

# Leveraging Unused Cache Block Words to Reduce Power in CMP Interconnect

Hyungjun Kim and Paul V. Gratz

Department of Electrical and Computer Engineering, Texas A&M University  
{hyungjuk, pgratz}@tamu.edu

**Abstract**—Power is of paramount importance in modern computer system design. In particular, the cache interconnect in future CMP designs is projected to consume up to half of the system power for cache fills and spills [8]. Despite the power consumed by spills and fills, a significant percentage of each cache line is unused prior to eviction from the cache. If unused cache block words can be identified, this information can be used to improve CMP interconnect power and energy consumption. We propose a new method of CMP interconnect packet composition, leveraging unused data to reduce power. These methods are well suited to interconnection networks with high-bandwidth wires, and do not require expensive multi-ported memory systems. Assuming perfect prediction, our techniques achieve an average of  $\sim 37\%$  savings in total dynamic link power consumption. With our current best prediction mechanism, our techniques reduce dynamic power consumption by  $\sim 23\%$  on average.

**Index Terms**—Multicore, NoC, dynamic power, flit encoding, memory system

## 1 INTRODUCTION

As the core count in chip-multiprocessor (CMP) systems increases, networks-on-chip (NoCs) present a scalable alternative to traditional, bus-based designs. NoCs must be carefully designed to meet design constraints on area, bandwidth and latency. As in most current VLSI designs, power efficiency has also become a first-order constraint in NoC design. The energy consumed by the NoC itself is 28% of the tile power in the Intel Teraflop chip [4] and 36% of the total chip power in MIT RAW chip [11]. The NoC packet datapath, consisting of the link, crossbar and FIFOs, in consumes a significant portion of interconnect power, up to 55% of total network power in the Intel Teraflop chip [4].

Existing NoCs implement channels with relatively large link bit-widths (128-256 bits), 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. Often, much of the cache block ends up not accessed at all. Pujara et al. show that only 57% of words in each cache line are actually used by the processor and the pattern of word usage is quite predictable [9]. Their prediction mechanism achieves an accuracy of 95% for SPEC 2K benchmarks. As we will describe, we propose to leverage unused words of the block transfers between the lower and upper cache levels to save energy.

Useless words also reside in dirty cache blocks. Isen et al. proposed a novel method of energy saving with the help of semantic information from the program such as operating systems [5]. They identify words in a cache line which had been dynamically allocated to programs but do not contain any useful data yet or are freed by the program thus there is no useful data any longer. They leverage those *inconsequential* data to save energy by reducing the number of accesses to the DRAM resulting in the reduction in the number of pre-charge discharge cycles as well as more opportunity for the DRAM to enter power saving mode.

Minimizing data transfer among multiple caches has also been explored in the literature. A sectored cache was proposed

Manuscript submitted: 18-Mar-2010. Manuscript accepted: 02-May-2010.  
Final manuscript received: 09-May-2010.

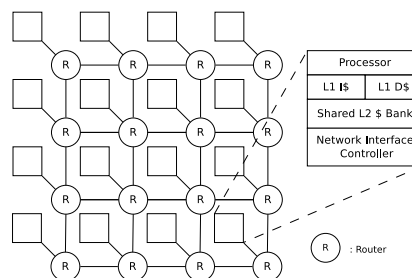


Fig. 1: General CMP Architecture

to reduce the data transmitted for larger cache design while keeping the overhead minimized [10]. With the sector cache, only a portion of the block (or sector) is fetched, significantly reducing both the miss time and the bus traffic. Jin et al. proposed a compression mechanism to reduce the amount of data transfer due to frequent values in a cache line [6].

In this paper, we introduce a new method of packet composition for lower power CMP interconnect. The strategy of our scheme is to first identify useful words in cache lines. Once we know which words will be useful, we use this information to compose and encode the CMP interconnect packets such that power and network occupancy is reduced, leveraging the activity factor in dynamic power. Our ultimate goal is to reduce dynamic power of the NoC datapath of CMP connecting NoCs by actually and virtually reducing the amount of data transmission.

## 2 BACKGROUND

### 2.1 Dynamic Power Consumption

When a bit is transmitted over interconnect wire, dynamic power is consumed when the signal toggles between logical "0" and "1". Dynamic power of links is mainly caused by charging and discharging of capacitive loads. Hence, the less frequently the wires change polarity, the less dynamic energy dissipation. For clarity, here is the CMOS dynamic power equation:

$$P = \alpha \cdot C \cdot V^2 \cdot F \quad (1)$$

where  $P$  is the power consumed,  $C$  is the switched capacitance,  $V$  is the supplied voltage, and  $F$  is the clock frequency.  $\alpha$  is the activity factor, representing the probability of bit transition on

the wire, and is the focus of this work. Leaving the other values such as C, V and F as they are, reduction of  $\alpha$  linearly reduces the dynamic energy consumed.

