD 0				WORD 1	WORD 2	WORD 3
BODY1	WORD 4				WORD 5	WORD 6	WORD 7
TAIL	WORD 12				WORD 13	WORD 14	WORD 15

0xFF0F = 1111 1111 0000 1111

(c) Flit-Drop Response

HEAD	8	1	RESP	0xFC0A	0x1200	WORD 2	WORD 3
BODY0	WORD 0				WORD 1	WORD 2	WORD 3
BODY1	WORD 4				WORD 5	WORD 2	WORD 3
BODY2	WORD 4				WORD 5	WORD 2	WORD 3
TAIL	WORD 12				WORD 5	WORD 14	WORD 3

0xFC0A = 1111 1100 0000 1010

(d) Word-Repeat Response

Figure 2.5: Read request and corresponding response packets (VC is not shown in this figure.)

2.3.2.2 Dynamic Packet Composition

The effectiveness of static packet compositioning schemes is reduced in two commonly-occurring scenarios: (a) when single-flit, atomic packets are being transmitted, and (b) when flits from multiple packets are interleaved in the channel. In both cases, repeated words in the flits cannot be statically leveraged to eliminate switching activity in the corresponding parts of the datapath. In response, we propose *dynamic packet composition* to reduce NoC switching activity by taking advantage of invalid words on a flit-by-flit basis. The difference between dynamic and static composition schemes resides primarily in how *word-repeat* treats unused words. In static composition, the unused portion of a flit is statically set at packet injection by the NIC to minimize inter-flit switching activity, requiring no changes to the router datapath. In dynamic composition, portions of the router datapath are dynamically enabled and disabled based on the validity of each word in the flit. In effect, an invalid word causes the corresponding portion of the datapath to hold its previous value, creating the illusion of word repeat.

To facilitate dynamic composition, the “used-vector” is distributed into each flit as shown in Figure 2.4(b). As a result the link width must be widened by four bits to accommodate the new “valid-word-vector”, where each bit indicates whether the corresponding word in that flit is valid. As the figure shows, the head flit’s “valid-word-vector” is always set to `1100` because the portion which corresponds to $Word_2$ and $Word_3$ of a body/tail flit are always unused.

Dynamic packet compositioning requires some modifications to a standard NoC router to enable datapath gating in response to per-flit valid bits. Figure 2.6 depicts the microarchitecture of our dynamic packet compositioning router. Assuming that a whole cycle is required for a flit to traverse a link, latches are required on both sides of each link. The additional logic required for link encoding is shaded in the magnified output port. Plain D-flip-flops are replaced with enable-D-flip-flops to force the repeat of the previous flit’s word when the “valid-word-vector” bit for that word is set to zero, indicating that word is not used. Alternately, if the “valid-word-vector” bit for the given word is one, the word is propagated onto the link in the following cycle, as it would in the traditional NoC router. In cases where the link traversal consumes less than a full cycle, this structure could be replaced with a tristate buffer to similar effect.

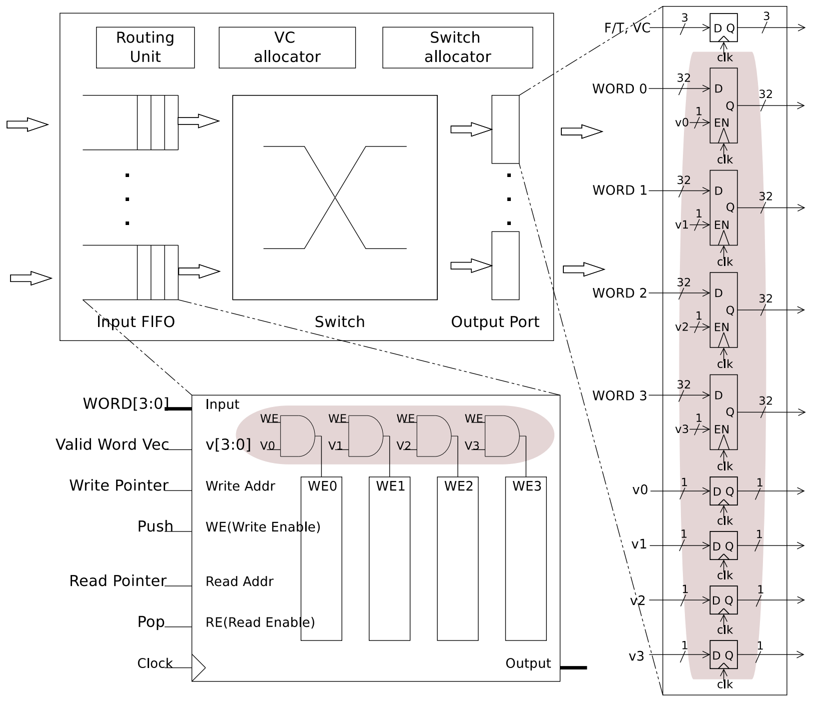


Figure 2.6: Dynamic packet compositioning router. (Shaded portion not present in baseline router.)

We further augment the router's input FIFO buffers with per-word write enables connected to the "valid-word-vector" as shown in Figure 2.6. In our design, the read and write pointer control logic in the router's input FIFOs remain unmodified; however, the SRAM array storage used to hold the flits is broken into four banks, each one word in width. The "valid-word-vector" bits would gate the valid write enables going to each of word-wide banks, disabling writes associated with unused words in incoming flits, and saving the energy associated with those word writes. The combination of these techniques for dynamic packet composition will reduce the power and energy consumption of the NoC links and router datapath proportional to the reduction in activity factor due to the *word-repeat* and *flit-drop* of unused words.

As *flit-drop* and *word-repeat* are complementary, we will also examine their combination in the evaluation section. One alternative technique we explored packs together used words into a minimal size packet. Experimentally we found this approach produces latency and power benefits negligibly different from the combination of *flit-drop* and *word-repeat*, while our technique requires less additional hardware in packet composition, so these results are not presented. These encoding schemes also are used for writebacks by marking clean words as unused.

2.4 Evaluation

2.4.1 Baseline Architecture and Physical Implementation

Figure 2.2 depicts the baseline architecture, representing a 16-node NoC-connected CMP. A tile consists of a processor, a portion of the cache hierarchy and a Network Interface Controller (NIC), and is bound to a router in the interconnection network. The baseline architecture employs a 4×4 2D mesh topology with X-Y routing and wormhole flow control. Each router contains 2 VCs and each input buffer is four flits deep. In our baseline configuration we assume the tiles are $36mm^2$ with 6mm-long links between nodes. Our target technology is 45 nm.

Processor Tiles: Each $36mm^2$ tile contains an in-order processor core similar to an Intel Atom Z510 ($26mm^2$) [38], a 512KB L2 cache slice ($4mm^2$), two 32KB L1 caches ($0.65mm^2$ each) and an interconnect router ($0.078mm^2$). The remaining area is devoted to a directory cache and a NIC. Our system is composed of 16 tiles and results in $576mm^2$, approximately the size of an IBM Power7 die [43]. We used CACTI 6.0 [59] to estimate cache parameters.

The L1 caches are two-way set-associative with a 2 cycle access latency. The L2 banks are 8-way set-associative with a 15-cycle access time. The 16 L2 banks spread across the chip comprise an 8-MB S-NUCA L2 [46]. Cache lines in both L1 and L2 caches are 64B wide (16 four-byte words), except where otherwise noted. Each node also contains a slice of the directory cache, interleaved the same as the L2. Its latency is 2 cycles. The number of entries in each directory cache is equal to the number of sets in an L2 bank. We assume the latency of the main memory is 100 cycles. The MESI protocol is used by the directory to maintain cache coherence. The predictor's performance is examined with the threshold value of 1 unless stated otherwise. The NoC link width is assumed to be 128 bits wide, discounting flow-control overheads.

NoC Link Wires: NoC links require repeaters to improve delay in the presence of the growing wire RC delays due to diminishing interconnect dimensions [39]. These repeaters are major sources of channel power and area overhead. Equally problematic is their disruptive effect on floorplanning, as large swaths of space must be allocated for each repeater stage. Our analysis shows that a single, energy-optimized 6 mm link in 45 nm technology requires 13 repeater stages and dissipates over 42 mW of power for 128 bits of data at 1 Ghz.

In this work, we consider both full-swing repeated interconnects (*full-swing links*) and an alternative design that lowers the voltage swing to reduce link power consumption (*low-swing links*). We adopt a scheme by Schinkel et al. [72] which uses a capacitative transmitter to lower the signal swing to 125 mV without the use of an additional low-voltage power supply. The scheme requires differential wires, doubling the NoC wire requirements. Our analysis shows a 3.5× energy reduction with low swing links. However, low-swing links are not as static-word-repeat friendly as much as full-swing links are. There is link energy dissipation on low-swing links, even when a bit repeats the bit ahead because of leakage currents and high sense amp power consumption on the receiver side. Thus, the repeated unused-words consume $\sim$ 18% of what used-words do. The dynamic encoding technique fully shuts down those portions of link by power gating all components with the "valid-word-vector" bits.

Router Implementation: We synthesized both the baseline router and our dynamic encoding router on a TSMC 45nm library to an operating frequency of 1Ghz. Table 2.1 shows the area and power of the different router designs. Note that the baseline router and the one used in static encoding scheme are identical. The table shows the average power consumed under PARSEC traffic, simulated with the methodology described in Section 2.4.2. The dynamic power for each benchmark is

Table 2.1: Area and Power

	baseline	static	dynamic
Area (mm^2)	0.073		0.078
Static Power (mW)	0.71		0.74
Router with full-swing link			
Dynamic Power (mW)	1.73	1.28	0.92
Total Power (mW)	2.45	1.99	1.67
Router with low-swing link			
Dynamic Power (mW)	0.62	0.46	0.33
Total Power (mW)	1.34	1.18	1.08

computed by dividing the total dynamic energy consumption by the execution time, then by the number of routers. Summarizing the data, a router design supporting the proposed dynamic composition technique requires ∼7% more area, while reducing dynamic power by 46% under the loads examined over the baseline at the cost of 4.3% more leakage power.

Table 2.2 shows the dynamic energy consumed by a flit with a given number of words encoded as used, traversing a router and a link, with respect to the three flit composition schemes: baseline(*base*), static(*sta*) and dynamic(*dyn*) encoding. In baseline, a flit always consumes energy as if it carries four used words. In static encoding, as the number of used words decreases, flits consume less energy on routers and full-swing links. Static-encoding reduces NoC energy by minimizing the number of transitions on the wires in the links and in the routers' crossbars. Dynamic-encoding further reduces router energy by gating flit buffer accesses. The four-bit, valid-word-vector in each flit controls the write enable signals of each word buffer, disabling writes associated with unused words. Similarly, it also gates low-swing links, shutting down the transceiver pair on wires associated with the unused words. CACTI 6.0 [59] was used to measure the energy consumption due to accessing the

Table 2.2: Per-Flit Dynamic Energy (pJ)

	Router			Full Swing Link			Low Swing Link		
n	base	sta	dyn	base	sta	dyn	base	sta	dyn
0		0.73	0.34		0.99	2.30		0.35	0.66
1		1.31	1.01		11.52	12.83		3.34	3.67
2	3.58	1.90	2.01	43.10	22.04	23.36	12.31	6.33	6.67
3		2.77	2.79		32.57	33.89		9.32	9.68
4		3.58	3.65		43.10	44.41		12.31	12.69

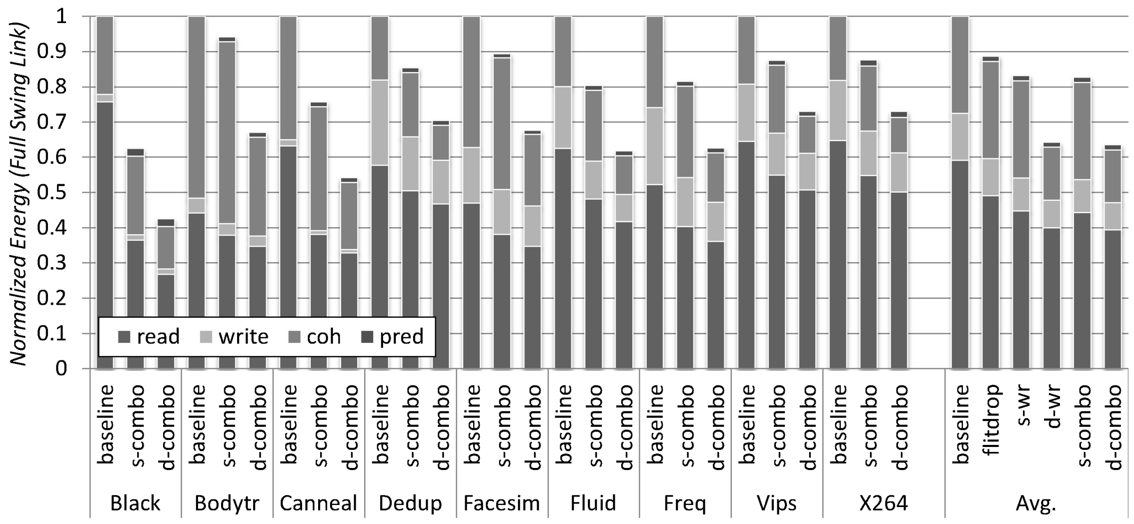
n: number of used words

predictor; which is 10.9 pJ per access.

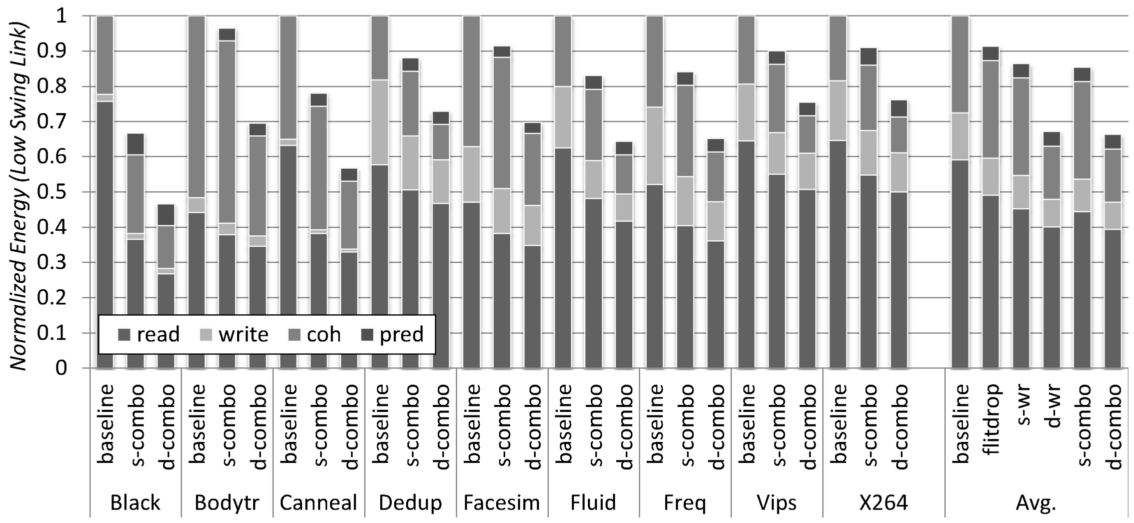
2.4.2 Simulation Methodology

We used the M5 full system simulator to generate CMP cache block utilization traces for multi-threaded applications [12]. Details of the system configuration are presented in section 2.4.1. Our workload consists of the PARSEC shared-memory multi-processor benchmarks [10], cross-compiled using the methodology described by Gebhart et. al [27]. All applications in the suite currently supported by M5 were used. Traces were taken from the "region of interest." Each trace contains up to a billion memory operations; fewer if the end of the application was reached. Cycle accurate timing estimation was performed using the *Netrace*, memory system dependence tracking methodology [34].

The total network energy consumption for each benchmark is measured by summing the energy of all L1 and L2 cache fill and spill and coherence packets as they traverse routers and links in the network. In effect, Table 2.2 is consulted whenever a flit with a certain number of used words traverses a router and a link. Note that even for the same flit, the used word number may vary according to the encoding scheme in use. For example, for an atomic flit, $n = 4$ in static encoding while $n = 2$ in dynamic. The predictor's energy is also added whenever the predictor is accessed.



(a) full-signal swing link



(b) low-signal swing link

Figure 2.7: Dynamic energy breakdown

2.4.3 Energy Consumption

Figure 2.7 shows the breakdown of dynamic energy consumption. For each benchmark, we conducted energy simulations for three configurations, each represented by one stacked bar for that benchmark: 1) *baseline* - baseline, 2) *s-combo* - static-word-repeat and flit-drop combined, and 3) *d-combo* - dynamic-word-repeat and flit-drop combined. We also show the *average* energy consumption with pure flit-drop (*flit-drop*), static-word-repeat (*s-wr*) and dynamic-word-repeat (*d-wr*). The bars are normalized against the energy consumed by baseline. Each bar is subdivided into up to four components. The first bar shows the "read" energy, energy consumed by cache fills and the second bar, "write", by writebacks. The third bar, "coh", shows the energy due to the cache coherence packets and the fourth bar, "pred" shows the energy consumed by predictors. The figure shows data for both full-swing and low-swing links.

In the *baseline* configuration we see that, on average, read communication consumes the most dynamic energy with ∼59% of the total. Coherence traffic consumes the second most with ∼28% of the total energy followed by write communication with ∼13% of the total energy. This breakdown follows the intuition that reads are more frequent than writes. Thus, techniques which only focus on writebacks will miss much potential gain. It is also interesting to note that cache coherence traffic shows a very significant contribution to overall cache interconnect energy. Similarly, work which does not consider atomic packets may miss significant gain.

The figure also shows that among the flit encoding schemes, *d-combo* shows the greatest improvement with ∼36% dynamic energy savings on average when full-signal swing link is used. If low-signal swing links are used, it becomes ∼34%. The pure dynamic-word-repeat (*d-wr*) is the second best resulting in additional ∼1% energy consumption. This implies that dropping flits only with flow control bits does not significantly contribute to energy reduction when dynamic- word-repeat is used. However, combining flit-drop is still beneficial to reduce latency. The combined static encoding (*s-combo*) provides an energy savings of only ∼17% and ∼15% of baseline, under full-swing and low-swing links, respectively. This indicates the

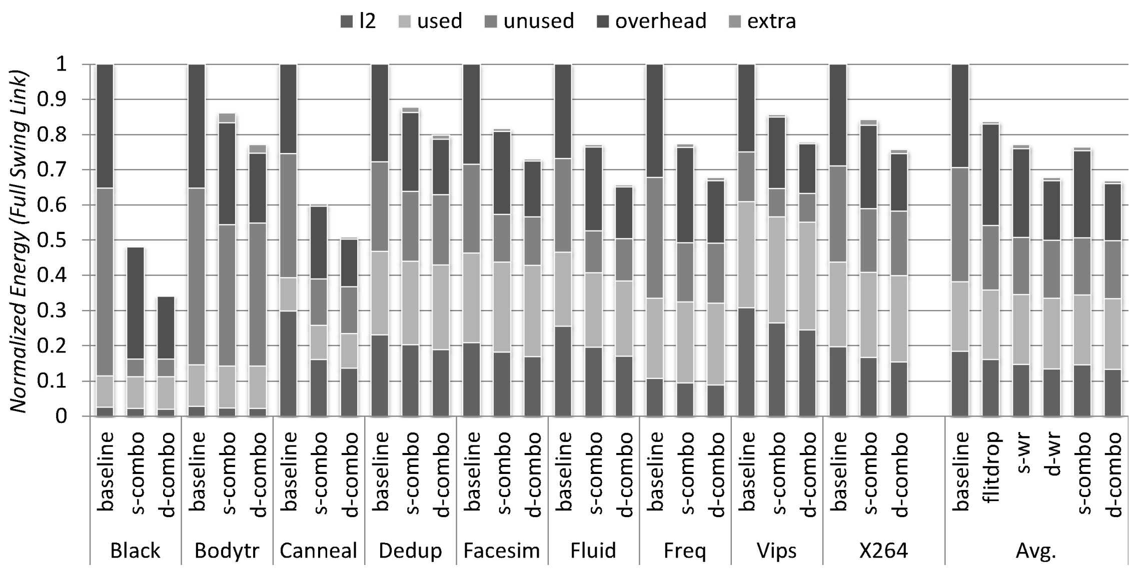


Figure 2.8: Dynamic energy breakdown for reads

significant gains that dynamic encoding provides, primarily in the cache coherence traffic which is predominately made up of single flit packets. We find the predictor merely contributes 1.5% of the total energy when full-signal swing link is used, and 4.1% when low-signal swing link is used.

Table 2.1 shows the average power with either type of links. It reveals that despite the increased static power, the dynamic encoding scheme still outperforms the baseline and the static encoding as well, regardless of link type.

In the following sections we will examine each of the traffic types in detail to gain a deeper understanding of the performance of our proposed technique. As full-swing link and low-swing link show similar trends, only graphs for full-swing links will be shown hereafter.

2.4.3.1 Read Energy Discussion

Figure 2.8 shows the breakdown of dynamic energy consumption for reads. Each bar is subdivided into five components and also normalized against baseline. The first bar "l2" depicts the energy consumed by L2 cache fills and spills. Although the prediction actually occurs on L1 cache misses, the encoding schemes are also used for the transactions between the L2 and memory controller, based upon used-vector generated on the L1 cache miss the lead to the L2 miss.

The second bar shows the "used" energy, energy consumed by the words which will be referenced by the program, hence "used" bars are nearly equal, with the exception of a slight increase in energy due to router overheads in the dynamic scheme. The third bar, "unused", shows the energy consumed to bring in words which will not be referenced prior to eviction. This also includes the energy consumed by words which result from *false-positive* predictions, i.e. an incorrect prediction that the word will be used. The fourth bar, "overhead", shows the energy for NoC packet overheads, including header information and flow control bits. The fifth bar, "extra", shows the energy consumed by the packet overhead due extra cache line fills to correct "false-negative" mispredictions. Our goal is to remove, as much as possible, the dynamic datapath energy consumed by unused words denoted by *unused* and, where possible, the packet overheads in *overhead*, while minimizing redundant misses due to mispredictions in *extra*. Unused words consume an average of 33% of total dynamic datapath energy, and up to 53% of total dynamic datapath energy in case of *blackscholes* (shown as *Black* in the graphs.)

The *d-combo* scheme, on average, reduces total dynamic datapath energy by ~32%. Our prediction mechanism combined with the encoding schemes approximately halves the "unused" portion, on average. In case of *Black* where the predictor performs the best, the speculation mechanism removes 90% of the "unused" portion resulting in a 66% energy savings for cache fills when combined dynamic encoding is used. The extra transmission due to mispredictions, shown as "extra" in the stack, contributes less than 1% of energy consumption for cache fills.

2.4.3.2 Coherence Energy Discussion

In the simulated system coherence is maintained via the MESI protocol. Coherence protocol messages and responses represent a significant fraction of the network traffic. Those protocol messages are composed primarily of single-flit packets, and contribute ∼28% of total network energy consumption. Figure 2.9 shows the breakdown of dynamic energy consumption for coherence packets. Although these single-flit packets contain ∼50% unused data, as discussed in Section 2.3.2.1, static-encoding can not be used to reduce their energy dissipation. Dynamic-encoding, however, reduce it by up to 45.5%.

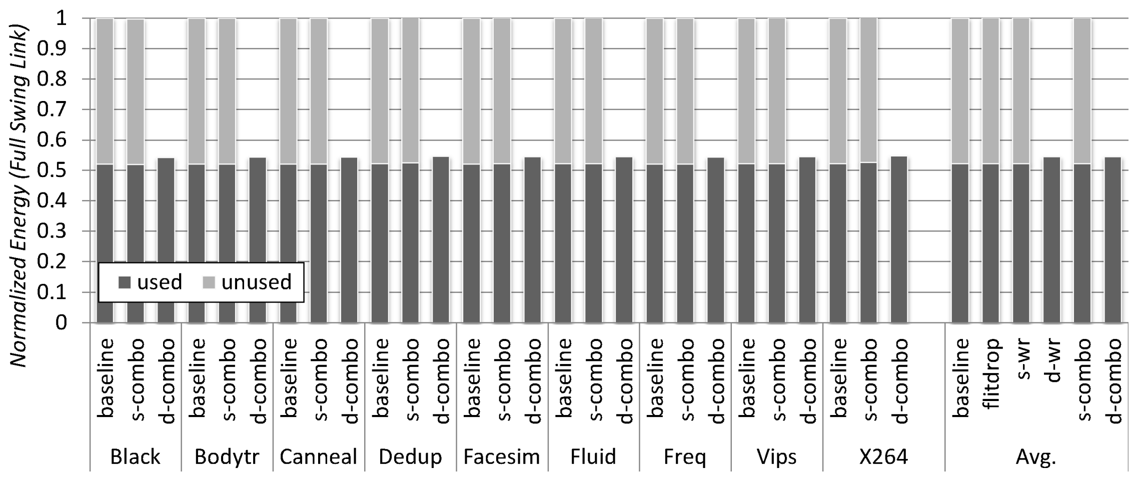


Figure 2.9: Dynamic energy breakdown for coherent packets

2.4.3.3 Write Energy Discussion

Figure 2.7 shows that writebacks consume an average of 13% of total dynamic energy. Upon dirty line writeback, we find that, on average, 40% of the words in the block are clean, and those words contribute 23% of the total energy consumed by writebacks.

Figure 2.10 shows the dynamic energy breakdown caused by writebacks. The first bar, “dirty”, shows the energy consumed by the dirty words in cache lines. The second bar “overhead” shows the energy consumed by NoC packet overheads. The third bar, “clean”, includes the link energy consumed by sending clean words in writeback packets. Our goal is to remove the portion of the energy consumption associated with transmitting “clean” words. On average, the *s-combo* scheme reduces the energy consumption due to writebacks by 29%. Further savings are achieved by *d-combo*. It encodes not only body/tail flits but also head flits of the writeback packets resulting in a 40% savings. When full swing links are used, it is possible to remove all of energy dissipation due to clean words with the static flit-encoding scheme. However, when static word repeat is used with low swing links, although clean words are encoded to repeat the words in the flit ahead, those words cause energy dissipation due to leakage currents and high sense amp power consumption on the receiver side.

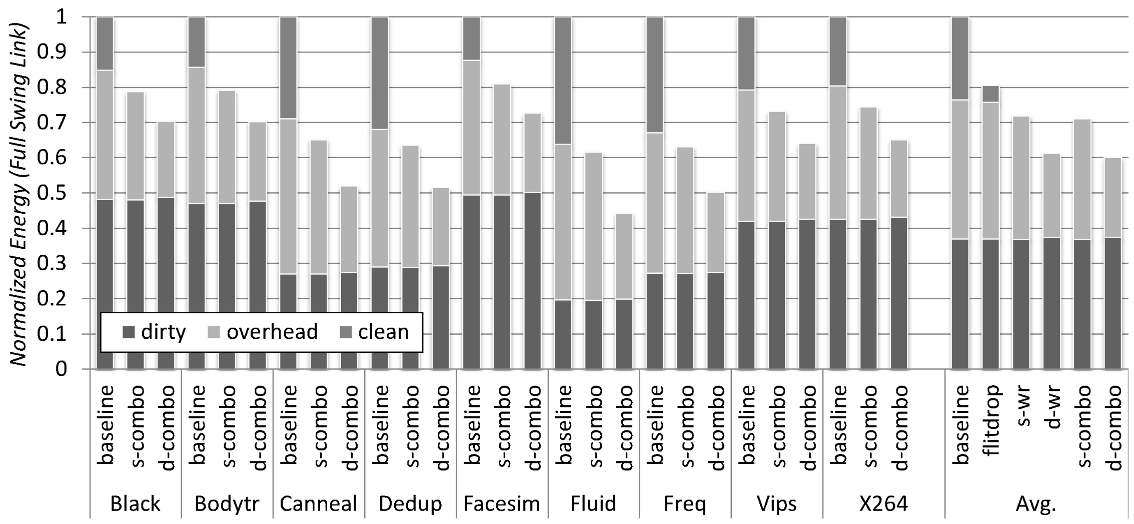


Figure 2.10: Dynamic energy breakdown for writes

2.5 Analysis

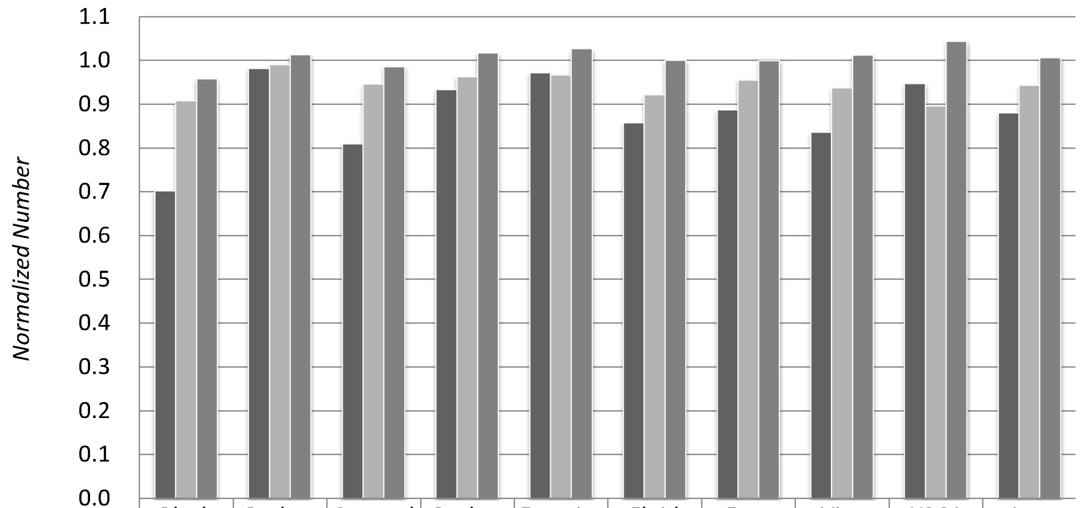
In this section we analyze the performance impact of the proposed energy reduction technique, explore predictor training and compare against a 32-byte cache line baseline design.

2.5.1 Performance

The performance impact of the proposed technique is governed by two divergent effects. First, the scheme should improve performance because flit-drop reduces the number of flits injected into the network. Decreased flit count improves performance through less serialization latency, and reduced congestion due to lower load. Second, offsetting the benefit from flit-drop, incorrectly predicting a word unused can lead to more misses, increasing network load and average memory access time (AMAT). To quantify the impact of the proposed technique, Figure 2.11 shows the reduction in flits injected into the network for each benchmark, the reduction in individual packet latency, and the AMAT for each benchmark, all normalized against baseline. Each value number is normalized against baseline. As the figure shows, although flit count and individual packet latency decrease significantly, AMAT is essentially flat across the benchmarks. In this section we examine the relationship between network performance and system performance.

2.5.1.1 Network Performance

As shown in Figure 2.11, the flit count is reduced by 12% on average. Optimally, the flit count reduction should be directly proportional to the block utilization. From Figures 2.1, 2.7 and 2.11, we see that Blackscholes, which has the lowest block utilization, and the greatest portion of read energy consumption, has the greatest reduction flits across the PARSEC benchmarks. Alternately, Bodytrack, which also



	Black	Bodytr	Canneal	Dedup	Facesim	Fluid	Freq	Vips	X264	Avg.
■ Flit Count	0.702	0.982	0.810	0.933	0.972	0.858	0.887	0.836	0.947	0.881
■ Packet Latency	0.908	0.991	0.946	0.963	0.967	0.922	0.955	0.938	0.896	0.943
■ AMAT	0.958	1.013	0.986	1.018	1.026	1.000	1.000	1.012	1.044	1.006

Figure 2.11: Flit count, packet latency and AMAT normalized against baseline.

shows one of the lowest block utilizations, removes merely 1.8% the injected flits. This is because flit count reduction is related not only to block utilization, but also prediction accuracy, proportion of single flit packets, and the used-unused pattern within the packet. In Bodytrack, flit reduction is low because single-flit coherent packets make up a larger portion of the injected packets, and the predictor is less accurate than for Blackscholes.

Lowered flit count should be correlated with reduced packet latency. Figure 2.11 shows normalized packet latency. On average, the network latency is reduced by ∼6% as the number of flits has decreased. In Blackscholes, with greatest reduction in flit count, the packet latency is reduced by 9%, showing one of the best network performance improvements across the PARSEC benchmarks. Alternately, Bodytrack's network performance is improved by only 1%. Interestingly, flit count reduction and

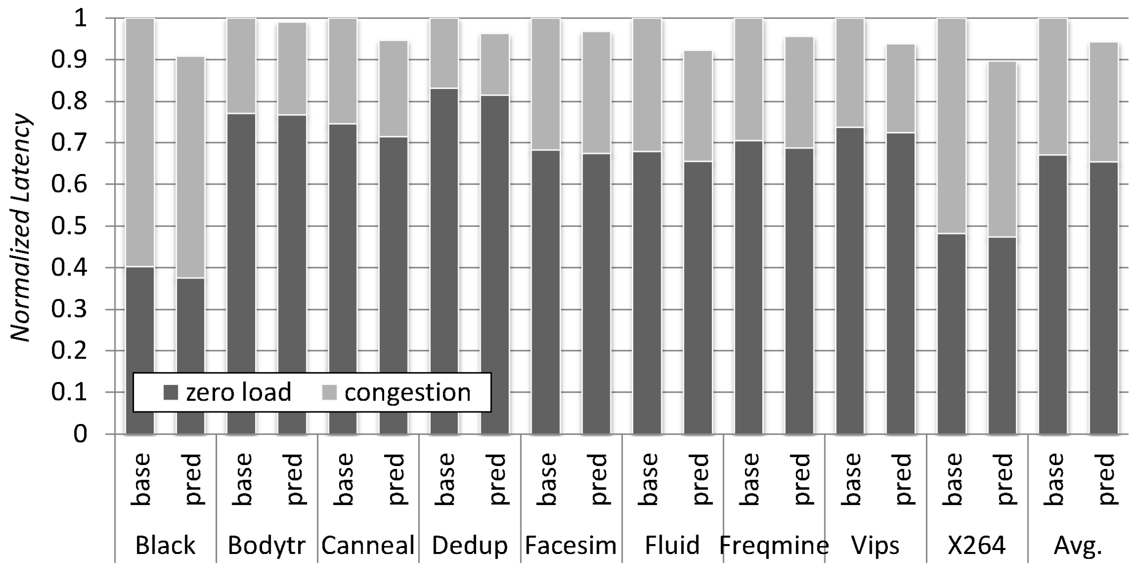


Figure 2.12: Packet latency breakdown

network latency are not always strictly proportional to each other; X264, counter-intuitively shows a greater improvement in packet latency than its reduction in flit count, warranting further analysis.

Flit-drop improves network performance not only by reducing the serialization latency but also by avoiding network congestion. Figure 2.12 shows the breakdown of average packet latency. Each bar consists of two components; 1) *zero load* shows the packet latency due to static hop count and serialization latencies, 2) *congestion* shows the latencies due to the resource conflicts. Although each component contributes 67% and 33% of the average latency, respectively, the greater impact of reduced flit count lies in *congestion*. This effect is illustrated by X264 which has the second greatest congestion latency, as a result a relatively small reduction in flits translates into a greater reduction in packet latency. The overall average packet latency has been reduced by 5.7%.

2.5.1.2 Overall System Performance Discussion

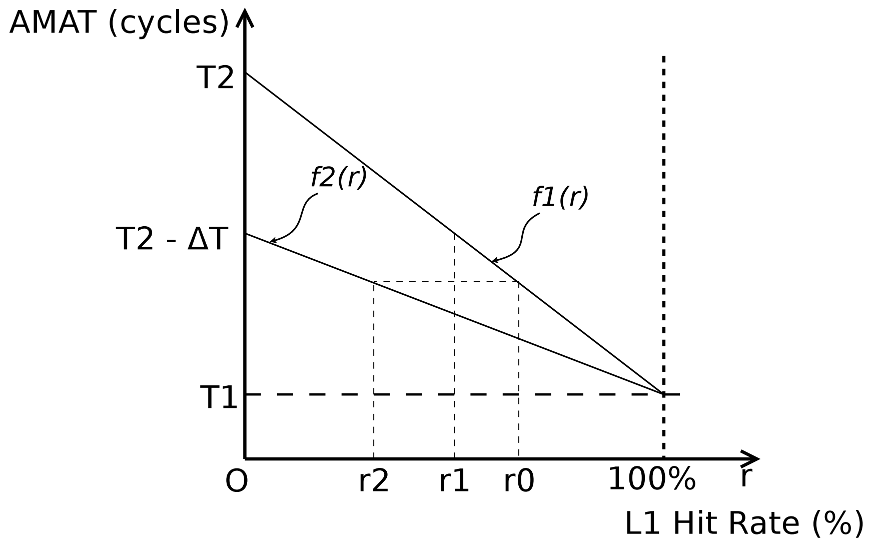


Figure 2.13: AMAT graph

As a proxy for overall system performance we examine the technique's impact on average memory access time (AMAT). Despite an improvement in packet latency, Figure 2.11 shows that AMAT is unchanged on average, with some benchmarks showing a slight improvement, while others showing a slight degradation. To explore this counter-intuitive result we examine how AMAT relates to packet latency and L1 miss rate. AMAT in this work and it can be estimated by Equation 2.2.

$$AMAT = Latency(L1) + (1 - HitRate(L1)) \times Latency(L2+) \tag{2.2}$$

In this equation, $Latency(L1)$ is the constant L1 latency for the system, and $HitRate(L1)$ is the hit rate of L1 accesses which varies by benchmark locality and is effected by false unused-word predictions. $Latency(L2+)$ is the latency of memory accesses served by L2 and beyond, which is also known as L1 *miss* latency. It is a function of the constant L2 cache access time, L2 miss rate, network latency and the constant memory access time. Assuming $Latency(L2+)$ is fixed, AMAT is a linear function of $HitRate(L1)$. Figure 2.13 visualizes AMAT as $f(r)$ where r is L1 hit rate.

In Figure 2.13, let $f_1(r)$ be the AMAT characteristic for a benchmark, where T_1 is the L1 latency and T_2 is L1 latency plus L1 miss latency. Say, with the baseline scheme, the L1 hit rate is r_0 and AMAT becomes $f_1(r_0)$. If our prediction mechanism drops the L1 hit rate to r_1 and that does not change L1 miss latency, the AMAT becomes $f_1(r_1)$. In such a case, $f_1(r_1) - f_1(r_0)$ represents the performance loss due to mispredictions. However, thanks to our packet composition technique, L1 miss latency, in general, is lower than that of baseline cases. Thus, its AMAT characteristic function should be redrawn as $f_2(r)$ and the AMAT at r_1 is $f_2(r_1)$. If $f_2(r_1) < f_1(r_0)$ as in this example, the difference, $f_1(r_0) - f_2(r_1)$ denotes the performance benefit from our prediction technique.

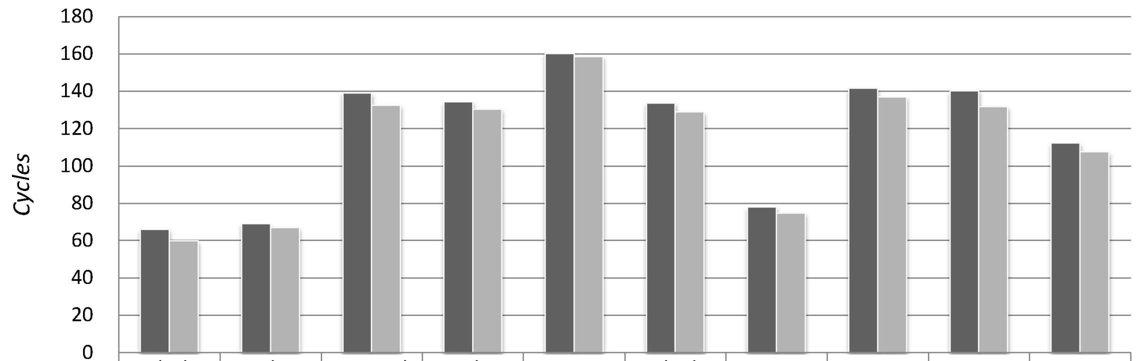
In this example, we can also see that as long as the predictor drops the L1 hit rate no lower than r_2, performance improvement is expected. We define *safe range* as the range of L1 hit rate where our prediction scheme shows equal or better AMAT than the baseline design. In this particular example, the safe range is $[r_2, r_0]$. To generalize, safe range, Δr, is calculated as below:

$$\Delta r = \frac{\Delta T}{T_o - \Delta T}(1 - r_o) \tag{2.3}$$

where T_o and r_o are the original L1 miss latency and L1 hit rate, respectively, and ΔT the reduced amount of L1 miss latency. The wider safe range we have, the better chance that we achieve the performance improvement.

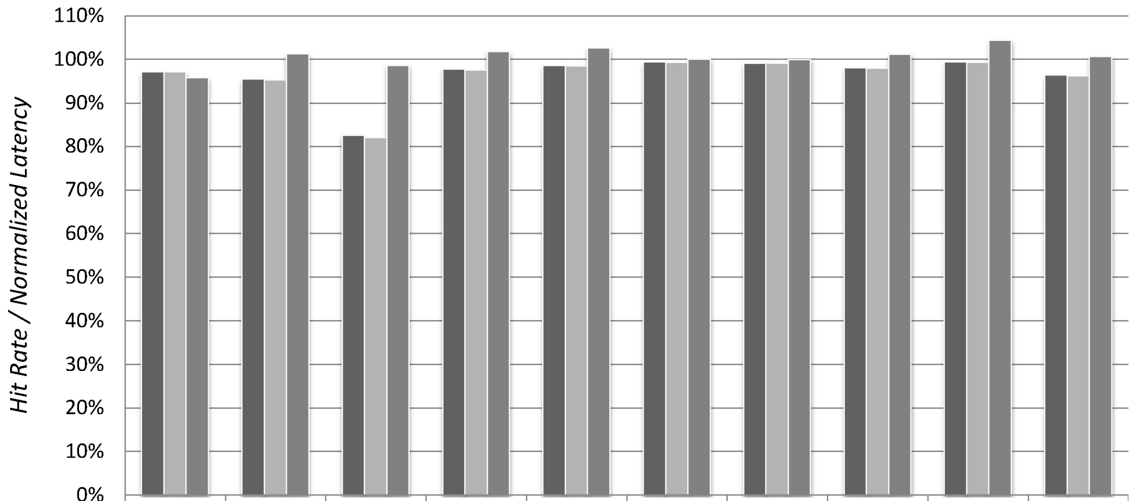
Figure 2.14(a) shows the average L1 miss latency for each benchmark. The bars marked as *base* show the L1 miss latency for the baseline design, while *pred* shows the latency for the proposed scheme. In every case, a reduction in L1 miss latency is observed. This reduction is closely related to the reduction in the interconnect network latency shown in Figure 2.12. Although the packet latency in the figure is normalized, the L1 miss latency shows the similar trend to that; the more reduction in the network latency we have, the more reduction in L1 miss latency. According to Equation 2.3, the wider safe range, and, in turn, no performance degradation, is expected for benchmarks with a lower *base* and a bigger gap between *base* and *pred*. However, the safe range is still up to L1 hit rate.