We propose two complementary means to reduce  $\alpha$  in CMP interconnect, and directly reduce dynamic energy: 1) Remove data (flits) from NoC packets (*flit-drop*); or 2) Keep as many of interconnect wires at the same polarity during flit traversal (*word-repeat*). For example, say two bits are sent over a wire and the second bit is predicted unused. Our *flit-drop* scheme would drop the second bit to reduce the number of bits transmitted over the wire. Our *word-repeat* scheme makes the second bit the same polarity as the first one to virtually cut down the number of bits in terms of dynamic power consumption.

## 2.2 Baseline Architecture

Figure 1 depicts the general CMP/NoC architecture. 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 interconnection network. Interconnection network is composed of routers and interconnection wires which provide connections between routers. When the L1 cache needs to communicate with an L2 cache bank in another tile, it sends a request to the NIC. The NIC composes a packet and injects it into the interconnection network. It is also responsible for extracting a request from an incoming packet and forwarding it to the local L1 or L2 cache bank.

Our baseline architecture employs a  $4 \times 4$  mesh topology where deterministic routing and wormhole switching are used. Each processor has a 64KB data and a 32KB instruction L1 cache, both 2-way-set-associative. The shared L2 cache has an 8MB capacity and is organized as an 8-way-set-associative NUCA cache, where each processor node contains a shared L2 cache bank [7]. In each case, a cache line is 64B wide (16 four byte words). The NoC link width is assumed to be 128 bits wide, discounting flow-control overheads; therefore cache-line-bearing packets are five flits long, each flit containing up to four words, as shown in Figure 4(a).

## 3 UNUSED WORD DISCOVERY

Our flit-encoding technique manipulates unused words while transmitting cache lines. Thus, we require a method to find those unused words. In this section, we describe how to discover unused words in fills and spills.

### 3.1 Fills

Pujara et al. observed that not all of the words fetched into the cache are used [9]. They define *cache noise* as the words fetched but not used. We also leverage unused words in our scheme. At this early stage in our study, we implement a simple used-word-predictor with some similarities to the *memory context predictor* described by Pujara et al. [9]. In our cache design, we adopt a sectored cache [10] where a cache line is composed of 16 sectors for this prediction mechanism.

Our predictor records used words in a cache block when the cache block is about to be evicted. When re-fetching that particular block, the prediction is made based upon that record. When the predictor does not contain an entry for a given block, all of the words in the block are predicted to be used. If a prediction turns out to be incorrect resulting in false negatives, the system requests a secondary packet containing the rest of the words in the block which have been predicted unused. To lower miss prediction rates, we use two recent history bit-vectors and “OR” them to generate our prediction bit-vector. Those records are associated with a portion of address bits of the block.

Although Pujara et al. found that data accessed from a contiguous group of address will be accessed in a similar

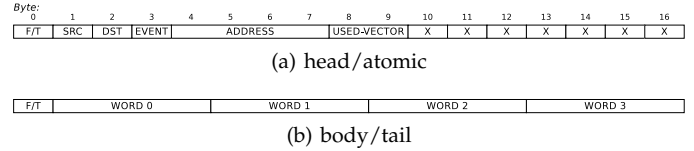


Fig. 2: Flit Format

fashion and indexed their predictor with the upper bits of the address, we found this assumption does not hold in our experimental data. We believe this to be in part due to the randomizing effects of virtual to physical page translation which our simulator emulates. Hence, we use the low order bits of the cache index to index our prediction table; 15 LSBs out of 26 bits of the cache index are used to index the predictor history table. In this version of implementation, we pay little attention to the size of the table, and there could be up to  $2^{15}$  entries per prediction table. The practical size and replacement policy of the table will be examined with prudence in our future design.

It should be noted that while the work presented here exploits used word predictability to save energy on interconnection wires and is therefore orthogonal to Pujara et al.’s Cache-Noise design, the two ideas are complementary. We expect that further saving would be achieved by combining both of those techniques.