Figure 2.14(b) shows the original L1 hit rate (*"l1 hit base"*) the new L1 hit rate (*"l1 hit pred"*) and the changed average memory access time (*"norm lat"*). On average, L1 hit rate is decreased by 0.15%. Blackscholes ("Black") has the third greatest improvement in L1 miss latency, the lowest L1 miss latency and the second lowest L1 hit rate, it results in the greatest overall performance improvement of 4.2%. By contrast, "X264", while having the greatest L1 miss latency improvement, also has one of the highest L1 miss latency, and the highest L1 hit rate, therefore achieves the worst overall system performance. Although "Canneal" shows the worst impact on L1 hit rate, due to its low original L1 hit rate, the reduced L1 hit rate is still in its safe range, thus, no performance penalty for mispredictions is shown.



	Black	Bodytr	Canneal	Dedup	Facesim	Fluid	Freqmine	Vips	X264	geo mean
■ base	65.97	68.97	139.01	134.24	160.14	133.53	77.96	141.53	140.29	112.17
■ pred	59.94	67.04	132.53	130.34	158.64	128.92	74.60	136.95	131.73	107.46

(a) L1 Miss Latency



	Blacks	Bodytr	Canneal	Dedup	Facesim	Fluid	Freqmine	Vips	X264	Avg.
■ l1 hit base	97.12%	95.47%	82.63%	97.75%	98.60%	99.39%	99.15%	98.12%	99.46%	96.41%
■ l1 hit pred	97.10%	95.24%	82.06%	97.61%	98.52%	99.37%	99.11%	98.02%	99.33%	96.26%
■ norm lat	0.958	1.013	0.986	1.018	1.026	1.000	1.000	1.012	1.044	1.006

(b) L1 Hit Rate

Figure 2.14: Overall system performance

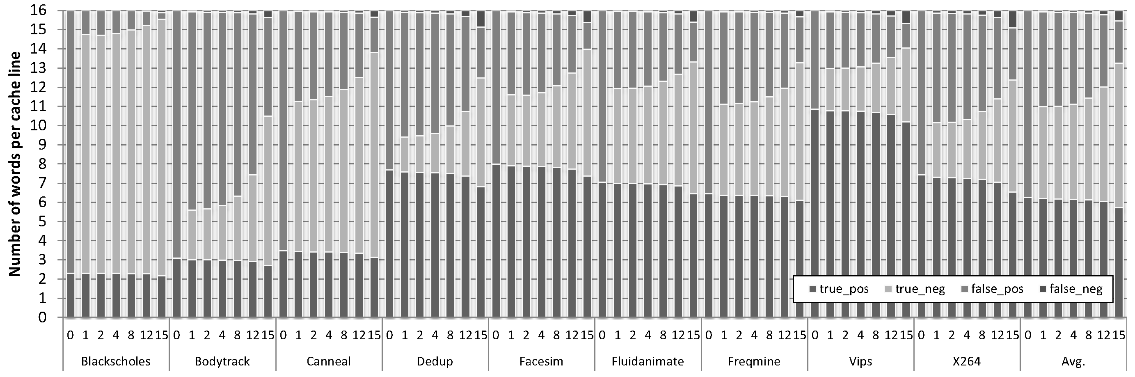


Figure 2.15: Breakdown of predictions outcomes

2.5.2 Predictor Tuning

As with many speculative techniques, our scheme incurs a performance penalty when mis-speculation occurs. In this case, the penalty manifests as increased L1D misses. As Figure 2.14(b) shows, L1D hit rates are barely impacted by our technique. One possible interpretation of this data is that our predictor is overly conservative, and that energy gain could be achieved by more aggressively tuning our predictor, in this section we explore predictor tuning to this end.

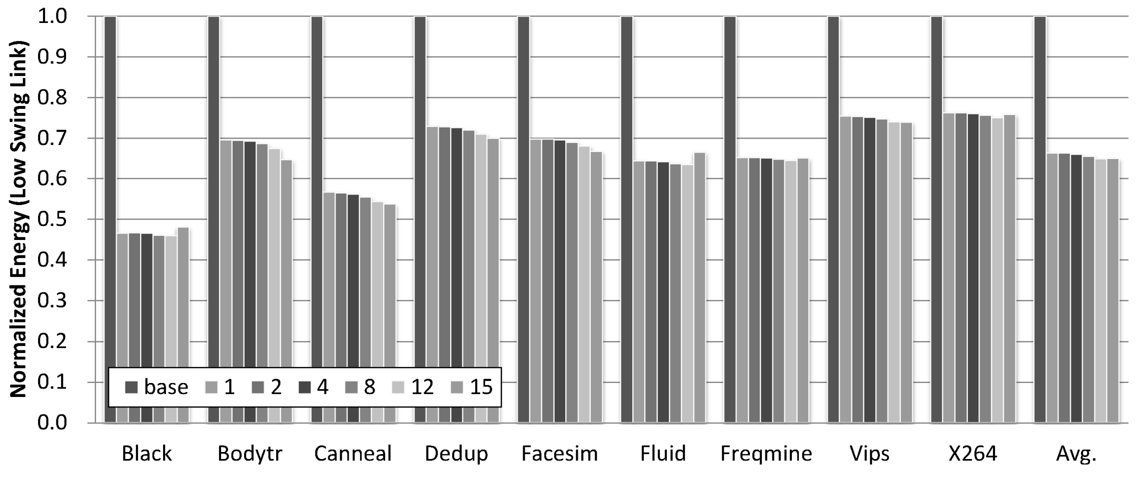
Our prediction model requires a threshold value to be configured at design time. As described in Section 2.3.1.1, the threshold determines whether a certain word will be used or not according to its usage history counter. If the counter value is less than the threshold, the word is predicted to be unused. The smaller threshold value, the more biased the predictor towards predicting a word will be used. Thus, the threshold value tunes the trade-off between energy consumption and memory access time.

Figure 2.15 shows the prediction outcomes with respect to various threshold values (numbers along the bottom) for each of the benchmarks examined. Each bar is broken into components to show average number of words in a cache line with the following characteristics. The bars marked *true_pos* show the fraction of true positives: words predicted used and actually used. The bars marked *true_neg* show the portion of true negatives: words predicted unused and actually not used. The words in this category are the source of the energy reduction in our design. These two categories form the portion of the words that the predictor correctly speculates their spatial localities. The bars marked *false_pos* show the fraction of false positives: words predicted used but actually unused. The words in this category do not cause any miss penalty but represent a lost opportunity for energy reduction. Finally, the bars marked *false_neg* show the portion of false negatives: words predicted unused but actually used. These words result in additional memory system packets, potentially increasing both energy and latency. The threshold value "0" in this figure represents the baseline configuration, where all words are assumed used. In general, as the threshold value increases, the portions of *true_neg* and *false_neg* increase while *true_pos* and *false_pos* decrease. This implies that the higher threshold chosen, the lower energy consumption (due to *true_neg* predictions) but also the higher the latency(due to *false_neg* predictions). We also note that even with the most aggressive threshold setting, a significant number of *false_pos* predictions remain, despite significant increases in *false_neg* predictions, implying that headroom for improvement via a more accurate prediction mechanism exists.

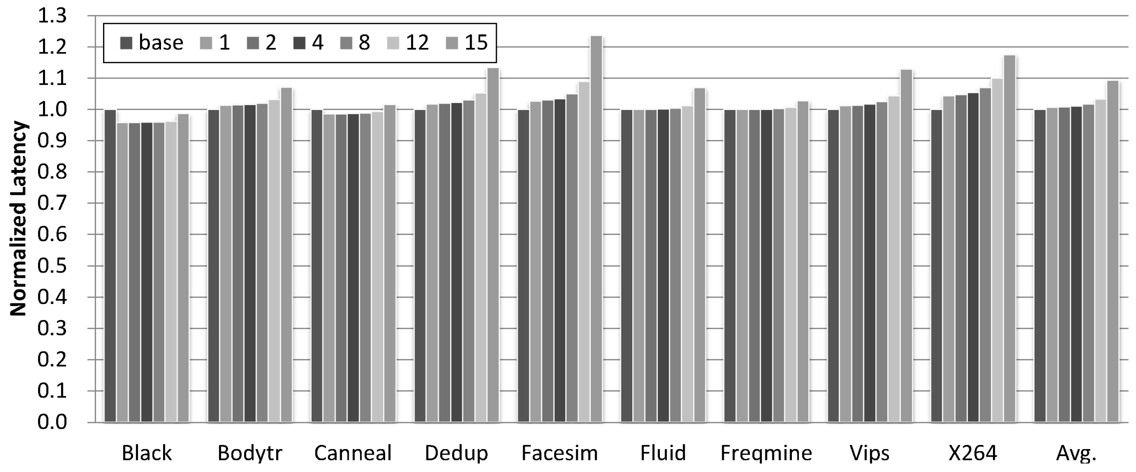
Figure 2.16 depicts the normalized energy consumption and normalized average memory access time (AMAT) for threshold values from 1 to 15. For this experiment, we use low swing links, a similar trend of energy consumption and AMAT was found for full swing links. Figure 2.16(a) shows a modest downward trend in energy consumption as the threshold value increases, with the greatest increase between thresholds of 8 and 12. This is the expected outcome of growing *true_neg* with higher threshold values in Figure 2.15. In some benchmarks, such as “Black”, “Fluid” and “X264”, there is slight increase at the highest threshold value. The main reason for this increase is the energy required to service increased L1D misses, which overcome the benefit of transmitting fewer words in the initial request.

Figure 2.16(b) shows the normalized AMAT with varying threshold. In general, the latency shows a modest upward trend as the threshold grows. The higher the threshold, the more words are speculated as unused by the predictor, leading to increased L1 miss rates and degrading the overall memory system latency. Though this trend becomes dramatic for a threshold of 15, increasing AMAT by up to 23% for one application; we find that thresholds of less than 4 have a minimal negative impact on AMAT.

Given our goal was to decrease energy with a minimal impact on performance, we use the $Energy \times Delay^2$ metric as a figure of merit for our design. Experimentally we determined that $Energy \times Delay^2$ is approximately equal across the thresholds between 1 and 8, however, it considerably increases beyond the threshold 12. This result validates our choice a threshold value of 1 in our experiments. We find the performance impact with this bias is negligible. On average, with this threshold, the additional latency of each operation is $\sim$0.6%. These results show that further energy savings could be achieved through improved predictor accuracy, which we leave to future work.



(a) Normalized Energy Consumption



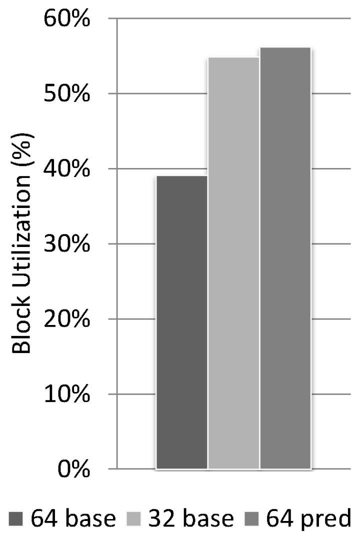
(b) Normalized Average Memory Access Time (AMAT)

Figure 2.16: Normalized energy and AMAT for different threshold values

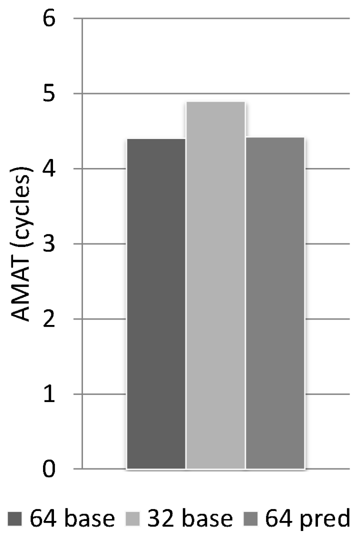
2.5.3 Case Study: Comparison with Smaller Lines

Naïvely, one might conclude that the low cache block utilization shown in Figure 2.1 could be an indication that cache line size is in-fact too long, and that utilization could be improved by implementing a smaller cache line size. To explore this concern we examine our technique versus a 32-byte cache lines baseline.

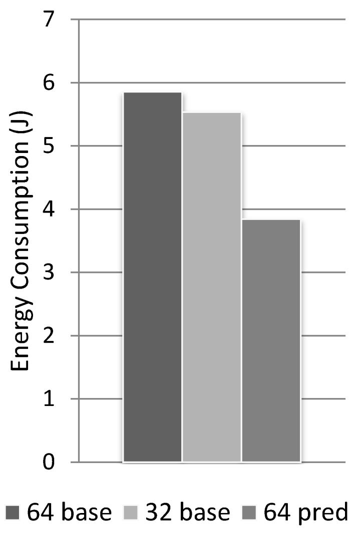
Figure 2.17 shows results for three different configurations; 1) the baseline design with 64-byte cache lines (*64 base*), 2) a baseline design with 32-byte lines (*32 base*) keeping the cache size and associativity the same as *64 base*, and 3) 64-byte lines with our prediction and dynamic packet composition technique (*64 pred*). Figure 2.17(a) shows the arithmetic average of block utilization for each configuration across the PARSEC benchmarks. Figure 2.17(b) shows the geometric mean of AMAT and Figure 2.17(c) depicts the geometric mean of total energy consumption. These figures show, 32-byte lines have better utilization than the 64-byte baseline. Compared with *64 base*, however, only a marginal energy reduction is achieved at the cost of considerable performance loss. The smaller cache block size results in the increased L1 misses, and thereby, increased latency of memory accessing operations. *64 pred*, shows even greater block utilization than *32 base* while maintaining the performance of *64 pred* and consuming much less energy than the rest. Hence, the proposed technique is a better design choice than shrinking cache lines to reduce power.



(a) Block Utilization



(b) AMAT



(c) Energy

Figure 2.17: Comparison to a smaller cache line

2.6 Conclusions

In this work, we introduce a simple, yet powerful mechanism using spatial locality speculation to identify unused cache block words. We also propose a set of static and dynamic methods of packet composition, leveraging spatial locality speculation to reduce energy consumption in CMP interconnect. These techniques combine to reduce the dynamic energy of the NoC datapath through a reduction in the number of bit transitions, reducing α the activity factor of the network.

Our results show that with only simple static packet encoding, requiring no change to typical NoC routers and very little overhead in the cache hierarchy, we achieve an average of 17% reduction in the dynamic energy of the network if full-signal swing links are used. Our dynamic compositioning technique, requiring a small amount of logic overhead in the routers, enables deeper energy savings of 36% and 34%, for full-swing and low-swing links respectively.

3. ENERGY EFFICIENCY THROUGH DVFS

3.1 Introduction

With the scaling transistors, modern chip-multi processors (CMPs) are designed to accommodate bigger and bigger on-chip last-level cache (LLC). This design decision is in the effort to circumvent the off-chip memory bottleneck. The growing LLC in turn requires an increase of on-chip communication bandwidth. Although Networks-on-chip (NoC) are recognized as a scalable approach to addressing the increasing demand for on-chip communication bandwidth, its power consumption must be carefully examined. Indeed, the energy consumptions of on-chip communication fabrics and shared, last-level caches (LLCs) have grown to occupy a significant portion of the overall chip power, as much as 30% of total power in recent Intel's single-chip cloud computer [36].

The energy efficiency of NoC and LLC can be improved by Dynamic Voltage and Frequency Scaling (DVFS). DVFS has been widely used to reduce the energy consumption of a system when it is underloaded with the rationale that power should be provided based on dynamic requirement rather than a constant level. DVFS has been intensively studied not only for individual microprocessor cores but also for NoCs [74, 53, 77, 62, 30, 57, 68, 14, 18]. Much of this prior work, however, assumes a core-centric voltage/frequency (V/F) domain partitioning where the shared resources (NoC and/or LLC) are divided and allocated to the core-based partitions according to physical proximity. Although such a configuration allows a huge freedom in V/F controlling, it results in the large inter-domain interface overhead. In addition, since these shared resources are to be utilized as a whole, with cache line interleaving which equalizes the traffic amount and the cache accesses on each tile, per-slice V/F tunings makes little sense.

In this work, we assume that the entire NoC and LLC are on a single V/F domain such that the inter-domain overhead is largely prevented, and there is a unified policy over the whole shared resources. In the literature, a few studies have addressed DVFS for such scenario. Liang et al. propose a rule-based control scheme based on the network load [53]. Chen et al. propose a PI controller based on average memory access time (AMAT) [18]. Although both works demonstrate the energy savings from DVFS, they have two critical problems to overcome. First, the metrics based on which the controllers in those previous works decide on V/F levels may not be optimal. Second, both studies do not estimate the impact on the overall system performance with realistic workload.

The contributions of this study are two folds. We, first, introduce a new control scheme which controls V/F level based on the throughput of the LLC. We, then, evaluate the energy consumption and the overall system performance using a full system simulation with PARSEC [10] benchmarks. Our experimental results show that this new control scheme achieves better energy delay product than the previous work [18].

3.2 Preliminaries

3.2.1 Problem Description

Figure 3.1 illustrates our architectural assumptions. The entire chip is composed of an array of tiles where each tile includes a processor core and private caches. Each tile has a router which allows the corresponding tile to communicate with other tiles forming 2D mesh NoC. Each tile also has a slice of the shared LLC partitioned and distributed uniformly based on address. We refer the NoC and the LLC together as the *uncore* system in this study. Each tile has its own V/F (voltage/frequency) domain and the entire uncore works at a single V/F domain. NI in the figure denotes Network Interface which is the interface between a tile and the uncore.

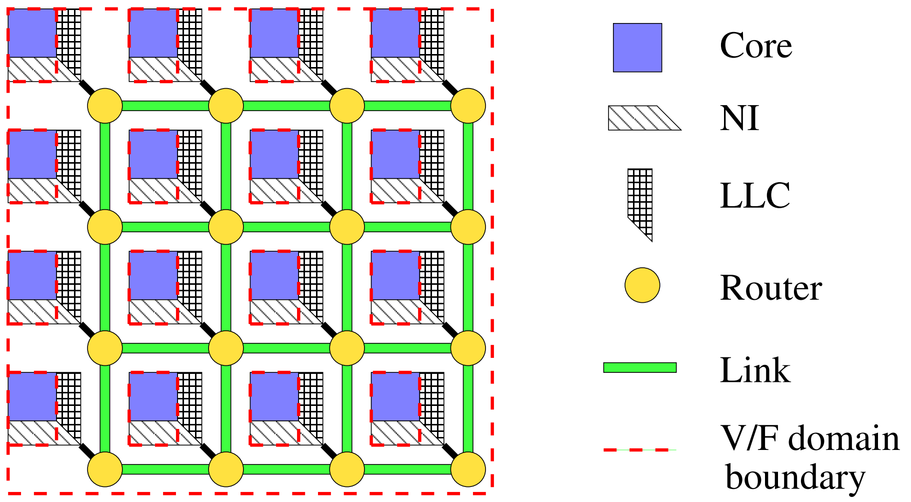


Figure 3.1: A multicore processor design where the uncore (NoC+LLC) forms a single V/F domain.

The problem we attempt to solve is formulated as follows:

Uncore Dynamic Voltage and Frequency Scaling: within a set time window, find the voltage/frequency level for the uncore such that the uncore energy consumption is minimized while the chip performance, in terms of total application runtime, has minimal impact.

Please note that the problem formulation in this study is different from previous works in NoC DVFS, which merely take care of the performance of NoC itself, not considering its utility to the entire system. In contrast, this study has the more challenging goal of optimizing uncore energy consumption considering the entire system performance. The uncore energy we try to minimize includes both dynamic and leakage energy of which models are well-known.

3.2.2 Previous Work

Liang and Jantsch propose a DVFS controller that controls network voltage and frequency such that the network operates just below the saturation point [53]. The controller increases (decreases) network V/F level by one step if the traffic volume is significantly greater (less) than the saturation point at each control interval. The metric used in this controller is the number of flits in the network and the saturation point is heuristically chosen. This technique overlook the fact that network load has not only temporal variance but also spatial variance which have significant impact on the network performance. In addition, network load does not always matter. For instance, many *store* operations may result in large network load but they are not critical to the overall system performance. The controller they use may not be spontaneous enough for bursty traffic as it only changes one step V/F level in each control interval.

In [18], Chen et al. introduce a PI controller based on AMAT (average memory access time). AMAT is formulated as the average turn around time of *load* operations, hence it includes the effects of the private caches, NoC, LLC and off-chip memory reflecting network load and contention inherently. Therefore, it captures a wider system effects than the network oriented metric in Liang and Jantsch's approach [53]. The AMAT, however, fails to separate the LLC and NoC utility to the core from the effects of the off-chip memory. Thus, programs which frequently miss in the LLC, causing off-chip memory accesses, will lead to high AMAT values, and, in turn, high LLC and NoC frequencies, despite the low LLC utility in this case. Furthermore, when programs include few memory operations, which fall into private cache misses resulting in high AMAT values, the controller loses the opportunity of energy savings by operating the network at unnecessarily high V/F levels. As the saturation point in [53], the reference point for the PI controller is also empirically chosen via offline simulations. The fixed reference point makes the controller less adaptive to various range of applications. Moreover, neither work provides performance results from full-system simulation to validate their approach.

3.3 Throughput-Driven DVFS

To overcome the downsides of the previous work, we propose to use a throughput metric with which a naturally dynamic reference is enabled for the DVFS PI control and more aggressive energy saving is achieved. Throughput, $R_{U,Out}$, is the amount of data processed by the uncore per unit time, or the rate of data flowing out of the uncore to cores. Injection rate $R_{U,In}$, on the other hand, is defined as the cores' requested data rate. In the long run, outside of saturation, the throughput should generally follow the injection rate.

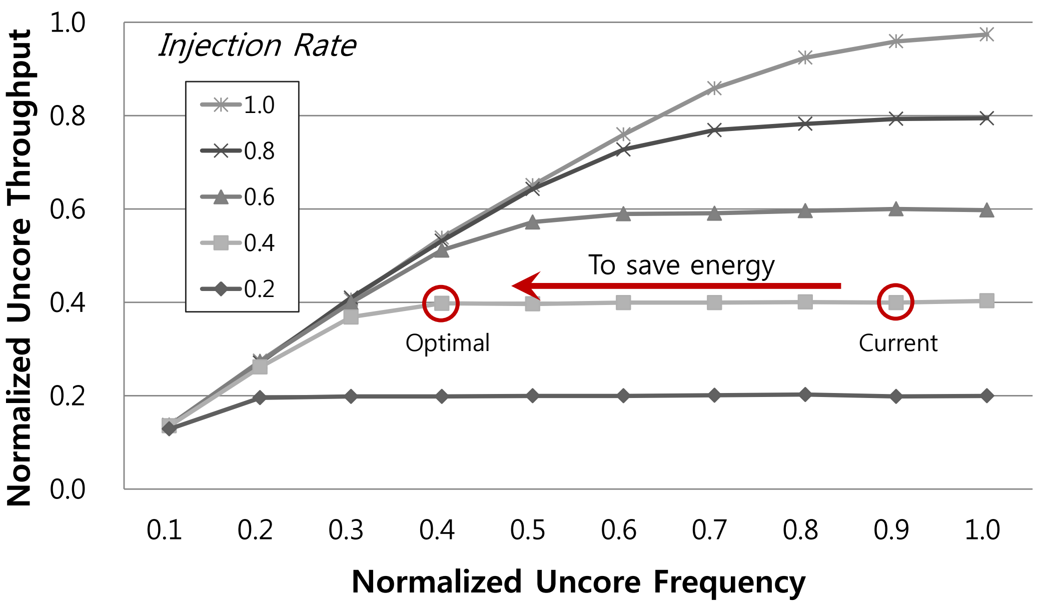


Figure 3.2: Throughput vs. uncore frequency.

The goal of this controller is to operate uncore at the minimum V/F level while ensuring the throughput satisfy the injection rate. Figure 3.2 depicts the throughputs for various injection rates with respect to varying uncore frequency. As the uncore frequency increases, we observe that the throughput increases asymptotically to the given injection rate. In a transient period, e.g., control interval i, if the uncore frequency is not high enough, the throughput $R_{U,Out,i}$ can be different from data injection rate $R_{U,In,i}$ of the same interval. We have then a PI controller with state (output) variable $R_{U,Out}$ and reference point $R_{U,In}$, which is dynamically decided during operations. Intuitively, such controller ensures that $R_{U,Out}$ becomes $R_{U,In}$, and it possibly saves energy by setting the uncore V/F to a lower level than the maximum level. The reference point is dynamically decided during online operation. This approach overcomes the difficulty of reference point selection in PI controllers seen in prior work [18].

Finding the optimal uncore V/F level here, however, is not trivial when the uncore V/F level is higher than necessary. For example, in Figure 3.2, where the given injection rate is 0.4 and the current normalized uncore frequency is 0.9, the controller must reduce the frequency to 0.4 to save energy. In such a case, unfortunately, the error ($R_{U,In} - R_{U,Out}$) becomes zero and the PI controller will not change the uncore V/F level. To avoid this, we magnify $R_{U,Out}$ by a very small percentage δ, e.g., 1%, resulting in a slightly negative value of the error. By doing so, the controller moves the V/F level down to the saturation point for the given injection rate without significantly hurting the throughput. A similar approach has been used in [53] where the saturation point is statically defined for a certain synthetic workload. In our work, on the other hand, the controller dynamically finds the saturation point for any realistic traffic pattern.

The proposed controller requires care to evaluate the injection rate $R_{U,In}$, especially when the program phase changes to increase $R_{U,In}$. Typically, a core stalls after issuing a number of requests yet to be served. The maximum value of outstanding requests is determined by the capacity of its MSHR (Miss Status Handling Register) for out-of-order processors, or it is 1 in case of in-order processors. At this point, the core has to wait till a request is served by the uncore, and then it will resume injecting further requests. If there are a sizable number of such stalled cores, the overall injection rate does not grow enough to increase the uncore frequency, although each core actually starts generating more requests than before. To address this issue, the number of stalled cores and the duration of waiting are considered. If the number of cycles when there are more than N cores waiting their uncore requests is M, and the size of control interval is P, the effective injection rate, which is also the reference request rate for the controller, is estimated by $R_{ref,i} = R_{U,In,i} \cdot (1 + \frac{M}{P})$ where i is the index of control interval. Then, the overall error function for the throughput-driven controller is:

$$e_i = R_{U,In,i} \cdot (1 + \frac{M}{P}) - R_{U,Out,i} \cdot (1 + \delta) \tag{3.1}$$

where δ is a small number, e.g., 1%. The value of N affects the behavior of the controller as the other controller tuning parameters. One can use a greater value of N to achieve more energy savings but at the higher performance cost. The value of N is set to 3 in our experiment for the best energy-performance trade off.

3.4 Evaluations

3.4.1 Experiment Setup

We use Gem5 [11] full system simulator with PARSEC shared-memory multiprocessor benchmarks [10]. The baseline architecture in this experiment is a 16-tile chip multiprocessor with a 2-level cache hierarchy. Each core is assumed to be an in-order Alpha like processor. Table 3.1 summarizes our experimental configurations and parameters. For each benchmark, the application is run in full-system mode; the results are obtained based upon statistics from the region of interest (ROI), which is the parallel part of the application. Gem5 "Ruby" memory system (L1-LLC+directory) and "Garnet" network simulator are used for all results. The control interval for the uncore DVFS is 50K core clock cycles. According to our experience and existing literature, such interval size allows sufficient time for the uncore V/F change to settle, and is sufficiently small to capture fine-grain program phase behavior. Both dynamic and leakage power are considered in the experiment. We use ORION 2.0 [44] and CACTI 6.0 [59] based on $65nm$ technology for the power models of NoC and LLC, respectively. The overall performance is evaluated as the execution time for the ROI of each application.

3.4.2 Experiment Results

We compare the energy efficiency of the AMAT based PI controller ("AMAT") proposed by Chen et al. [18] and the proposed throughput based controller ("Throughput") where the values are normalized to those of "baseline."

Figure 3.3 shows the energy efficiency in terms of energy-delay product (EDP) for different methods. "Throughput" achieves the lowest EDP among the techniques for every application in the PARSEC suite proving itself as the most energy efficient method. It improves EDP of "AMAT" by 46% approximately achieving 2$\times$ energy

Table 3.1: Simulation Setup

Parameter	Values
Core Frequency	1GHz
#processing cores	16
L1 data cache	2-way 256Kb, 2 core cycle latency
L2 cache (LLC)	16-way, 2MB/bank, 32MB/total, 10 uncore cycle latency
Directory cache	MESI, 4 uncore cycle latency
NoC	4×4 2D mesh, X-Y DOR, 4 flits depth/VC
Voltage/Frequency	10 levels, voltage: 1V–2V, frequency: 250MHz–1GHz
V/F transition	100 core cycles per step

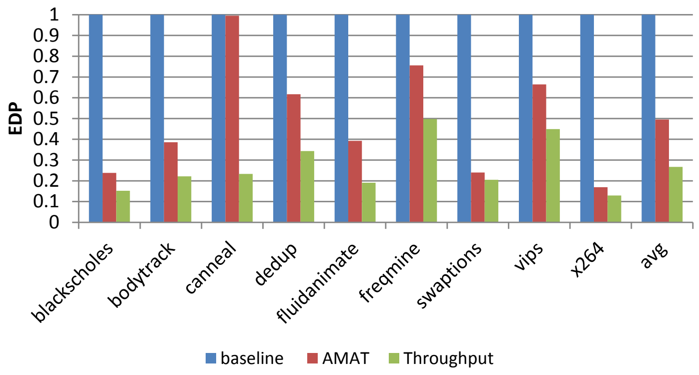


Figure 3.3: Energy delay product

efficiency, and 4× compared to "baseline." The graph also shows that "Throughput" even works where "AMAT" fails. In case of *canneal*, "AMAT" does not improve the energy efficiency. It may be the case where the most of memory operations result in off-chip memory access or the accesses to LLC is not so critical that the increased LLC delay does not affect the overall chip performance.

We found that the energy efficiency improvement is achieved by further energy savings rather than improved performance. Figure 3.4 and 3.5 depict, the normalized energy consumption and normalized program run time, respectively. In general,

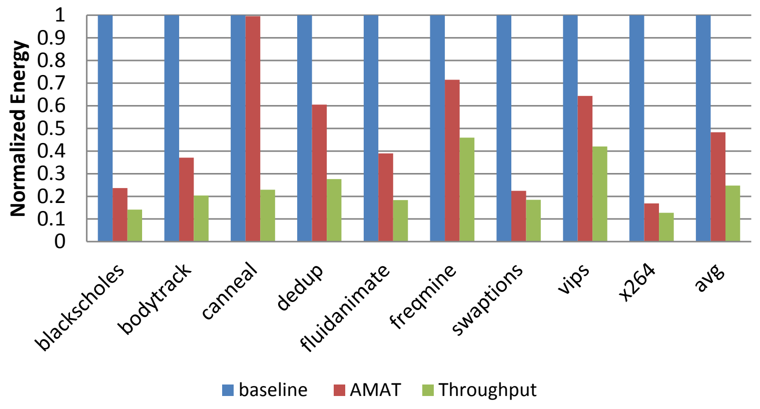


Figure 3.4: Energy consumption

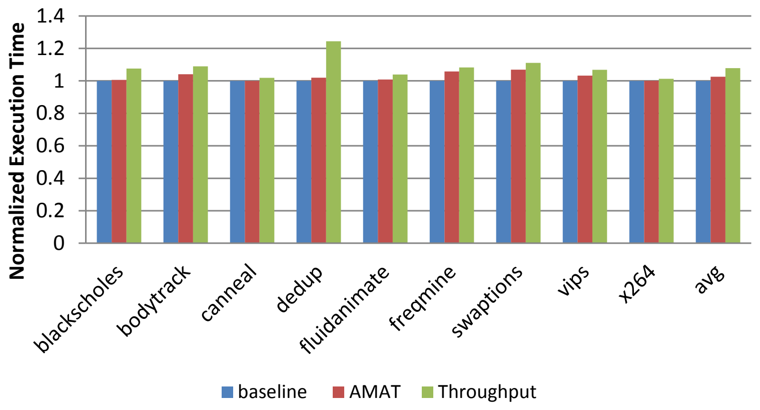


Figure 3.5: Execution time

"Throughput" reduce the energy consumptions to ∼50% of "AMAT." However, this aggressive energy saving results in 7.5% increase in execution time on average while "AMAT" only increases it by 2.5%.

3.5 Conclusions

In this study, we explore DVFS for NoC and LLC in CMP design. We propose a new metric and reference point for the PI controller overcoming the shortcomings of the previous work. We achieve more aggressive energy reductions at the cost of some increased execution time compared to the previous technique. The proposed technique is evaluated on benchmark widely used in architecture research using full-system simulation.

4. WEAR-OUT AND LIFETIME IN FUTURE CHIP MULTIPROCESSORS

4.1 Introduction

The continuous aggressive miniaturization of CMOS feature sizes and the resulting increase in transistor density has recently sparked the multicore era. Architects have harnessed this increasing supply of transistors, resulting in the design of parallel systems, including Chip Multi-Processors (CMPs) [36]. In these systems, the on-chip interconnect, typically organized as a Network-on-Chip (NoC) [21], plays a vital role in enabling communication among the various on-chip computational, memory and peripheral components, as illustrated in Figure 4.1. Unfortunately, deep sub-micron CMOS process technology is marred by increasing susceptibility to wearout [39], dramatically shortening the useful lifespan of such on-chip parallel systems. As we will illustrate, wearout does not effect all components equally, wear of the cores can often be managed, while wear of the NoC interconnect can be catastrophic. Furthermore, we will show that wear in the NoC is highly dependent upon the operational stresses caused by real CMP workloads. In this work, we develop techniques to *proactively* maintain the CMP NoC in the face of workload-dependent wear, and hence improve the overall functional lifetime of the CMP as a whole.

Two key operational stress-induced wear mechanisms in current and future CMOS technology are Hot Carrier Injection (HCI) and Negative Bias Temperature Instability (NBTI) [61]. Both HCI and NBTI lead to a shift of the transistor's threshold voltage, eventually leading to switching delay and critical path degradation [41]. Though these effects do not result in circuit opens or shorts, over time they can cause critical path timing violations. Given equivalent supply voltage and temperature, HCI and NBTI degradation is primarily dependent upon the time transistors

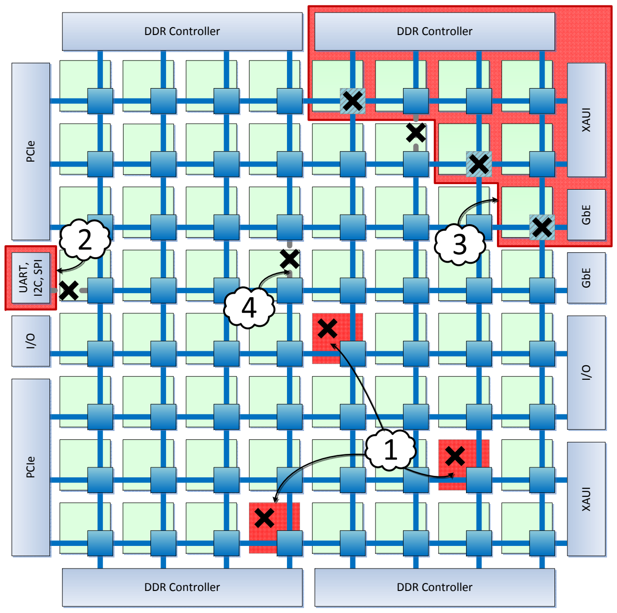


Figure 4.1: A 64-core CMP interconnected with an 8×8 2D mesh NoC. Components marked with a black $\times$ illustrate wearout failure. The failure scenarios are as follows: (1) failure of cores; (2) peripheral device disconnected from the system due to link failure; (3) network segmentation resulting in a disconnected sub-network; (4) individual link failure.

have been operating under stress. These types of stresses are primarily data- and usage-dependent, in terms of the activity factor (i.e. the fraction of cycles in which a transistor switches) and duty cycle (i.e. the percentage of time the gate's voltage is held at a constant zero), respectively, of the gates in typical CMOS logic circuits.

Prior work notes, individual core wearout and failure need not be catastrophic to the functionality of many-core CMPs due to the inherent core redundancy that a CMP implies [45, 13, 76, 65, 52, 37]. With increasing numbers of cores, a proportionally smaller portion of the overall system's required throughput is dependent upon each individual core. Thus, as illustrated in the component failure scenario (1) of Figure 4.1, failure caused by wearout of some cores need not result in full-system failure. Instead the system could suffer some performance loss while preserving correct functionality, assuming core-level error detection and appropriate OS support is available [13, 76, 65, 52, 37].

For the NoC interconnecting the cores, however, the assumption of redundancy-based wear resilience breaks down, as illustrated in component failure scenarios (2), (3) and (4) of Figure 4.1. In the same figure, scenario (2) illustrates the case where a wearout-induced link failure has made it impossible to access a key I/O peripheral, while in scenario (3) link and router wearout has partitioned away a large fraction of the network, making all cores and I/O components inaccessible to the rest of the system. In both cases, wearout is catastrophic, in that the system will likely be rendered unusable due to these failures, unlike the core wearout in scenario (1) discussed earlier. Even scenario (4), in which a single link is broken due to wear-induced failure, might lead to a communication protocol-induced deadlock(s), or subnetwork isolation, if the network is not provisioned to address wear-induced failures.

Prior work has proposed various fault-tolerant routing algorithms and fault-insensitive router and link designs in an attempt to manage faults as they occur [84, 73, 25, 9, 8, 24], however, network isolation and key resource partitioning cannot be fully resolved using only such *reactive* techniques. Ideally, one would prefer to develop *proactive* mechanisms to extend the healthy status of the system without failure, rather than react to the faults once they occur. Such proactive mechanisms could be coupled to the reactive mechanisms, in the hopes that the latter would be required less frequently as faults in the system would occur less frequently.

In this work we present such a proactive technique, designed to decelerate the effects of aging in the NoC of a CMP. Based upon detailed HCI and NBTI transistor-level aging models, we develop a novel, critical path-based model to characterize the effects of aging related wear. Based upon this model we analyze the NoC router microarchitecture to find the paths most susceptible to wearout. Using real workloads from the PARSEC benchmark suite [10], we characterize various wearout mechanisms that map onto those paths. Finally, we develop a wearout-resistant router microarchitecture which prolongs circuit lifetime through targeted mitigation techniques with negligible influence on the router's timing, pipeline, CMOS area requirements, and power consumption. This proposed technique yields a 13.8×-65× increase in CMP lifetime.

This paper provides the following contributions:

- Generalized, path-based, microarchitecture-level (rather than device-level) HCI and NBTI wearout-induction models.
- Characterization of NoC router and link wearout due to HCI and NBTI under realistic workloads from the PARSEC benchmark suite [10].
- A novel wear-resistant router microarchitecture which dramatically improves interconnect lifetime, and hence full-system survivability in the presence of both HCI- and NBTI-wearout mechanisms.

This paper is organized as follows. In Section 4.2 examines existing transistor-level models for HCI- and NBTI-induced wear. Next, in Section 4.3 examines the sensitivity of the router's critical path to wear by analyzing the activity and duty cycle of its critical path, and thereafter characterize the router's wear caused by the behavior of real application workloads. From these wearout models, in Section 4.4 we then develop a circuit path delay model for workload stress-induced wear. In Section 4.5 we propose a novel router microarchitecture to improve the lifetime of NoC routers under realistic workloads, while Section 4.6 evaluates the proposed design. Finally Section 4.7 presents prior related work, while Section 4.8 concludes.

4.2 Background

Prior research shows that the two dominant CMOS transistor physical failure mechanisms are Negative Bias Temperature Instability (NBTI) and Hot-Carrier Injection (HCI) [60]. Under both failure mechanisms charge becomes trapped in or near the gate oxide resulting in a slow increase the threshold voltage of transistors (V_{th}). This in turn causes the delay in transistor state switching to expand.

In traditional synchronous circuit CMOS designs, the clock frequency of a given design is determined by the circuit path which exhibits the longest latency between its end latches, within a given system design. This *critical path* comprises a number of chain-connected gates between these end latches. HCI- and NBTI-induced aging progresses gradually in extending the delay of each gate found in this chain, slowing down the entire critical path. In modern CMOS designs, due to this age-induced slow-down, and other causes, such as process variation [48], designs are given timing guard-bands so as to guarantee their intended functionality for a certain duration of time [4]. Once the aggregate increase in delay along a timing-critical path exceeds this guard-band, due to the aggregation of increasing delays occurring in individual gates along this path, the functionality of the system is no longer assured. The moment at which this timing violation first occurs defines the system's useful life span.

In this section, we first describe the impact of these aging mechanisms upon V_{th} using transistor-level analytical models. We then examine a number of specific NoC router critical paths which are most susceptible to these aging effects, since they are the critical paths which determine and impact a system's maximum clocking rate.

4.2.1 Failure Mechanisms

Design rules and operating conditions are precisely chosen to ensure correct product functional operation over its intended lifetime [51]. To obtain a given level of performance, when utilizing an integrated circuit, under various design constraints, it becomes imperative to create and analyze the reliability model of the digital system under consideration and design.

As previously discussed, the HCI and the NBTI mechanisms do not induce failures, rather shift parameters over time under circuit operational stresses. The Reaction-Diffusion (R-D) model uses the threshold voltage V_{th} shift as a proxy of NBTI and HCI stress [80]. The threshold voltage shift causes transistor delay degradation according to the alpha power law [71]:

$$d_g \propto \frac{V_{dd}}{\mu(V_{dd} - V_{th})^{\alpha}} \tag{4.1}$$

where $\mu \propto T^{-1.5}$ and $\alpha = 1.3$.

Lifetime can be defined as the time until an important material of a component or device parameter degrades beyond the point at which the device or circuit can function properly in its originally intended application. For a single gate, when ΔV_{th} reaches some level (in practice, it is usually 10% [80]) the transistor is considered to be over-aged. For the multi-gate path, the cumulative transistor delay shift increases faster than a single gate's worst-case delay degradation. Therefore, a total gate delay shift over the entire path, when its value reaches or exceeds the 10% threshold mark, can serve as a lifetime period indicator.

Figure 4.2 shows a typical CMOS inverter, indicating the failure mechanisms associated with each type of transistor: HCI affects both the nMOSFET and pMOSFET transistors, while NBTI affects only the pMOSFET transistor (note that PBTI

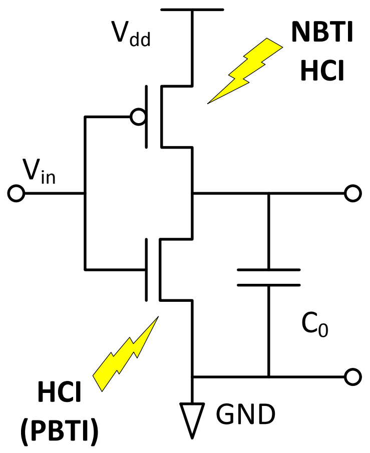


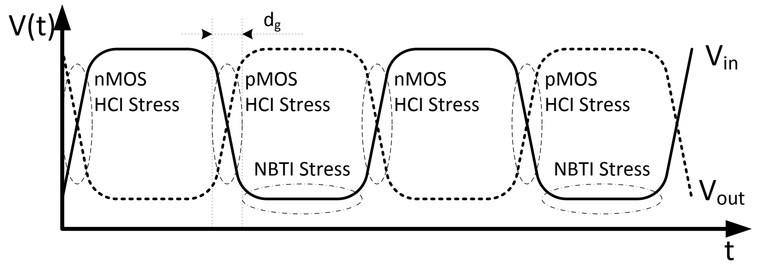
Figure 4.2: CMOS inverter

is the complement of NBTI and affects nMOSFET transistors only, however, its effect is generally considered to be much smaller than that of NBTI). The following subsections present a device parameter degradation model that captures the HCI and NBTI wearout effects.