## 3.2 Spills

Finding unused words is more straightforward in spills than in fills. For spills, all untouched (i.e clean) words may be considered “unused” for the purposes of our flit-encoding schemes. We propose to replace the single dirty bit in the tag with a dirty bit-vector, where each bit represents the dirty/clean status of one word in a given cache line. We assume write-back as our write policy. On the eviction of a dirty cache block, the cache system looks up the dirty bit-vector and asks the NIC to generate a packet only with those dirty words using one of our flit-encoding techniques.

Translating inconsequential words as proposed by Isen and John [5] into unused words may lead us to additional energy savings but this is left for future work and only clean words are regarded as unused words in spills in this paper.

## 4 FLIT-ENCODING DESCRIPTION

### 4.1 Flit Format

Figure 2 depicts the format of 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 line) or it consists of one atomic flit (when the packet does not). The head/atomic flit contains a bit-vector, labeled “used-vector”, which identifies the words predicted used in this cache line. A used-vector of 0xFFFF represents a prediction that all words will be used while a used-vector of 0xFF00 signifies that only the first eight words will be used. The head flit also contains source and destination node identifiers, and the location of the cache block in memory space. The remaining bytes in the head/atomic flit, labeled with “X” are unused. We assume a flow-control overhead of one byte (labeled “F/T” for flit type on Figure 2). As each of body/tail flit contains data of four words (16 bytes), a flit is 17 bytes wide including flow control overheads.

Packets are composed by the NIC on a request from cache system. A read request, an invalidation or a writeback response is packed into an atomic flit. A packet which contains a cache block, such as a read response or a write request, may have various numbers of flits depending on flit-encoding scheme as described below.

ATOM	1	8	REQ	0x1234	0xFC0A	X	X	X	X	X	X	X	X
0xFC0A = 1111 1100 0000 1010													

Fig. 3: Request Packet

HEAD	8	1	RESP	0x1234	0xFFFF	X	X	X	X	X	X	X	X
BODY0	WORD 0	WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12
BODY1	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15	
BODY2	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15					
TAIL	WORD 12	WORD 13	WORD 14	WORD 15									
0xFFFF = 1111 1111 1111 1111													

(a) Baseline

HEAD	8	1	RESP	0x1234	0xFF0F	X	X	X	X	X	X	X	X
BODY0	WORD 0	WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12
BODY1	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15	
BODY2	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15					
TAIL	WORD 12	WORD 13	WORD 14	WORD 15									
0xFF0F = 1111 1111 0000 1111													

(b) Flit-Drop

HEAD	8	1	RESP	0x1234	0xFC0A	X	X	X	X	X	X	X	X
BODY0	WORD 0	WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12
BODY1	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15	
BODY2	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15					
TAIL	WORD 12	WORD 13	WORD 14	WORD 15									
0xFC0A = 1111 1100 0000 1010													

(c) Word-Repeat

Fig. 4: Response Packet

Figure 3 depicts an example of read request (L1 fill). In this example, tile #1 requests a memory block at address 0x1234 in the 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 packet should 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  $word_{15}$  inside as shown in figure 4(a). In the following sections, we discuss how *flit-drop* and *word-repeat* techniques compose a response packet for this particular request.

## 4.2 Flit-Drop

In the *flit-drop* scheme, 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. Reduction in the number of flits of a packet, in turn reduces the number of bit transitions over interconnect wires and therefore the power consumed. Latency due to packet serialization and NoC bandwidth are reduced as well. Although a read request packet may have an arbitrary used-vector, the response packet must contain all the words for any flit predicted to have even just one used word.

Figure 4(b) depicts the response packet corresponding to the request shown in Figure 3 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 by the processor. These extra words are overheads of the *flit-drop* scheme in the sense that they are not predicted used but must be sent. Although these words waste dynamic power when the prediction is correct, they reduce miss-prediction penalty.

## 4.3 Word-Repeat

The *word-repeat* scheme reduces the activity factor of flits containing unused words to reduce power. The activity factor of a flit containing unused words is reduced by repeating the contents in previous flit in the place of unused words. Flits with less 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. The

bit-lines over which unused words are about to be transmitted have transferred words with the exactly same values. 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. Extra MUX and logic gates are also necessary in the NIC to encode repeated words.

Figure 4(c) depicts the response packet for the request in Figure 3 using the *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.

While *flit-drop* excludes flits with four unused words, *word-repeat* includes all the flits of the packet regardless of usefulness of words inside the flits; therefore, these techniques are complementary and their combination is discussed in section 5.