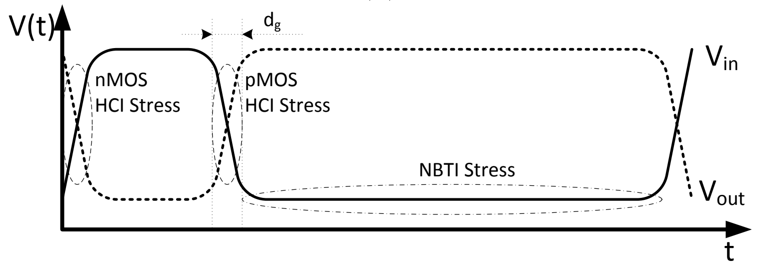
4.2.1.1 Hot Carrier Injection (HCI)

Hot Carrier Injection (HCI) is a wear-out mechanism which occurs when carriers flow along the channel in MOSFET transistors and gain sufficient kinetic energy to be injected into the gate oxide resulting in a charge trap and interface state generation. This leads to a gradual transistor parameter shifting, including switching frequency degradation, rather than causing an immediate failure event [41].

A substrate current-based (I_{sub}) model is commonly used to estimate the effect of HCI. Prior work shows that transistor threshold voltage shifts due to HCI under



(a)



(b)

Figure 4.3: HCI and NBTI stress time windows for a CMOS inverter: (a) higher switching activity, lower duty cycle; (b) lower switching activity, higher duty cycle.

DC stress is

$$\Delta V_{th\_HCI}|_{DC} = A(I_{sub})^m t_{stress}^n, \tag{4.2}$$

where A is the material-dependent parameter, t_{stress} is the stress time, and n and m are technology-related exponents [51, 41].

According to the alpha power law 4.1, the delay of a transistor depends linearly on threshold voltage for small shifts, so the gate delay shift can be expressed as

$$\Delta d_{g\_HCI}|_{DC} = \hat{A}(I_{sub})^m t_{stress}^n, \tag{4.3}$$

where $\hat{A}$ is a fitting constant.

The lifetime of a device exposed by a direct HCI effect is [41]:

$$TTF_{HCI}|_{DC} = A_{HCI}\,(I_{sub})^{-N}\,exp\left(\frac{E_{aHCI}}{kT}\right), \tag{4.4}$$

where E_{HCI} is the apparent activation energy, I_{sub} is the substrate current under stress at $V_G = V_D$, T is the run-time temperature, k is the Boltzmann's constant, N is the technology-related exponent, and A_{HCI} is a fitting constant.

HCI stresses the device only during dynamic transitions when current flows through the device. Figure 4.3 shows voltage waveforms of a standard CMOS inverter. The pMOSFET transistor suffers HCI stress when the output of the inverter is pulling-up and C_0 is charging up (see Figure 4.2). The nMOSFET transistor experiences HCI degradation during the reverse dynamic stage, when the output of the inverter is discharged to the ground (low-voltage level) [51]. Thus, each of the CMOS transistors experiences degradation only during half of a switching period, and hence the relation between HCI stress time t_{stress}^{HCI} and run-time t can be derived as:

$$t_{HCI\_stress} = d_g f \alpha t, \tag{4.5}$$

where d_g is the transition delay, α is the switching activity, and f is the clock frequency.

Since HCI stress occurs during the device turn-on and turn-off periods, the impact of HCI under AC stress can be extracted from 4.3 and 4.4 using 4.5:

$$\Delta d_{g\_HCI}|_{AC} = A(I_{sub})^m (d_g f \alpha t)^n, \tag{4.6}$$

And finally, the last equation is transformed into relation for HCI lifetime:

$$TTF_{HCI}(T, \alpha)|_{AC} = A_{HCI} \frac{1}{d_g f \alpha} (I_{sub})^{-N} exp\left(\frac{E_{aHCI}}{kT}\right). \tag{4.7}$$

This relation shows that the lifetime of a transistor due to *HCI is inversely related to the switching activity α of the gate input.* Hence, frequent switching, such as those shown in Figure 4.3-(a), not only increase the dynamic power consumption, but also speed-up the aging effect, whereas a gate with less frequently occurring transitions, as shown in Figure 4.3-(b), will experience lighter HCI-induced aging.

4.2.1.2 Negative Bias Temperature Instability (NBTI)

Negative Bias Temperature Instability (NBTI) is a wear-out effect that influences pMOSFET transistors as long as they operate in inversion (i.e. a '0' voltage on the input of an inverter, as shown in Figure 4.2). Thus the data-dependent stress caused by NBTI is very different from that of HCI, which is under stress during voltage level switching. NBTI changes the pMOSFET transistor parameters over time. In particular, it leads to an increase in the threshold voltage (V_{th}), as well as a reduction in the drive current due to charge carrier mobility degradation. As with HCI, NBTI does not result in complete circuit failure, but rather in circuit speed degradation. JEDEC reports that process technology scaling will lead to a larger NBTI-induced threshold voltage in pMOSFET transistors [41]. It has been reported, unlike for HCI, some degree of recovery from NBTI degradation can occur in the event that a relaxation period occurs after the stress period [41, 51, 80].

Although NBTI models under DC stress are widely discussed in the literature [41, 51], they are not suitable for architectural-level modeling. The AC stress condition is more realistic model of high-frequency CMOS operation, therefore we employ the approximated long-term NBTI model developed by Lu et al. [55] which provides a theoretical upper bound estimation of the NBTI effect in terms of time

$$\Delta V_{th\_NBTI} = A\left(\frac{\beta}{1-\beta}\right)^{n} t^{n} exp\left(-\frac{nE_{aNBTI}}{kT}\right), \tag{4.8}$$

where E_{aNBTI} is the apparent activation energy, T is the run-time temperature, t is the operating time, k is the Boltzmann's constant, n is the time exponent, and A is the fitting constant [55].

According to the alpha power law 4.1, the first-order gate delay can be approximated as a linear function of the threshold voltage. Hence, the gate delay shift can be expressed as:

$$\Delta d_{g\_NBTI} = \hat{A}\left(\frac{\beta}{1-\beta}\right)^{n} t^{n} exp\left(-\frac{nE_{aNBTI}}{kT}\right). \tag{4.9}$$

The lifetime of a single transistor under AC stress can be derived from 4.9 as follows:

$$TTF_{NBTI} = \left[A_{NBTI}\left(\frac{1-\beta}{\beta}\right)^{n} exp\left(\frac{nE_{aNBTI}}{kT}\right)\right]^{1/n}. \tag{4.10}$$

Thus, lifetime degradation due to *NBTI depends on the duty cycle of the input signal.* Transistors which experience a smaller duty cycle, such as the duty cycle shown in Figure 4.3-(a) in comparison to the duty cycle shown in Figure 4.3-(b), experience a slower degradation rate.

4.2.1.3 HCI and NBTI failure mechanism analysis

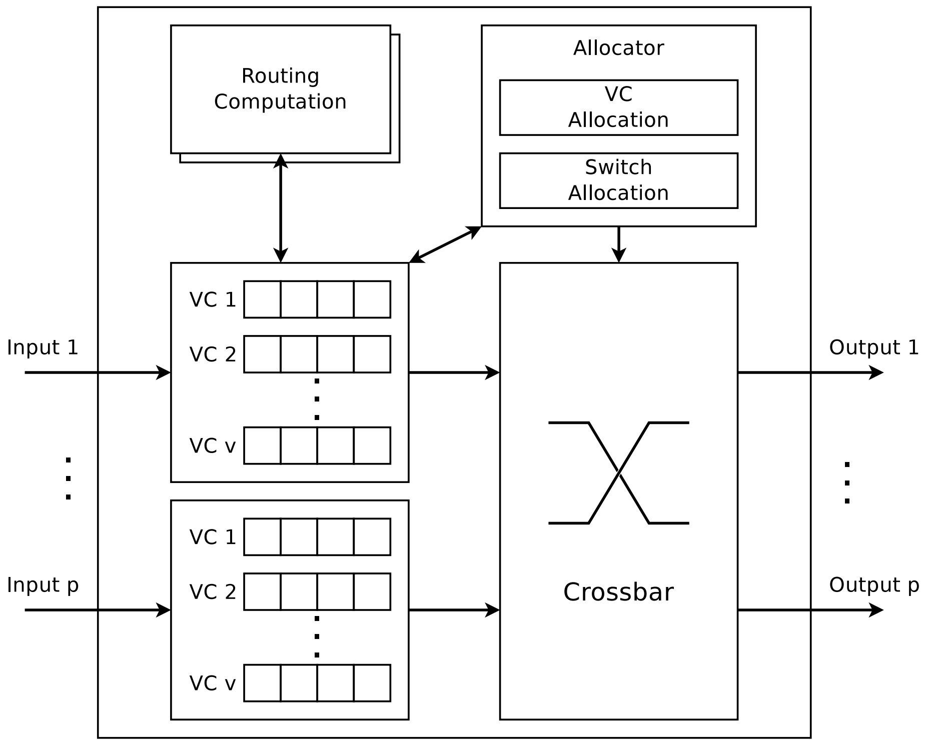
It may first appear that a technique which improves device lifetime by decreasing NBTI must come at the cost of a comparable degradation caused by HCI-related wear (and vice-versa). We note, however, that the activity factor α *is not the inverse of duty cycle* β; when β is large, it is possible to make a substantial change to β without proportionally impacting α. Furthermore, because of the $\frac{1}{(1-\beta)}$ term in Equations 4.8 and 4.9, large β's tend to have a disproportionate impact on aging-related slow-down. Even a small improvement in the value of β can therefore have a substantial positive effect on the overall device lifetime (especially when β is relatively large).

4.2.2 Router Microarchitecture

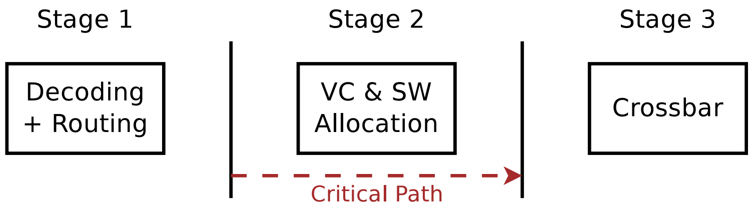
The canonical NoC virtual channel router was proposed by Peh and Dally [64]. Its block diagram is shown in Figure 4.4(a). The major building blocks of this NoC router are input channels, a crossbar (switch), and the control logic which includes the switch and virtual channel allocators. When used in a 2-D Mesh NoC architecture, typically five input and output channels (p) are used to connect its four immediate neighbors at the cardinal points, and its local processing element. An input channel is composed of a given number of virtual channels (VCs), each of which includes registers to keep track of their statuses, and buffers to store flits (flow-control units, a logical fixed-segment of a packet). The routing units also accesses the flits found in the input channel, and determine the next-hop direction packets should take, in the east, west, north or south directions. The VC allocator assigns a free virtual channel in a downstream router to a head flit, the first flit of a packet. If the head flit successfully obtains a VC, it competes with any other flits destined for the same output port during the switch allocation. The body and tail flits in the same packet skip the routing and VC allocation stage, and directly proceed to the switch allocation stage. Once the switch allocation is done, then a flit traverses the crossbar.

Our baseline router performs both VC and switch allocation during the same cycle by speculatively allowing a packet to compete for the switch while it is yet competing for the VC at the downstream router [64]. Figure 4.4(b) shows the baseline router pipeline. Flit decoding and routing computation are done in *Stage 1.* The combined VC and switch (SW) allocation is done in *Stage 2.* In *Stage 3*, flits traverse the crossbar.

Since NBTI and HCI decelerate transistors, the damage from the circuit aging



(a) Router Block Diagram



(b) Router Pipeline Stages

Figure 4.4: Baseline router

first takes place where timing is most critical. To determine the critical path of the baseline router, we adapted our baseline RTL from the publicly available router RTL model designed by Becker [7]. This RTL model was synthesized using Synopsys Design Compiler and mapped to TSMC 45 nm standard cell library to a 1GHz frequency. All critical paths with slack less than 10% of the clock frequency were gathered using Synopsys Design Vision and analyzed off-line.

The results of this analysis are highlighted in Figure 4.4(b). We found that all timing critical paths (i.e. those within 10% of the 1GHz clock frequency), pass through the VC and switch (SW) allocator. These results correspond well with prior research [64]. The utilization of the allocators is closely related to the incoming rate, or the number of flits traveling through this router per cycle, because the allocators are enabled by the input channel which sends the request signal to the allocators when it has flits to forward. As the critical path is initiated by the request signal, the wire activity along the path is dependent upon how often the request signal is set, which in turn is determined by the workload's utilization of that router. We therefore expect that the stress time for HCI and NBTI, which is closely related to the activity factor and the duty cycle of the wire, thereby should be also closely correlated to the utilization of the router. In the next Section, we perform an in-depth study on the impact of typical CMP workloads on the router's critical path in terms of activity factor and duty cycle.

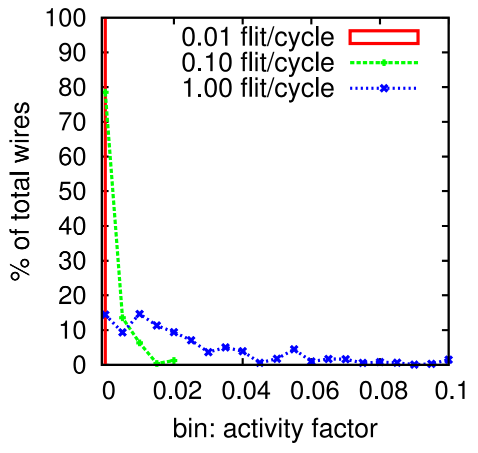
4.3 Router Wear Characterization

To examine the sensitivity of the router's critical path to HCI and NBTI wear, we first analyze the activity of wires residing in the baseline router under synthetic workloads. The router has 5 physical channels, 4 virtual channels per physical channel, and a 4 flit-deep buffer per virtual channel. Oblivious minimal XY routing is used. The network is designed to transfer 64-byte memory blocks where the link-width between adjacent routers is 128 bits, discounting any flow-control signals. Hence, if a packet includes such data it is composed of 5 flits (1 head flit for routing information and meta-data, and 4 data flits), otherwise, it is composed of only 1 head flit. The workload is generated by controlling the number of flits per cycle injected to this router while the portion of 1-flit and 5-flit packets are maintained to be 50-50%. As described in Section 4.2.2, the critical paths from the post-synthesis router model with 10% or less slack relative to the 1GHz clock frequency assumed, were examined. In particular, information about the activity factor and duty cycle of all wire nodes along each critical path under these workloads was retained for analysis.

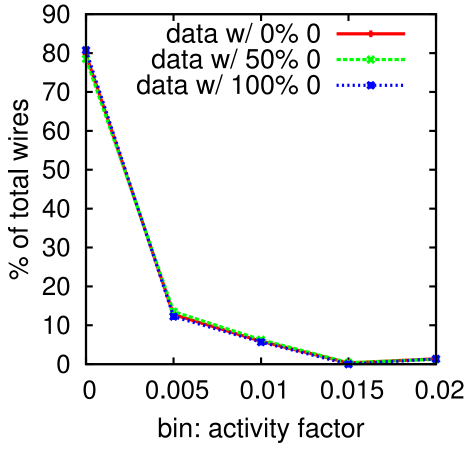
4.3.1 Activity Factor

HCI is proportional to the activity factor of the interconnect NoC wires, such that a higher activity factor results to an accelerated (higher) aging rate (refer to Section 4.2.1.1). Figure 4.5(a) shows the histogram of activity factors of the wires on the critical paths of an NoC router with respect to varying incoming rates. The first observation we make is that the nodes have a quite low activity factor, where the vast majority is switching at less than 10% of the time (activity factor of 0.1).

Intuitively, the higher incoming rate should cause a correspondingly higher activity factor. This implies that a router running for applications that inject network traffic more frequently ends up aging at a faster rate. Hence, it is desirable to reduce



(a) Activity factor with respect to the incoming rate



(b) Activity factor with respect to data content, in terms of the percentage of zeros in a data vector(512 bit)

Figure 4.5: Sensitivity to the activity factor.

(or keep low) the incoming rate so as to improve a router's longevity. Interestingly, however, the activity factor does not appear to be very sensitive to the incoming rate, in that it does not increase as much as the incoming rate increases. For instance, even at the very high incoming rate of 1.0 flits/cycle, the activity factors of most of the wires remain at a relatively low value and only 7.7% of wires have an activity factor greater than 0.1.

We observed in our preliminary study that applications differed in their data value payloads that they produce, due to differing bit-level data biases in their cache blocks. One might expect that the content of the data traversing the network might also affect the activity of routers since the NoC is mainly used to transfer these cache blocks. Figure 4.5(b) shows the histogram of activity factor with various data contents (percentage of zeros in each data-flit vector) at a fixed incoming rate of 0.10 flits/cycle, to examine the router's critical path activity factor sensitivity to data content. The data is modeled to include 0%, 50% and 100% of "0" bits in the

512-bit payload cache lines. As the figure shows, we find that the data content does not affect the activity factor of the wires along the router's critical path (all lines are almost perfectly overlaid). While the transferred data may bias the activity factor on other data path components, such as buffers and the crossbar, it does not appear to have much impact on the allocators that make up the primary critical path of the router.

4.3.2 Duty Cycle

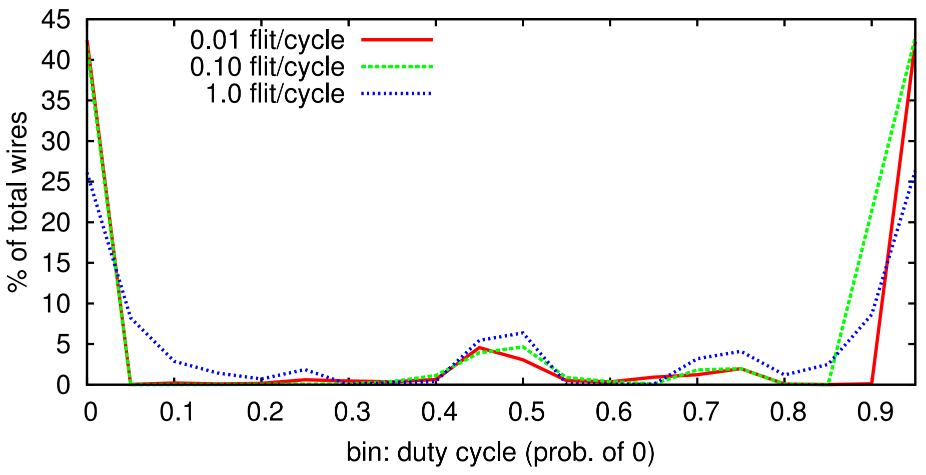
NBTI is highly sensitive to the duty cycle of gates (see Section 4.2.1.2). Figure 4.6(a) depicts the histogram wires along the critical path with a given duty cycle for different incoming rates, in terms of flits per cycle. In the figure, the width of each bin is 0.05; hence bin[0] shows the percentage of gates with duty cycle in the range of [0, 0.05) and so on. In general, significant portions of gates fall into the bins of the duty cycles of near 0, 0.5 or near 1.0, regardless of the incoming rate. As the incoming rate grows, the bins at the two ends of the spectrum fall while the central part moves up, indicating that an increasing incoming rate causes less skew in the duty cycle towards these extremes. Despite this change, we found that the average duty cycle hardly changes, retaining a duty cycle of approximately 0.5 for all incoming rates. According to the netlist derived from the synthesized router, there are many inverters along these paths. Thus the duty cycles of the two wires, the input and the output of the inverter, β and $1 - \beta$ respectively.

Figure 4.6(b) and 4.6(c) magnify the two ends of the upper figure. To improve observability in the figure we use a narrower bin width of 0.001. With the increased resolution we note that higher incoming rates have a great impact on duty cycle at the extremes, pulling down the percentage of wires along the critical path with the highest and lows duty cycle from $\sim$35% to near 0%.

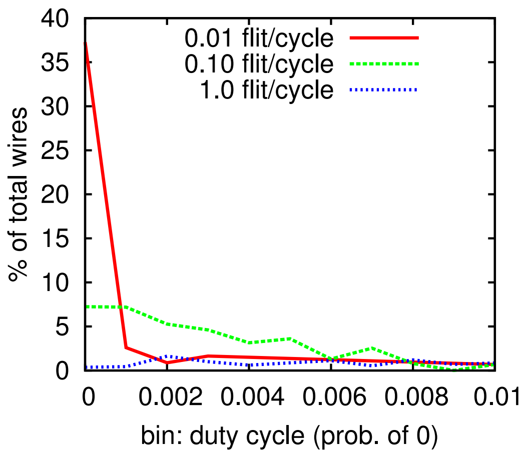
Although the incoming rate does not affect the average duty cycle, it actually causes notable differences in the NBTI impact on the gate delay. As shown in Equation 4.9, there is a non-linear relation between duty cycle and gate delay such that the gate delay shoots up as the duty cycle gets closer to 1.0. Hence, for example, having two gates in a given path with the same duty cycle of 0.5 it is better than having two gates with duty cycle of 0.0 and 1.0, respectively,in terms of the path-cumulative impact of NBTI upon gate delay. The delay increase due to NBTI from a single gate under a duty cycle of 1.0 alone will greatly exceed the sum of the delay increases from the two gates with duty cycle of 0.5 each. In this light, it is better to have a higher incoming rate and have more gates with duty cycles closer to 0.5, than to have a lower incoming rate and gates that have duty cycles closer to 1.0; although it is notably counterintuitive that wear-out occurs when routers are under-utilized. Based upon these observations we will next develop a multi-gate delay model in Section 4.4.

4.3.3 Workload Characterization

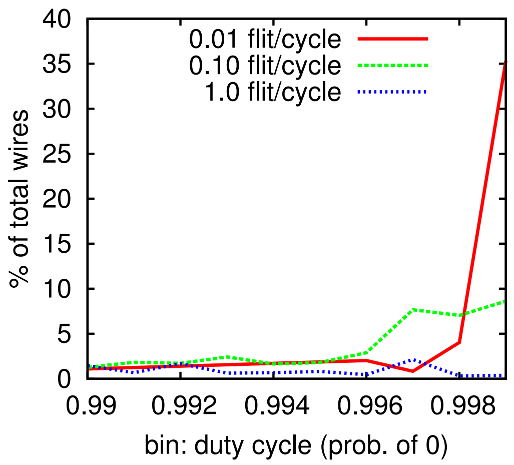
Having identified that the per-router incoming rate to be a critical workload characteristic that correlates with wear, we now examine the router-to-router incoming rate variance in realistic workloads. In this study, we use the PARSEC benchmark suite [10] as our workload, as these benchmarks mimic a range of representative next-generation large shared-memory multi-threaded programs for CMPs. The diversity of the PARSEC benchmarks makes them especially useful for this study, as they span a diverse range of emerging applications with varying on-chip communication spatio-temporal characteristics. Specifically, with the PARSEC benchmarks one observes different and varying behaviors in the NoC's packet (or flit) incoming rate, as will be outlined next in detail.



(a) range = 0∼1.0, bin width = 0.05



(b) range < 0.01, bin width = 0.001



(c) range > 0.99, bin width = 0.001

Figure 4.6: Histogram of duty cycle w.r.t incoming rate

The realistic workloads are generated from the gem5 simulator [11] emulating a 64-core system executing multithreaded programs in the PARSEC v2.1 [10] suite. Table 4.1 shows the details of the system setup used in our simulations. We first generate NoC traffic for each application for its region of interest (ROI), which represents the parallel portion of the application. We then count the number of flits traversing through each router to compute incoming rate to the router. Note that we use the term *incoming rate* here as the number of flits injected to a particular

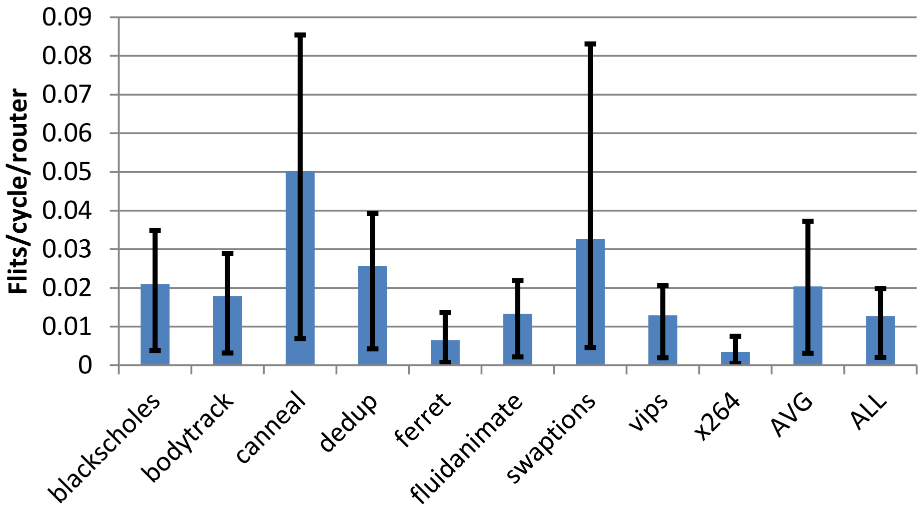


Figure 4.7: Average flit incoming rate per router for the entire range of the PARSEC benchmark suite (bars) and related range of incoming values (lines).

router, per unit time, rather than the number of flits generated and injected to the network as a whole. This includes the number of flits generated by the router's local processing element and the flits going through or headed for the router. In our experiments, we concentrated on capturing the incoming characteristics at each router, rather than capturing the inter-tile communication. The reason for concentrating on the incoming rate temporal characteristics is that the wire activity on the critical path is highly related to the frequency of flit arrival.

Figure 4.7 depicts the average number of flits injected into a router, captured by the solid bars, according to the aforementioned experimental setup, per unit time for each PARSEC benchmark. The incoming rate at routers, on average, is 0.02 flits per cycle (*AVG*). It varies across the programs under examination ranging from 0.003 (*x264*) to 0.05 (*canneal*). The average incoming rate also varies within the same application based on the cartesian location of the router, and the variance is captured by the dark line over each bar. The bottom of the line shows the incoming rate of the router which handles the least traffic amount among the routers in the network (*min incoming rate*), and the top of the bar denotes the incoming rate of the busiest router (*max incoming rate*). Hence, the per-router incoming rate under PARSEC workload actually varies between .0005 (*min* of *x264*) and .085 (*max* of *canneal*). In the graph, "AVG" shows the arithmetic means of the average incoming rate (the bar) and the *min* and *max* incoming rates (the line) across the entire range of benchmarks. "ALL" shows those three incoming rates when the system runs all the benchmarks, one at a time, sequentially.

In general, the average incoming rate is quite low, at 0.02 flits per cycle. Hence, *HCI-induced aging is not expected to contribute significantly to gate delay under these workloads*. Alternately, as discussed previously, a low incoming rate causes NBTI-induced aging. Thus, *routers running PARSEC workloads are exposed to accelerated NBTI-induced aging due to their light traffic*. Furthermore, there is an observable high variance in their spatially distributed network flit incoming rates, such that some routers executing the *ferret* and *x264* workloads experience even less than a 0.001 flits per cycle incoming rate. We therefore, focus on the NBTI aging in the remainder of this paper.

4.4 Path Delay

In Section 4.2.1, we examined existing formula characterizing the wear-induced transistor gate delay. While these equations accurately model the incremental breakdown of individual gates, we observe that a single gate is only one component of an entire critical path. In this subsection, we derive formula for lifetime comparison between two systems that operate under the same conditions , focusing on the observable microarchitectural-level. There are a number of delay models which take into consideration the aging effect in transistor or gate level, but few in architectural level. Ultimately, to calculate the point at which gate delay compromises timing along a particular path, one must examine the cumulative delta delay along that path. Here, we propose a method of lifetime computation along a path between latches, given the duty cycle of each gate along that path.

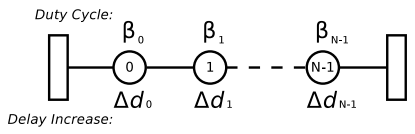


Figure 4.8: An example circuit path with multiple gates

We assume that a number of sequentially-placed gates in a serial chain comprise a circuit path as shown in Figure 4.8. Along the path, the delay increase due to the i-th gate with duty cycle β_i at time t can be simplified as:

$$\Delta d_i(\beta_i, t) = \psi \times t^n \times \left(\frac{\beta_i}{1-\beta_i}\right)^n \tag{4.11}$$

where the constant ψ includes all other terms in Equation 4.9 under the assumption that those will remain constant under the same condition. The sum of delay increase along a path with N gates at time t is computed as:

$$\Delta d(t) = \sum_{i=0}^{N-1} \Delta d_i(\beta_i, t) = \psi \times t^n \times \sum_{i=0}^{N-1} \left(\frac{\beta_i}{1-\beta_i} \right)^n \tag{4.12}$$

A system is reliable as long as the $\Delta d(t)$ of the critical paths is smaller than the guard-band. Hence, we define the lifetime, $T_{lifetime}$, such that $\Delta d(T_{lifetime}) < guardband$. The acceleration factor (AF) is defined as the ratio between the lifetime of the system under consideration, $T_{lifetime}(x)$, and a reference system, $T_{lifetime}(ref)$:

$$AF(x) = \frac{T_{lifetime}(x)}{T_{lifetime}(ref)} = \left(\frac{\sum_{j=0}^{M-1} \left(\frac{\beta_j}{1-\beta_j} \right)^n}{\sum_{i=0}^{N-1} \left(\frac{\beta_i}{1-\beta_i} \right)^n} \right)^{1/n} \tag{4.13}$$

where β_i is the duty cycle of the i-th gate on the critical path of the system under consideration, and β_j is the duty cycle of the j-th gate on the critical path of the reference system. In this equation, it is assumed that the number of gates on the critical path of the two systems are N and M, respectively.

4.5 Aging-Preventing Router Microarchitecture

4.5.1 Exercise Logic

The aging process is incoming rate-dependent along the critical path, as Section 4.3 outlined. The gate delay increases and the timing constraints are violated along the critical path first. A low incoming rate causes a biased duty cycle in the wire along the path and accelerates NBTI. Thus, the router needs work in the form of incoming packets to improve its longevity. However, increasing the incoming rate artificially yields other problems such as increased power consumption and acceleration of the HCI effect. Hence, the duty cycle must be improved without increasing the activity factor significantly. We now propose a method by which to exercise the critical path, compromising HCI in favor of NBTI, as the latter is more important. The goal of the proposed mechanism is (1) to improve the duty cycle by allowing the circuits to operate at a greater portion of their time in the “1” state, without affecting the actual data values being transferred, (2) to not change the state of the router, (3) to not worsen the critical path timing, and (4) to not significantly increase the activity factor which would compromise the entire goals of the mechanism by increasing the effects of HCI.

Figure 4.9 illustrates the critical path of a baseline router, along with our proposed modifications to reduce the wear-out effects of NBTI. The gates and wires in the lighter color (red) are our additions to the baseline router. The critical paths in an NoC router, as stated in Section 4.2.2, lie within the Virtual Channel (VC) and crossbar allocation stages handled by the router allocator. Each VC sends a one bit “Request” signal to the allocator to reserve a VC at the downstream router, and/or to bid for switch bandwidth at the crossbar so that the crossbar can be traversed by competing flits. There are p physical channels each of which has v virtual channels,

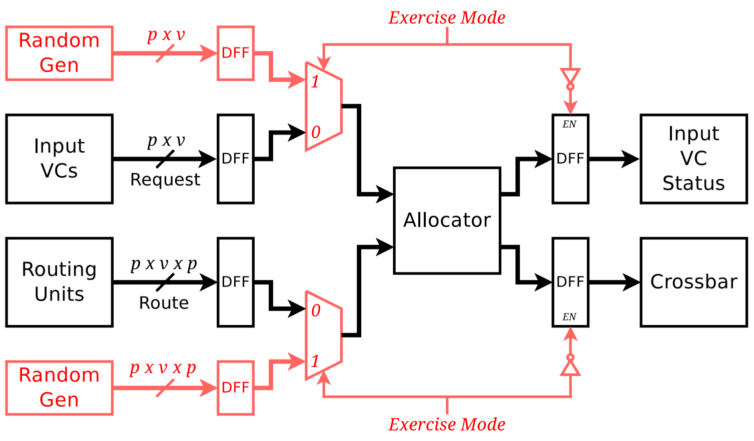


Figure 4.9: Exercise logic placed onto the critical path so as to improve the duty cycle.

hence, there are $p \times v$ of such control bits in total. Each "Request" signal has to be sent with a p bit-width "Route" signal giving the allocator the information as to where the corresponding flit is headed for, i.e. to which VC at a physical port downstream. Hence, there are $p \times v \times p$ bits for the "Route" signal.

The input signals to the allocator are biased so as contain more logical "0"s than logical "1"s, not only because the incoming rate is low, but also because only a couple of bits out of a greater number of bits of input signals will be set while handling a flit. If the flit arrival rate is r (flits/cycle) then the probability that a bit of the input signal will be set is also r. Since a flit in the router sets only one bit out of $p \times v$ "Request" bits, the probability of any wire being at logic "1" becomes $\frac{r}{p \times v}$. Since the duty cycle is defined as the probability of a wire being "0", it becomes, in this case, $1 - \frac{r}{p \times v}$. For instance, if there are 5 physical channels and 4 virtual channels, the duty cycle on this signal at the incoming rate of 0.01 flits/cycle becomes 0.9995 which will cause the NBTI effect shoot up. The lower incoming rate and the wider input signal combined, will result in a greater duty cycle. Hence, this is even worse in case of the "Route" signal which is composed of $p \times v \times p$ bits. Hence, these input signals modified to improve the duty cycle along the critical path in the allocator.

We propose to balance the duty cycles of these biased signals by injecting random data into the allocator. Figure 4.9 depicts how the proposed “exercise logic” works for the “Request” signal. First, the “exercise mode” is activated when there is no request from the input channels to the allocator. Then, the “Request” signal is replaced with a random vector which is generated by a random number generator (“Random Gen”) in the same figure. This random vector changes the duty cycle of the allocator input and of the wires along the paths through the allocator as well. Hence, it is more ideal to set the duty cycle at the input at a value of 50% instead of setting it at 0%, as there are inverters along the paths which will have the inverted duty cycle of the input. We utilize this additional logic for other inputs as well, such as the “Route” signals.

This additionally inserted circuit should not affect the router functionality. Hence, the “exercise mode” is enabled only when there is no packet in the input channels. Also, the random output from the allocator should not propagate to the next pipeline stages, such as the crossbar traversal stage. The states of the input virtual channels, similarly, should not be updated with the false output produced. Thus, the flip-fops (or latches) between the allocator and the logic gates in the next stage, are also disabled when “exercise mode” is turned on.

In the effort to minimize the impact upon the timing, and hence the clock rate, the random number generator and all other mode selection logic are placed off the critical path. We merely add a MUX (multiplexer) along the critical paths. A timing report of the synthesized proposed router micro-architecture acknowledges that the delay increase is negligible. It is so small that it will be absorbed by the chip’s guardband. As the circuit is used for a longer period and gets older, the additional circuit slows down the aging process, and in turn, the slows the rate of delay increase along the critical path. Therefore, adding this circuit is actually beneficial to timing/clocking in the long-term.

We explore two efficient mechanisms to implement the "Random Gen" component. First as a serial-update, parallel-read LFSR (Linear Feedback Shift Register)[1] and second as a set of pre-defined vectors, randomly generated at design time. The LFSR is composed of a 128-bit register, and we use the first 20 bits for the "Request" signal and the following 100 bits for the "Route" signal. The taps of the LFSR are placed at 0, 2, 27 and 99th registers such that it has the maximal length period. When "Random Gen" is implemented as a pre-defined set of random vectors, we generate a number of 120-bit random vectors and store them in a small ROM within the router. The quantity of random vectors is decided at design time and we tested a range of such numbers at 1, 2, 4, 8, 16, 32, 64 and 128 entries. This number has the implication on the circuit's energy consumption and lifetime which will be discussed later. In either case, the "exercise logic" adjusts the duty cycle by giving the allocator a series of random inputs, and this will increase the activity factor. In order to attenuate this side effect, the "Random Gen" changes its output vector once every pre-defined period of time. The length of this period is decided at design-time. We tested a range of such periods at 8, 32 and 128 clock cycles. These various values each has a different implication on the circuit's energy consumption and lifetime (see Section 4.6).

4.5.2 Other Approaches

In the earlier stage of this work, we also tried a couple of different approaches. As the NBTI effect is accelerated by a lack of load, we tried to solve this problem by

[1]Note, this is not a traditional implementation of an LFSR; a typical serial-read LFSR was found to consume too much energy to be practical for a random vector of this size. The trade-off being the extra iterations required to achieve pseudo-random data.

inserting artificial packets into the network to increase its load. However, this leads to other problems. First, it will significantly increase the network power consumption as we need substantially high incoming rate to each router. In addition, the increased incoming rate results in the increased activity factor, which accelerates the HCI aging. Second, distributing such artificial traffic is not trivial. Each router experiences different amount of traffic based on its location in the network. Some of the routers may already have enough traffic amount to avoid the NBTI effect while the other need more traffic. It is necessary and difficult to monitor the incoming rate of each router and to develop a routing algorithm, which steers the artificial packets to routers with low traffic load while preventing deadlocks. We also tried to solve the same problem with a gate-level solution. The NBTI aging is accelerated when a wire stays at logical "0" for a long time and, unfortunately, the input wires of the allocator are stuck at 0 most of the time because of the lack of load. Hence, one may come up with switching the meanings of "0" and "1" for the allocator logic in order to prevent those wires from staying at "0". However, we found that inverters are inserted to drive enough power along the paths when the logic is actually synthesized. Hence, forcing the input wires to stay at "1" eventually makes intermediate wires to be at "0".

4.5.3 NBTI Hardening

The proposed "exercise logic" can be deployed in general circuit designs. If the timing critical paths of the circuits are not frequently utilized resulting in extremely biased duty cycles on the wires, the "exercise logic" can be easily inserted for those paths to balance the duty cycle and mitigate the NBTI aging with its negligible overhead in terms of both performance and power consumption.

4.6 Evaluation

In this section we first examine the experimental setup. This is followed by a detailed exploration of the benefits and costs of our proposed technique.

4.6.1 Experiment Setup

The evaluated router, adapted from RTL code made publicly available by Becker [7] contains 3-pipeline stages. The detailed parameters of the router are listed in Table 4.1. Its RTL code was modified to include the "exercise mode" described in Section 4.5. It is synthesized using Synopsys Design Compiler and mapped to a TSMC 45nm standard cell library, achieving a clock frequency of 1GHz. We note that the added exercise circuit is largely off the critical path, with the exception of one additional, 2-input mux, thus, router synthesis produced the same clock frequency as the baseline router of 1GHz, although a small increase in the number of paths within 10% of the clock period was seen. The critical paths were extracted using Synopsys Design Vision. All paths within 10% slack versus the 1GHz clock cycle were retained and analyzed. The wire activity (activity factor and duty cycle) along the paths are extracted with Synopsys VCS. The power consumption is evaluated

Table 4.1: System Setup

Cores	64 on-chip, in-order, Alpha ISA
L1 Cache	32KB instruction/32KB data, 4-way, 64B lines, 3 cycle access time MESI coherent protocol
L2 Cache	64 bank fully shared S-NUCA, 16 MB, 64B lines, 8 way associative, 8 cycle bank access time
Memory	150 cycle access time, 8 on-chip memory controllers
Network	8x8 Mesh, X-Y routing, 4 VC/port, packet length: 1 flit or 5 flits,

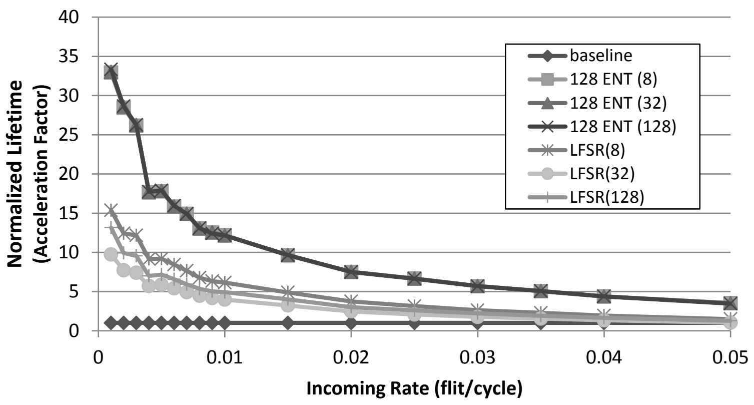


Figure 4.10: Normalized lifetime (acceleration factor) for router under a given synthetic incoming rate from 0.001 flits/cycle to .05 flits/cycle.

using Prime Time.

The router is evaluated under both synthetic and realistic workloads. The realistic workloads are captured as traces from the gem5 simulator [11] emulating a 64-core system executing multithreaded programs from the PARSEC v2.1 [10] suite. Table 4.1 summarizes the system configuration. Through offline analysis of these packet traces, we compute incoming rate of each router in 8 × 8 mesh network's individually under X-Y DOR routing. The per-router min, max and average incoming rates for each application were calculated. These rates are then applied to the synthesized router to extract the activity of the wires. For both synthetic and realistic workloads, we execute the post-synthesis router model for 100,000 cycles to measure the wire activity.

4.6.2 Experiment Results

4.6.2.1 Aging under synthetic workloads

We first examine the potential gain in lifetime of the router with the proposed technique versus the baseline router for a range of arbitrary incoming rates. We also compare the two kinds of “Random Gen” implementations. As previously discussed, the per-router incoming rate under PARSEC workloads tends to vary between .0005 (*min* of *x264*) to .085 (*max* of *canneal*). Figure 4.10 shows the normalized lifetime (i.e. acceleration factor, Equation 4.13) against a reference of the baseline router at the same incoming rate. As explained in Section 4.4, acceleration factor gives the lifetime of the system under consideration which is normalized to the lifetime of the reference system. In the graph, “baseline” means the normalized lifetime of the baseline router, which is always 1, while “128 ENT” and “LFSR” mean the cases when the “Random Gen” is implemented with 128 pre-defined random vectors and with the LFSR, respectively, and the number in the parenthesis shows the updating frequency.

The lifetime extension improves dramatically for the routers with lower incoming rate. The lower incoming rate causes the more bias in duty cycles, hence, the more room for the improvement. The “Random Gen” with the pre-defined random vectors (“128 ENT”) consistently outperforms the LFSR across the incoming rate. In addition, it is not sensitive to the updating frequency while the lifetime with the LFSR varies with varying updating period. The exercise logic with 128 random vectors achieves its greatest relative lifetime improvement of 33.3$\times$ at the incoming rate of 0.001 flit/cycle with any random number changing period. We note that for even smaller incoming rates the improvement will grow rapidly as discussed in Section 4.2.1.