## 5 EVALUATION

### 5.1 Methodology

We used the M5 full system simulator to generate traces of cache block utilization during program execution [2]. Details of the system configuration are presented in section 2.2. We used the PARSEC suite of shared-memory multi-processor benchmarks as a workload [1, 3]. With these traces, we count the number of words which cause bit transitions as well as packet overhead in order to calculate the effective dynamic power activity factor for each flit-encoding scheme, under perfect and realistic prediction.

### 5.2 Dynamic Power Reduction

Figure 5 shows the breakdown of dynamic network power consumption. For each benchmark, we conducted power simulations for seven configurations, each represented by one stacked bar for that benchmark: 1) *base* - baseline, 2) *fd* - flit-drop with perfect prediction, 3) *wr* - word-repeat with perfect prediction, 4) *fd+wr* - *fd* and *wr* combined, 5) *fd w/p* - flit-drop with realistic prediction, 6) *wr w/p* - flit-drop with realistic prediction, and 7) *fd+wr w/p* - *fd w/p* and *wr w/p* combined. The bars are normalized against the power consumed by the baseline approach. Each bar is subdivided into up to five components. The first bar shows the “used” power, power consumed by the words which will be referenced by the program, hence all the “used” bars are equal. The second bar, “overhead”, shows the power consumed by NoC packet overheads, including header flits and flow control bits. The third bar, “unused”, shows the power consumed to bring in words which will not be referenced. Note that for the *fd* and *fd w/p* approaches, “unused” words are correctly predicted to be unused but packed into flits with predicted used words. The fourth bar, “false+”, shows the power consumed by words which are the result from *false-positive* predictions, or an incorrect prediction of word use. All the “false+” bars are the same for techniques using the realistic predictor such as *fd w/p*, *wr w/p* and *fd+wr w/p*. The fifth bar, “miss penalty”, shows the power consumed by secondary cache line fills to correct “false-negative” miss-predictions.

Our goal is to remove as much as possible of the dynamic datapath power consumed by unused words denoted by *unused*. Unused words consume an average of 35.30% of total dynamic datapath power, and up to 55.54% of total dynamic datapath power in case of *fluidanimate*. With perfect prediction, on average, *flit-drop* reduces total dynamic datapath power by 28.99% and *word-repeat* by 40.05%. Here, *word-repeat* outperforms *flit-drop* due to the unused words of *flit-dropping*. Adopting *flit-drop* on top of *word-repeat* leads us to additional

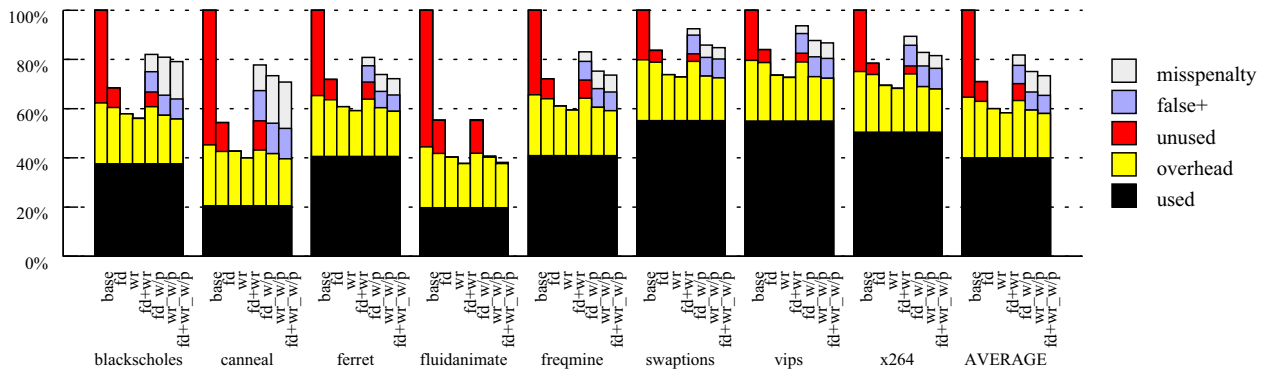


Fig. 5: Datapath Dynamic Power Breakdown

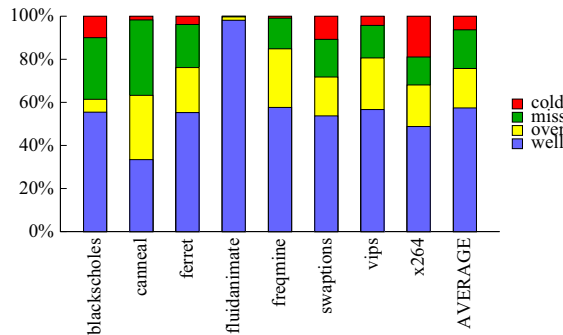


Fig. 6: Used Word Prediction

1.70% saving as flit-drop even removes *overhead* such as flit header (i.e. F/T field).