On the other hand, the exercise logic with 8 cycle LFSR, “LFSR (8)”, achieves its greatest relative lifetime improvement of 15.4$\times$ at the incoming rate of 0.001 flit/cycle. Despite its diverse random vectors, “LFSR” does not work better than the “128 ENT” schemes, which only have a limited set of random vectors. We observe that using the LFSR does not generate well-balanced random 120-bit vectors as

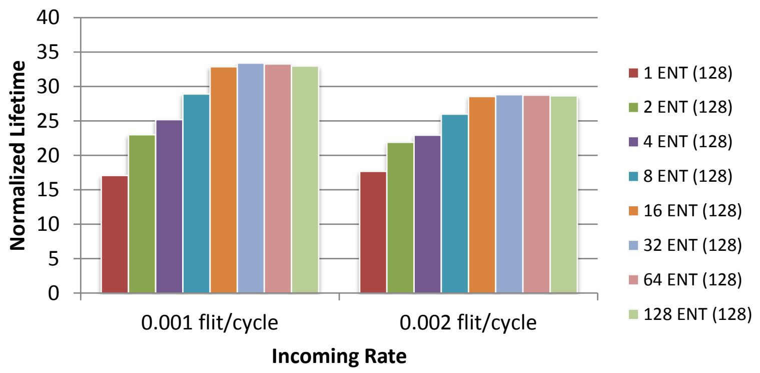


Figure 4.11: Normalized lifetime with varying entry numbers under incoming rate of 0.001 flit/cycle and 0.002 flit/cycle

using the pre-defined random vectors does. Although it balances the duty cycle of individual wires over time, it does not always produce a 120-bit string with 60 bits of "1" and 60 bits of "0". This spatial distribution matters since, for example, a 120-bit vector, which is mostly composed of "0"s, may not balance the duty cycle of wires through the allocator merely losing the opportunity of the lifetime improvement. "LFSR (32)" generates more diverse range of random vectors than "LFSR (128)" because it updates the value 4 times frequently than the other. However, it results in the worse lifetime improvement as it, we suspect, merely generates more bit strings, which do not improve the lifetime of the circuit. Theoretically, the same random number inputs at any generation rate should give the same duty cycle on the wires in the long term. Hence, we expect that the differences among those three will be removed when the experiments run for longer time.

Figure 4.11 illustrates the normalized lifetime when "Random Gen" is implemented with varying number of pre-defined random vectors. We test a range of such numbers at 1, 2, 4, 8, 16, 32, 64 and 128 entries. Here, we update the random number every 128 cycles. As the lifetime difference is observed more clearly at low incoming rate, we test it under relatively low incoming rates such as 0.001 flit/cycle

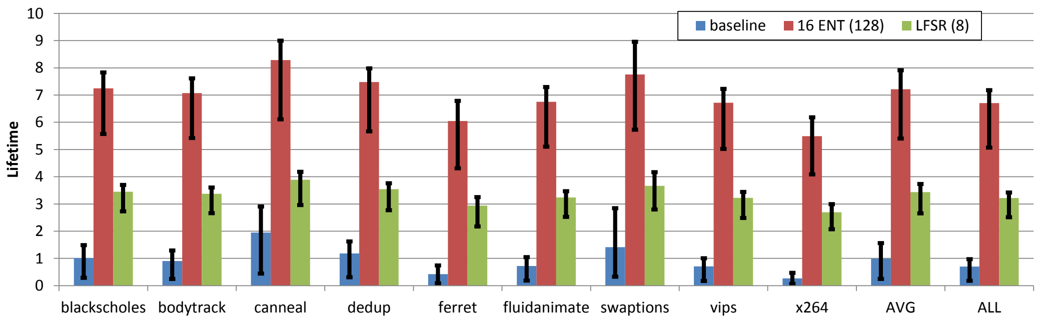


Figure 4.12: Relative lifetime expectation

and 0.002 flit/cycle. Again, the lifetime is normalized to the baseline case under the same incoming rate. In general, the lifetime increases as the number of entries increases. It is shown that only 16 entries are various enough to balance the duty cycle of the wires through the allocator. This implies that implementing the random vector generator using pre-defined vectors does not require extensive additional logics. We will discuss its hardware cost later.

4.6.2.2 Lifetime under PARSEC workloads

Figure 4.13 depicts the normalized lifetime of the network using the proposed technique under the PARSEC workload. The lifetime of the network is estimated by computing the acceleration factor of the router with the minimum incoming rate in the network as it is the most susceptible to the aging effect. The reference system is the baseline router getting the same incoming rate. For each benchmark, we evaluate the normalized lifetime of the exercise logic using the two different random data generation schemes, which are "16 ENT (128)" and "LFSR (8)". We choose "16 ENT (128)" among the pre-defined random vector schemes because it gives the best lifetime with the minimum impact on activity factor, power consumption and area. On the other hand, we use "LFSR (8)" instead of "LFSR (32)" or "LFSR (128)" despite its overhead because in the long term they will result in the similar lifetime extension. The "16 ENT (128)" performs better than the other setting for realistic workloads. It achieves average of about 22× improvement in the network lifetime which can be directly translated into the entire CMP lifetime improvement. The proposed technique performs better when incoming rate is low. As shown in Figure 4.7, "ferret" and "x264" are the applications with the two lowest incoming rates in the PARSEC suite. For those applications, the "16 ENT (128)" achieves 53× and 65× lifetime extension, respectively [2]. Even when the average incoming rate is as high as 0.05 flit/cycle ("canneal") it still achieves the normalized lifetime of 13.8 due to the extreme spread in per-router incoming rates from minimal to maximum seen in that application. The LFSR scheme achieves about half of the lifetime increase achieved by the pre-defined random vectors.

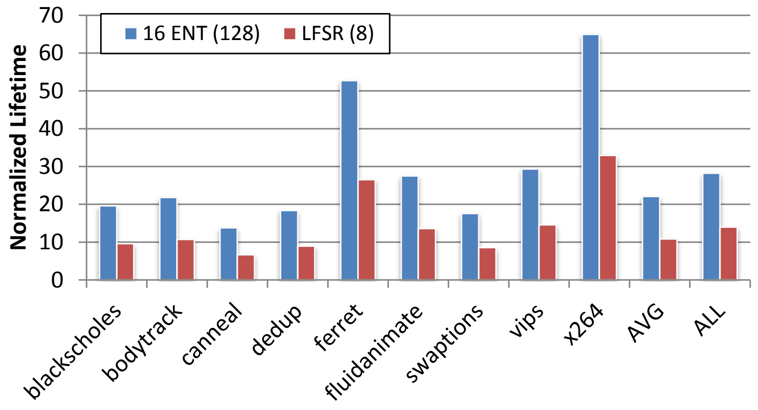


Figure 4.13: Normalized lifetime of the network using the proposed technique under realistic workload

The bars designated as "ALL" denote a case in which the system executes each of the applications sequentially one at a time. In this case, the improvement becomes ∼

[2]It is noted that 65× is greater than the normalized lifetime gain shown in Figure 4.10; this is because the minimum incoming rate of this application is smaller than the minimum value of 0.001 flits/cycle depicted in Figure 4.10.

28×. We found that the execution time for "ferret" and "x264" are the longest among the applications, and hence the incoming rate is dominated by those two applications when the suite is executed in such a way. Although the lifetime improvement achieved by "LFSR (8)" is less than what is achieved by "16 ENT (128)", it is also as high as 14×, on average.

4.6.2.3 Per-router relative lifetime under PARSEC workloads

To estimate the relative lifetime of the router across the PARSEC applications, we compute the acceleration factor of the routers under all PARSEC workloads using a baseline router with 0.02 flits per cycle incoming rate as the reference system. This incoming rate is chosen because it is observed as the average per-router incoming rate across all the PARSEC applications. In Figure 4.12 each bar denotes the lifetime of the routers at the average incoming rate of each program. The line over each bar displays the router lifetimes under *min* and *max* incoming rates of each application. The bars marked by "baseline" show the relative lifetime of the baseline router across the PARSEC suite. As the lifetime, in this graph, is normalized to that of a baseline router at the average incoming rate of all PARSEC benchmarks (0.002 flit/cycle), the "baseline" for "AVG" becomes 1. The lifetime is in inverse proportion to the incoming rate of the program due to the NBTI effect. Hence, routers using "x264" live a shorter time, and routers running "canneal" live a relatively longer time; the latter lives 7.5× longer than the former. The proposed technique performs better at low incoming rates making the lifetime ratio between those two applications to be about 1.5 when either of the two techniques is used. As discussed earlier, the proposed technique levels the lifetime at different workloads. It also reduces the lifetime variance within the network under the same workload. This lifetime variance mainly stems from the variance in spatially distributed network flit incoming rates.

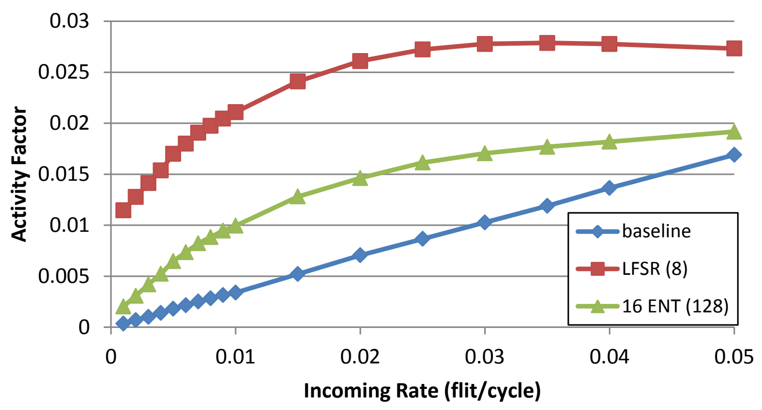


Figure 4.14: Activity factor versus injection rate under various bit-width router models

For example, in case of “swaptions”, originally, the difference between *min* and *max* was found to be 2.5 in relative lifetime; after using the “LFSR (8)” it settees to a 1.4 relative lifetime. Again, it is desirable to equalize the lifetime of the routers within the network as the lifetime of network is decided by the router which stops to operate first.

4.6.2.4 Activity factor

One potential downside of this technique is that it could increase the activity factor along the critical path resulting to potential HCI-induced aging problems. Figure 4.14 shows the average activity factor along the critical paths at various flit incoming rates for different router models. For the “baseline” router, the activity factor is linearly proportional to the incoming rate as the incoming flits are the only stimuli to the allocator. In case of modified routers, the activity factor also increases as the incoming rate grows but it increases more rapidly than the “baseline” case because of the exercise logic. The growth of activity factor with respect to the incoming rate is more rapid at low incoming rates, as the exercise logic has more opportunity to become active. As the incoming rate increases and the exercise logic

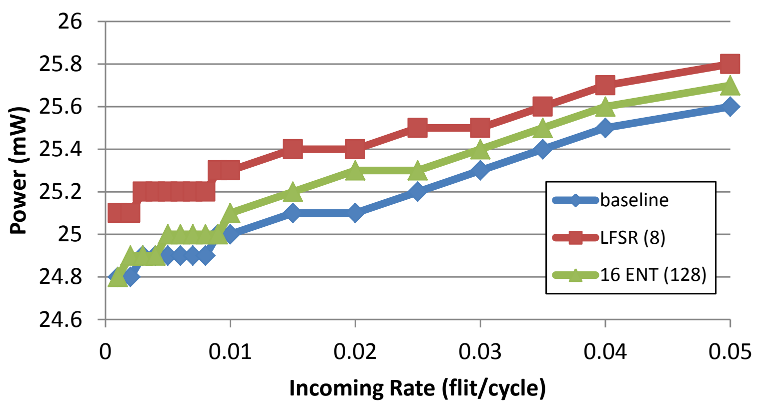


Figure 4.15: Router power consumption under various bit-width router models

misses opportunities to generate a new random vector, the increase in the activity factor slows down. Interestingly, the slope, in case of "LFSR (8)" goes on a plateau where the increase of the activity factor from the increased incoming rate and the decrease of activity factor from losing the chance to activate the exercise logic cancel out each other. As the exercise logic updates the random vector more frequently, the activity factor also increases. Hence, the "LFSR (8)" has higher activity factors than the "16 ENT (128)". Another interesting observation is that the higher activity factor does not necessarily promise the better duty cycle. Although "LFSR (8)" changes the random bit vector more frequently, "16 ENT (128)" balances the duty cycle better even with the slower updating frequency. Although the exercise logic increases the activity factor in general, we can minimize it by increasing the updating period to be even longer than 128 cycles without sacrificing the lifetime gain.

4.6.2.5 Power analysis

The increased activity factor leads to additional power consumption. The extra logic, however, only exercises the allocator logic which merely takes 1.3% of the total power. Our power analysis using PrimeTime indicates that the proposed technique increases total power by less than 1%. Figure 4.15 shows the power consumption with respect to varying incoming rate for the different router models. The router power consumption increases as the incoming rate increases, however, it does not increase as much as the incoming rate does. For example, when the incoming rate increases from 0.001 flits per cycle to 0.05 flits per cycle (50×), the power consumption merely increases by 3.2%. We observed that the power consumption is dominated by the clock distribution, which makes the power consumption less sensitive to any other activity. Hence, even “LFSR (8)” merely increases the power consumption of the allocator alone by 7% despite the relatively significant increase in activity factor shown in Figure 4.14. “16 ENT (128)” does not affect the activity factor of the allocator as much as “LFSR (8)” resulting in 0.3% increase of the allocator power consumption. The increase of the power consumption mostly comes from the additional logic gates which increase the area by less than 1% of the area of original router.

4.7 Related Work

Aging models for transistors have been extensively studied since technology crossed the submicron border, which inherently made the CMOS manufacturing process vulnerable to run-time faults. NBTI and HCI are dominant wearout effects and have thus been more intensively studied [61]. Conventionally, aging effects are studied under DC stress conditions where it is easier to measure transistor parameters [41, 51, 56]. However, AC stress conditions are more realistic for high-frequency long-term CMOS operation; hence works such as those of [45, 79, 61] discuss such long-term models, while other works introduce the relationship between DC and AC stress conditions [80].

The newly emerging devices, such as multi-gate field effect transistors MuFETs and FinFETs [5, 70] at 22 nm process technology and below, have gradually become an object for aging effects exploration. Wang *et al.* [80] make an attempt towards presenting a unified aging model for the effects of both HCI and NBTI. They derive models for double- and triple-gate FinFETs, for each of these aging effects, that capture specific FinFET geometrical aspects.

Unfortunately, the vast majority of the reported models lack important details, such as values for various constants, measurement conditions, detailed explanation of parameters, etc., so it becomes fairly challenging to employ the existing proposed frameworks in the context of microarchitecture, and reproduce further models that will lead to meaningful aging effect calculations, and thereafter important device or components wearout models and conclusions.

PBTI (positive voltage temperature instability) affects NMOS transistors, as well. It occurs when a positive voltage is applied to the gate of NMOS, and the physics behind NBTI on PMOS transistors and PBTI on NMOS ones are basically the same. It has been widely believed that PBTI degradation in NMOS transistors is practically negligible when compared to NBTI in PMOS transistors [69]. However, state-of-the-art experiments [82] have shown that PBTI can dominate under certain circumstances. Although we focus on NBTI, the proposed technique still mitigates the PBTI effect as it eliminates the duty cycle imbalance, which accelerates PBTI.

Various techniques have been proposed so as to mitigate the aging effect in processor core architectures. Among those proposed mechanisms, Gunadi *et al.* [31] suggested the Colt duty cycle equalizer which balances duty cycle by alternating true and one's complement data representations. Abella *et al.* [1] introduced the Penelope NBTI-aware processor architecture where they discuss a number of techniques to combat NBTI, including a mechanism which writes special values in memory cells in order to keep the duty cycle at an ideal 50%. Next, Kumar *et al.* [49] proposed periodic cache flipping so as to provide periods of relaxation for the influenced pMOSFET allowing dynamic recovery of threshold voltage. The aforementioned three works bear similarities to the "exercise logic" proposed in this paper, in the context that these techniques periodically insert inverted or random values to balance duty cycle. However none of the three has dealt with NoC router micro-architectures, which as we discuss are critical to the system's survivability.

Although our approach is similar to the approaches in the aforementioned previous works, here we actually handle a different problem. In the previous works, the researchers focused on the duty cycle bias on the data paths, while we mainly balance the uneven duty cycles along the control paths. In fact, the sources of the uneven duty cycle are different. In the previous works, it is dominant in biased data contents, while in our work it is more evident during the proved extremely low activity seen in NoC routers (see Section 4.3). Although the proposed "exercise logic" is also able to resolve the aging problems due to the skewed data content along the data path of the NoC routers, we utilize it only for the timing critical paths where the NBTI impact occurs in its most critical form or impact.

Aging has been also examined in the NoC domain. Bhardwaj *et al.* [9, 8] propose routing algorithms to mitigate multiple aging mechanisms. They also point out that NBTI plays a major role in NoC router aging, and their routing techniques balance the traffic load across the network to level-out the aging rates among the routers. The approach is reasonable, in that they force the network traffic detour through routers of low utilization which, on the contrary, accelerates NBTI-caused aging. However, they use these routing techniques for the opposite reason. The routing algorithms are actually designed to reduce the workload onto the routers which exhibit high utilization, which as we show here are not actually the routers likely to exhibit the most stress-related aging. Fu *et al.* [26] propose a similar technique to ours, in that it inserts special values to idle arbiters to mitigate NBTI. However, they propose this technique to make arbiters less frequently utilized so as to give these routers a chance to recover from the effects of NBTI, which is actually not necessary applicable to frequently utilized circuits. In these previous studies, it is assumed that the NBTI stress time is proportional to the router utilization, however, on the contrary, we prove that this is not the actual case. Through detailed, gate-level analysis, not found in earlier works, we demonstrated that the duty cycle becomes more skewed when the NoC router is actually under-utilized and not when it is high- or over-utilized.

4.8 Conclusions

Prolonged operational stress gives rise to accelerated wearout and failure, due to HCI and NBTI. Each failure mechanism correlates with different usage-based stresses, all of which can eventually generate permanent faults. While the wearout of an individual core in many core CMPs may not necessarily be catastrophic for the entire parallel-processing system, a single fault in the inter-processor Network-on-Chip (NoC) fabric could render the entire chip useless. In this paper, we have developed critical path models for HCI- and NBTI-induced wear due to stresses caused by realistic workloads, and apply them onto the interconnect microarchitecture. A key finding from this modeling being that, counter to prevailing wisdom, wearout in the CMP on-chip interconnect is correlated with lack of load observed in the NoC routers, rather than high load. We then develop a novel wearout-decelerating scheme in which routers under low load have their wearout-sensitive components exercised, without significantly impacting cycle time, pipeline depth, area or power consumption of the overall router. We subsequently show that the proposed design yields a 13.8×-65× increase in CMP lifetime.

5. CONCLUSIONS

Moore's Law scaling is continuing to yield even higher transistor density with each succeeding process generation, leading to today's multi-core Chip Multi-Processors with tens or even hundreds of interconnected cores or tiles. In such systems, the interconnect, typically organized as a Network-on-Chip, plays a vital role enabling communication among the on-chip components. However, the power consumption of NoC has become a growing concern as the complexity of NoC increases and the power consumption of the individual transistor stops to scale unlike the feature size. In addition, deep sub-micron CMOS process technology is marred by increasing susceptibility to wearout shortening the useful lifespan of such systems. Hence, energy efficiency and reliability must be first considered while designing such on-chip systems. In this dissertation, I have proposed two energy saving techniques to improve the energy efficiency of NoC and LLC. I have also examined the cause of wearout in NoC and introduced a novel wear-decelerating router microarchitecture overcoming the aging effects.

It was observed that there has been energy waste due to transference of unused cache-line words, and I proposed a energy saving router microarchitecture leveraging those unused words. It was observed that only 40% of words in fetched cache lines are actually used, and 60% of words consume energy being transferred over the interconnect for nothing. The pattern of used and unused words in cache lines was predictable as a certain fields of data structures are repeatedly used within program iterations. A spatial locality predictor which speculates the unused words in cache lines and a new packet encoding scheme are proposed to encode those unused words such that they do not cause any bit transition along the interconnect data-path.

Using this technique, an average of 35% dynamic energy reduction has been achieved at negligible performance loss.

I also conducted an experiment with a new DVFS policy to reduce uncore energy consumption. DVFS has been introduced to reduce energy consumption when system is under-loaded. It has been widely studied and used in processor core design. However, employing DVFS in uncore design introduces new challenges. It is necessary to find a metric which is controllable by configuring uncore V/F level, and has direct impact on the overall system performance. AMAT has been used as it has some sensitivity to the V/F level of uncore and some impact on the overall system performance to some extent. However, it is hard to define the reference point in terms of AMAT. I proposed to use throughput of uncore as the metric as it is dependent upon the V/F level of uncore and also it affects the overall performance. It has been examined that such throughput based control scheme results in improved energy efficiency compared to the AMAT based control.

I also addressed reliability issues in CMP design in this dissertation. With the continuous down-scaling of process technologies, reliability has become an important concern in current and future CMOS logic circuit design. Wear of NoC has more critical impact on the lifetime of CMP than wear of core does because even a single failure of a link or a router can incapacitate the entire system. The key contribution of this work is the discovery of the relation between wearout and workload observed in the NoC routers. Counter to the prevailing wisdom, wear-out in NoC is accelerated when routers are under-utilized rather than highly- or over-utilized. A novel anti-aging router microarchitecture to overcome the failure mechanisms is also introduced in this work.

REFERENCES

[1] J. Abella, X. Vera, and A. Gonzalez, “Penelope: The NBTI-Aware Processor,” in *Proceedings of the 40th Annual IEEE/ACM international symposium on Microarchitecture*, ser. MICRO-40. Washington, DC, USA: IEEE Computer Society, 2007, pp. 85–96.

[2] S. G. Abraham, R. A. Sugumar, D. Windheiser, B. R. Rau, and R. Gupta, “Predictability of Load/Store Instruction Latencies,” in *Proceedings of the 26th annual international symposium on Microarchitecture*, ser. MICRO-26. Washington, DC, USA: IEEE Computer Society, 1993, pp. 139–152.

[3] Advanced Micro Devices (AMD) Inc., “AMD Opteron Processors for Servers: AMD64-Based Server Solutions for x86 Computing,” http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_8796,00.html.

[4] M. Agarwal, B. Paul, M. Zhang, and S. Mitra, “Circuit Failure Prediction and Its Application to Transistor Aging,” in *Proceedings of the 25th IEEE VLSI Test Symposium*. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 277–286.

[5] C. Auth, “22-nm Fully-Depleted Tri-gate CMOS Transistors,” in *Proceedings of the Custom Integrated Circuits Conference*, ser. CICC. Washington, DC, USA: IEEE Computer Society, 2012, pp. 1–6.

[6] A. Banerjee, R. Mullins, and S. Moore, “A Power and Energy Exploration of Network-on-Chip Architectures,” in *Proceedings of the 1st international sympo-*

sium on Networks-on-Chip, ser. NOCS. New York, NY, USA: ACM, 2007, pp. 163–172.

[7] D. U. Becker, "Efficient Microarchitecture for Network-on-Chip Routers," Ph.D. dissertation, Stanford University, 2012.

[8] K. Bhardwaj, K. Chakraborty, and S. Roy, "An MILP-based Aging-aware Routing Algorithm for NoCs," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE. San Jose, CA, USA: EDA Consortium, 2012, pp. 326–331.

[9] ——, "Towards Graceful Aging Degradation in NoCs through an Adaptive Routing Algorithm," in *Proceedings of the 49th Annual Design Automation Conference*, ser. DAC. New York, NY, USA: ACM, 2012, pp. 382–391.

[10] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, ser. PACT. New York, NY, USA: ACM, 2008, pp. 72–81.

[11] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The Gem5 Simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, 2011.

[12] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt, "The M5 Simulator: Modeling Networked Systems," *Micro, IEEE*, vol. 26, no. 4, pp. 52–60, 2006.

[13] J. Blome, S. Feng, S. Gupta, and S. Mahlke, "Self-calibrating Online Wearout Detection," in *Proceedings of the 40th Annual IEEE/ACM international sym-*

posium on Microarchitecture, ser. MICRO-40. Washington, DC, USA: IEEE Computer Society, 2007.

[14] P. Bogdan, R. Marculescu, S. Jain, and R. T. Gavila, "An Optimal Control Approach to Power Management for Multi-Voltage and Frequency Islands Multiprocessor Platforms under Highly Variable Workloads," in *Proceedings of the 2012 IEEE/ACM Sixth international symposium on Networks-on-Chip*, ser. NOCS. Washington, DC, USA: IEEE Computer Society, 2012, pp. 35–42.

[15] B. Calder, C. Krintz, S. John, and T. Austin, "Cache-Conscious Data Placement," in *Proceedings of the eighth international conference on Architectural support for programming languages and operating systems*, ser. ASPLOS VIII. New York, NY, USA: ACM, 1998, pp. 139–149.

[16] S. Carr, K. S. McKinley, and C.-W. Tseng, "Compiler Optimizations for Improving Data Locality," *SIGPLAN Not.*, vol. 29, no. 11, pp. 252–262, Nov. 1994.

[17] C. F. Chen, S. hyun Yang, and B. Falsafi, "Accurate and Complexity-Effective Spatial Pattern Prediction," in *Proceedings of The 10th international symposium on High-Performance Computer Architecture*, ser. HPCA-10. Washington, DC, USA: IEEE Computer Society, 2004.

[18] X. Chen, Z. Xu, H. Kim, P. Gratz, J. Hu, M. Kishinevsky, and U. Ogras, "In-network Monitoring and Control Policy for DVFS of CMP Networks-on-Chip and Last Level Caches," in *Proceedings of the 2012 IEEE/ACM Sixth international symposium on Networks-on-Chip*, ser. NOCS. Washington, DC, USA: IEEE Computer Society, 2012, pp. 43–50.

[19] T. Chilimbi, M. Hill, and J. Larus, "Making Pointer-Based Data Structures Cache Conscious," *Computer*, vol. 33, no. 12, pp. 67–74, 2000.

[20] T. M. Chilimbi, B. Davidson, and J. R. Larus, "Cache-Conscious Structure Definition," *SIGPLAN Not.*, vol. 34, no. 5, pp. 13–24, May 1999.

[21] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," in *Proceedings of Design Automation Conference*, ser. DAC. New York, NY, USA: ACM, 2001, pp. 684–689.

[22] R. Das, S. Eachempati, A. Mishra, V. Narayanan, and C. Das, "Design and Evaluation of a Hierarchical On-Chip Interconnect for Next-Generation CMPs," in *Proceedings of The 15th international symposium on High-Performance Computer Architecture*, ser. HPCA-15. Washington, DC, USA: IEEE Computer Society, 2009, pp. 175–186.

[23] R. Das, A. Mishra, C. Nicopoulos, D. Park, V. Narayanan, R. Iyer, M. Yousif, and C. Das, "Performance and Power Optimization through Data Compression in Network-on-Chip Architectures," in *Proceedings of The 14th international symposium on High-Performance Computer Architecture*, ser. HPCA-14. Washington, DC, USA: IEEE Computer Society, 2008, pp. 215–225.

[24] A. DeOrio, K. Aisopos, V. Bertacco, and L.-S. Peh, "DRAIN: Distributed Recovery Architecture for Inaccessible Nodes in multi-core chips," in *Proceedings of the 48th ACM/EDAC/IEEE Design Automation Conference*, ser. DAC. New York, NY, USA: ACM, 2011, pp. 912–917.

[25] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A Highly Resilient Routing Algorithm for Fault-tolerant NoCs," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE, 2009, pp. 21–26.

[26] X. Fu, T. Li, and J. A. B. Fortes, "Architecting Reliable Multi-Core Network-on-Chip for Small Scale Processing Technology," in *Proceedings of the 42nd Annual IEEE/IFIP international conference on Dependable Systems and Networks*, ser. DSN, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2010, pp. 111–120.

[27] M. Gebhart, J. Hestness, E. Fatehi, P. Gratz, and S. W. Keckler, "Running PARSEC 2.1 on M5," The Univ. of Texas at Austin, Dept. of Comp. Sci., Tech. Rep., 2009.

[28] P. Gratz, C. Kim, R. McDonald, S. Keckler, and D. Burger, "Implementation and Evaluation of On-Chip Network Architectures," in *Proceedings of the International Conference on Computer Design*, ser. ICCD. Washington, DC, USA: IEEE Computer Society, 2006, pp. 477–484.

[29] P. Gratz, K. Sankaralingam, H. Hanson, P. Shivakumar, R. McDonald, S. Keckler, and D. Burger, "Implementation and Evaluation of a Dynamically Routed Processor Operand Network," in *Proceedings of the 1st international symposium on Networks-on-Chip*, ser. NOCS. New York, NY, USA: ACM, 2007, pp. 7–17.

[30] L. Guang, E. Nigussie, L. Koskinen, and H. Tenhunen, "Autonomous DVFS on Supply Islands for Energy-Constrained NoC Communication," in *Proceedings of the 22nd international conference on Architecture of Computing Systems*, ser. ARCS. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 183–194.

[31] E. Gunadi, A. A. Sinkar, N. S. Kim, and M. H. Lipasti, "Combating Aging with the Colt Duty Cycle Equalizer," in *Proceedings of the 43rd Annual IEEE/ACM international symposium on Microarchitecture*, ser. MICRO-43. Washington, DC, USA: IEEE Computer Society, 2010.

[32] K. C. Hale, B. Grot, and S. W. Keckler, "Segment Gating for Static Energy Reduction in Networks-on-Chip," in *Proceedings of the 2nd international workshop on Network on Chip Architectures*, ser. NoCArc. New York, NY, USA: ACM, 2009, pp. 57–62.

[33] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[34] J. Hestness and S. W. Keckler, "Netrace: Dependency-Tracking Traces for Efficient Network-on-Chip Experimentation," The Univ. of Texas at Austin, Dept. of Comp. Sci., Tech. Rep., 2011.

[35] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz Mesh Interconnect for a Teraflops Processor," *Micro, IEEE*, vol. 27, no. 5, pp. 51–61, 2007.

[36] J. Howard, S. Dighe, S. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, and R. Van Der Wijngaart, "A 48-Core IA-32 Processor in 45 nm CMOS Using On-Die Message-Passing and DVFS for Performance and Power Scaling," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 1, pp. 173–183, 2011.

[37] L. Huang and Q. Xu, "AgeSim: A Simulation Framework for Evaluating the Lifetime Reliability of Processor-based SoCs," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2010, pp. 51–56.

[38] Intel, "Intel Atom Processor Z510," http://ark.intel.com/Product.aspx?id=35469 &processor=Z510&spec-codes=SLB2C.

[39] International Technology Roadmap for Semiconductors (ITRS) Working Group, "International Technology Roadmap for Semiconductors (ITRS), 2009 Edition," http://www.itrs.net/Links/2009ITRS/Home2009.htm.

[40] E. Jacobsen, E. Rotenberg, and J. E. Smith, "Assigning Confidence to Conditional Branch Predictions," in *Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture*, ser. MICRO-29. Washington, DC, USA: IEEE Computer Society, 1996, pp. 142–152.

[41] JEDEC Solid State Technology Association, "Failure mechanisms and models for semiconductor devices, JEP122G," 2011.

[42] Y. Jin, K. H. Yum, and E. J. Kim, "Adaptive Data Compression for High-Performance Low-Power On-Chip Networks," in *Proceedings of the 41st IEEE/ACM international symposium on Microarchitecture*, ser. MICRO-41. Washington, DC, USA: IEEE Computer Society, 2008, pp. 354–363.

[43] Jon Stokes, "IBM's 8-core POWER7: Twice The Muscle, Half The Transistors," http://arstechnica.com/hardware/news/2009/09/ibms-8-core-power7-twice-the-muscle-half-the-transistors.ars.

[44] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE, 2009, pp. 423–428.

[45] U. R. Karpuzcu, B. Greskamp, and J. Torrellas, "The BubbleWrap Many-core: Popping Cores for Sequential Acceleration," in *Proceedings of the 42nd Annual IEEE/ACM international symposium on Microarchitecture*, ser. MICRO-42. New York, NY, USA: ACM, 2009, pp. 447–458.

[46] C. Kim, D. Burger, and S. W. Keckler, "An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches," in *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, ser. ASPLOS X. New York, NY, USA: ACM, 2002, pp. 211–222.

[47] H. Kim, P. Ghoshal, B. Grot, P. V. Gratz, and D. A. Jimènez, "Reducing Network-on-Chip Energy Consumption Through Spatial Locality Speculation," in *Proceedings of the 5th ACM/IEEE international symposium on Networks-on-Chip*, ser. NOCS. New York, NY, USA: ACM, 2011, pp. 233–240.

[48] K. Kuhn, "Reducing Variation in Advanced Logic Technologies: Approaches to Process and Design for Manufacturability of Nanoscale CMOS," in *Proceedings of IEEE International Electron Devices Meeting*, ser. IEDM. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 471–474.

[49] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Impact of NBTI on SRAM Read Stability and Design for Reliability," in *Proceedings of the 7th international symposium on Quality Electronic Design*, ser. ISQED. Washington, DC, USA: IEEE Computer Society, 2006, pp. 210–218.

[50] A.-C. Lai, C. Fide, and B. Falsafi, "Dead-block prediction & dead-block correlating prefetchers," *SIGARCH Comput. Archit. News*, vol. 29, no. 2, 2001.

[51] X. Li, J. Qin, and J. Bernstein, "Compact Modeling of MOSFET Wearout Mechanisms for Circuit-Reliability Simulation," *Device and Materials Reliability, IEEE Transactions on*, vol. 8, no. 1, 2008.

[52] Y. Li, S. Makar, and S. Mitra, "CASP: Concurrent Autonomous Chip Self-test using Stored Test Patterns," in *Proceedings of the conference on Design,*

automation and test in Europe, ser. DATE. New York, NY, USA: ACM, 2008, pp. 885–890.

[53] G. Liang and A. Jantsch, "Adaptive Power Management for the On-Chip Communication Network," in *Proceedings of the 9th EUROMICRO Conference on Digital System Design*, ser. DSD. Washington, DC, USA: IEEE Computer Society, 2006, pp. 649–656.

[54] J. S. Liptay, "Structural Aspects of the System/360 Model 85, II: The cache," *IBM Systems Journal*, vol. 7, no. 1, pp. 15–21, 1968.

[55] Y. Lu, L. Shang, H. Zhou, H. Zhu, F. Yang, and X. Zeng, "Statistical Reliability Analysis under Process Variation and Aging Effects," in *Proceedings of the 46th Annual Design Automation Conference*, ser. DAC. New York, NY, USA: ACM, 2009, pp. 514–519.

[56] E. Maricau and G. Gielen, "A Methodology for Measuring Transistor Ageing Effects towards Accurate Reliability Simulation," in *Proceedings of the 15th IEEE International On-Line Testing Symposium*, ser. IOLTS. Washington, DC, USA: IEEE Computer Society, 2009, pp. 21–26.

[57] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das, "A Case for Dynamic Frequency Tuning in On-Chip Networks," in *Proceedings of the 42nd Annual IEEE/ACM international symposium on Microarchitecture*, ser. MICRO-42. New York, NY, USA: ACM, 2009, pp. 292–303.

[58] D. Molka, D. Hackenberg, R. Schone, and M. S. Muller, "Memory Performance and Cache Coherency Effects on an Intel Nehalem Multiprocessor System," in *Proceedings of the 18th international conference on Parallel Architectures and*

Compilation Techniques, ser. PACT. New York, NY, USA: ACM, 2009, pp. 261–270.

[59] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0," in *Proceedings of the 40th Annual IEEE/ACM international symposium on Microarchitecture*, ser. MICRO-40. Washington, DC, USA: IEEE Computer Society, 2007.

[60] S. Nassif, K. Bernstein, D. Frank, A. Gattiker, W. Haensch, B. Ji, E. Nowak, D. Pearson, and N. Rohrer, "High Performance CMOS Variability in the 65nm Regime and Beyond," in *Proceedings of IEEE International Electron Devices Meeting*, ser. IEDM. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 569–571.

[61] F. Oboril and M. Tahoori, "ExtraTime: Modeling and Analysis of Wearout due to Transistor Aging at Microarchitecture-level," in *Proceedings of the 42nd Annual IEEE/IFIP international conference on Dependable Systems and Networks*, ser. DSN. Los Alamitos, CA, USA: IEEE Computer Society, 2012, pp. 1–12.

[62] U. Y. Ogras, R. Marculescu, and D. Marculescu, "Variation-Adaptive Feedback Control for Networks-on-Chip with Multiple Clock Domains," in *Proceedings of the 45th annual Design Automation Conference*, ser. DAC. New York, NY, USA: ACM, 2008, pp. 614–619.

[63] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. Das, "Exploring Fault-Tolerant Network-on-Chip Architectures," in *Proceedings of international conference on Dependable Systems and Networks*, ser. DSN. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 93–104.

[64] L.-S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," in *Proceedings of the 7th international symposium on High-Performance Computer Architecture*, ser. HPCA-7. Washington, DC, USA: IEEE Computer Society, 2001, pp. 255–.

[65] M. D. Powell, A. Biswas, S. Gupta, and S. S. Mukherjee, "Architectural Core Salvaging in a Multi-core Processor for Hard-error Tolerance," in *Proceedings of the 36th annual international symposium on Computer architecture*, ser. ISCA. New York, NY, USA: ACM, 2009, pp. 93–104.

[66] P. Pujara and A. Aggarwal, "Cache Noise Prediction," *Computers, IEEE Transactions on*, vol. 57, no. 10, pp. 1372–1386, 2008.

[67] M. K. Qureshi, M. A. Suleman, and Y. N. Patt, "Line Distillation: Increasing Cache Capacity by Filtering Unused Words in Cache Lines," *High-Performance Computer Architecture, international symposium on*, vol. 0, 2007.

[68] A. Rahimi, M. E. Salehi, S. Mohammadi, and S. M. Fakhraie, "Low-Energy GALS NoC with FIFO-Monitoring Dynamic Voltage Scaling," *Microelectron. J.*, vol. 42, no. 6, pp. 889–896, Jun. 2011.

[69] V. Reddy, A. Krishnan, A. Marshall, J. Rodriguez, S. Natarajan, T. Rost, and S. Krishnan, "Impact of Negative Bias Temperature Instability on Digital Circuit Reliability," in *Proceedings of the 2012 IEEE International Reliability Physics Symposium*, ser. IRPS. Washington, DC, USA: IEEE Computer Society, 2002, pp. 248–254.

[70] M. Saitoh, K. Ota, C. Tanaka, Y. Nakabayashi, K. Uchida, and T. Numata, "Performance, Variability and Reliability of Silicon Tri-gate Nanowire MOS-

FETs," in *Proceedings of the 2012 IEEE International Reliability Physics Symposium*, ser. IRPS. Washington, DC, USA: IEEE Computer Society, 2012, pp. 6A.3.1–6A.3.6.

[71] T. Sakurai and A. Newton, "Alpha-Power Law MOSFET Model and Its Applications to CMOS Inverter Delay and Other Formulas," *Solid-State Circuits, IEEE Journal of*, vol. 25, no. 2, pp. 584–594, 1990.

[72] D. Schinkel, E. Mensink, E. Klumperink, E. van Tuijl, and B. Nauta, "Low-Power, High-Speed Transceivers for Network-on-Chip Communication," *IEEE Transactions on VLSI Systems*, vol. 17, no. 1, pp. 12–21, 2009.

[73] T. Schonwald, J. Zimmermann, O. Bringmann, and W. Rosenstiel, "Fully Adaptive Fault-Tolerant Routing Algorithm for Network-on-Chip Architectures," in *Proceedings of the 10th EUROMICRO Conference on Digital System Design*, ser. DSD. Washington, DC, USA: IEEE Computer Society, 2007, pp. 527–534.

[74] L. Shang, L. Peh, and N. K. Jha, "Power-Efficient Interconnection Networks: Dynamic Voltage Scaling with Links," *IEEE Computer Architecture Letters*, vol. 1, no. 1, 2002.

[75] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks," in *Proceedings of the 9th international symposium on High-Performance Computer Architecture*, ser. HPCA-9. Washington, DC, USA: IEEE Computer Society, 2003, pp. 91–.

[76] J. C. Smolens, B. T. Gold, J. C. Hoe, B. Falsafi, and K. Mai, "Detecting Emerging Wearout Faults," in *Proceedings of Workshop on SELSE*, 2007.

[77] S. W. Son, K. Malkowski, G. Chen, M. Kandemir, and P. Raghavan, “Integrated Link/CPU Voltage Scaling for Reducing Energy Consumption of Parallel Sparse Matrix Applications,” in *Proceedings of the 20th international conference on Parallel and distributed processing*, ser. IPDPS’06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 297–297.

[78] M. Taylor, M. B. Taylor, W. Lee, S. Amarasinghe, and A. Agarwal, “Scalar Operand Networks: On-chip Interconnect for ILP in Partitioned Architectures,” in *Proceedings of The 9th international symposium on High-Performance Computer Architecture*, ser. HPCA-9. Washington, DC, USA: IEEE Computer Society, 2003, pp. 341–353.

[79] B. Tudor, J. Wang, Z. Chen, R. Tan, W. Liu, and F. Lee, “An Accurate and Scalable MOSFET Aging Model for Circuit Simulation,” in *Proceedings of the 12th international symposium on Quality Electronic Design*, ser. ISQED. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1–4.

[80] Y. Wang, S. Cotofana, and L. Fang, “A Unified Aging Model of NBTI and HCI Degradation Towards Lifetime Reliability Management for Nanoscale MOSFET Circuits,” in *Proceedings of the 2011 IEEE/ACM international symposium on Nanoscale Architectures*, ser. NANOARCH. Washington, DC, USA: IEEE Computer Society, 2011, pp. 175–180.

[81] D. H. Yoon, M. K. Jeong, and M. Erez, “Adaptive Granularity Memory Systems: A Tradeoff between Storage Efficiency and Throughput,” in *Proceedings of the 38th annual international symposium on Computer architecture*, ser. ISCA. New York, NY, USA: ACM, 2011, pp. 295–306.

[82] S. Zafar, Y.-H. Kim, V. Narayanan, C. Cabral, V. Paruchuri, B. Doris, J. Stathis, A. Callegari, and M. Chudzik, "A Comparative Study of NBTI and PBTI (Charge Trapping) in SiO2/HfO2 Stacks with FUSI, TiN, Re Gates," in *Proceedings of the 2006 Symposium on VLSI Technology. Digest of Technical Papers.*, 2006, pp. 23–25.

[83] H. Zhang, V. George, and J. M. Rabaey, "Low-Swing On-Chip Signaling Techniques: Effectiveness and Robustness," *IEEE TRANSACTIONS ON VLSI SYSTEMS*, vol. 8, no. 3, pp. 264–272, 2000.

[84] Z. Zhang, A. Greiner, and S. Taktak, "A Reconfigurable Routing Algorithm for a Fault-tolerant 2D-Mesh Network-on-Chip," in *Proceedings of the 45th ACM/IEEE Design Automation Conference*, ser. DAC. New York, NY, USA: ACM, 2008, pp. 441–446.