With the simple predictor, *flit-drop*, *word-repeat* and their combination reduce the total dynamic datapath power consumption respectively by 18.16%, 24.93% and 26.62%, on average. Generally, although 36.93% of the power could be reduced with perfect prediction, around half of that is lost due to false positives and false negatives. However, in case of *fluidanimate*, where the prediction accuracy is 99.62%, *fd+wr w/p* approach removes almost all of the avoidable portion and introduces only 0.5% additional power consumption than *fd+wr*.

### 5.3 Predictor Performance

In this section, we characterize the performance of our simple used word predictor described in section 3.1. As shown in figure 6, we categorized the prediction results into four groups. The bars marked as *well* show the portion of prediction result which exactly matches the real usage. The bars marked as *over* show the portion of the prediction which includes all of the used words plus false positives. These two kinds of prediction results do not cause any miss penalty but we may lose some opportunity of power reduction in case of *over*. The *miss* includes all of the prediction with false negative that result in additional memory system packets. The *cold miss* shows the fraction of predictions to newly referenced cache lines.

At this early stage, the predictor has the average accuracy of 75.69% including *well* and *over*. Our next goal is to reduce the number of false positive in correct prediction and reduce the rate of miss predictions which cause penalties in terms of IPC, network latency and dynamic power. Although, at this stage the power reduction is only ~60% of that possible with a perfect predictor, it should be noted that these reductions should come with little or no impact on performance. In fact, network occupancy and contention should show some reduction which in turn should provide a performance benefit, an effect we plan to characterize in future work.

## 6 CONCLUSION AND FUTURE WORK

In this study, we have proposed flit-drop, word-repeat and a their combination as new flit-encoding methods leveraging unused words in cache lines. We have evaluated these techniques with a simple used-word-predictor which has the average accuracy of 75.69%. Our results show that our best technique achieves an average of 26.62% reduction in total dynamic link power. However, as the combination of flit-drop and word-repeat could have achieved 41.75% reduction in total dynamic link power consumption, further work is required to improve our predictor's performance. We also plan to work on minimizing the extra hardware cost for predicting mechanism and tracking valid words in packets. Finally, we intend to examine how flit interleaving, which occurs in typical virtual channel flow-control utilizing NoC, effects the dynamic power consumption, as well as how interleaving can be leveraged to further improve power savings.

## REFERENCES

- [1] 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*, October 2008.
- [2] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The M5 Simulator: Modeling Networked Systems," *IEEE Micro*, vol. 26, pp. 52–60, 2006.
- [3] M. Gebhart, J. Hestness, E. Fatehi, P. Gratz, and S. W. Keckler, "Running parsec 2.1 on m5," The University of Texas at Austin, Department of Computer Science, Tech. Rep., October 2009.
- [4] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-ghz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, pp. 51–61, 2007.
- [5] C. Isen and L. John, "Eskimo: Energy savings using semantic knowledge of inconsequential memory occupancy for dram subsystem," in *Micro-42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2009, pp. 337–346.
- [6] Y. Jin, K. H. Yum, and E. J. Kim, "Adaptive data compression for high-performance low-power on-chip networks," in *MICRO 41: Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2008, pp. 354–363.
- [7] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in *ACM SIGPLAN NOTICES*. ACM, 2002, pp. 211–222.
- [8] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, "Interconnect-power dissipation in a microprocessor," in *SLIP '04: Proceedings of the 2004 international workshop on System level interconnect prediction*. ACM, 2004, pp. 7–13.
- [9] P. Pujara and A. Aggarwal, "Cache noise prediction," *IEEE Transactions on Computers*, vol. 57, pp. 1372–1386, 2008.
- [10] J. B. Rothman and A. J. Smith, "Sector cache design and performance," in *MASCOTS '00: Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE Computer Society, 2000, p. 124.
- [11] M. Taylor, M. B. Taylor, W. Lee, S. Amarasinghe, and A. Agarwal, "Scalar operand networks: On-chip interconnect for ilp in partitioned architectures," in *International Symposium on High Performance Computer Architecture*, 2002, pp. 341–353.