COOLPRESSION — A HYBRID SIGNIFICANCE COMPRESSION TECHNIQUE FOR REDUCING ENERGY IN CACHES

Mrinmoy Ghosh *Weidong Shi* *Hsien-Hsin S. Lee*
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332
{mrinmoy, shiw, leehs}@ece.gatech.edu

ABSTRACT

This paper describes *CoolPression*, a hybrid hardware-based significance compression technique for reducing energy in caches. The scheme exploits data compression opportunities at bit granularity by employing two compression schemes: a novel significance compression scheme that counts leading ones and zeros, and a dynamic zero compression technique. Based on actual data, the more energy-efficient scheme is selected dynamically to minimize bitline drives needed for each cache access. Using SPECint2000 benchmark, our experiments show that the CoolPression improves dynamic energy consumption by more than 35% against a baseline cache, while having a saving ranged from 5 to 15% compared to each scheme applied alone.

1. INTRODUCTION

Continuous shrinking of transistor feature size and demands of the working set size from increasingly complex applications has led to ever-larger on-chip cache design with a slew of read/write ports making it a major consumer of on-chip power. A significant part of the cache energy is drawn by the bitline driver circuitry because the bitlines are densely loaded with a large number of storage cells thus increasing its effective switching capacitance. To address this issue, many low-power cache techniques were proposed including sub-banking, segmented bit-lines [4, 7], and pulsed word-line drivers [1] , etc.

Another interesting approach is to perform data compression which allows gating off unused bit-lines while reading from and writing data to the cache. Kim et al. [5] describes a sign compression technique, where the most significant half word is compressed to a sign bit to reduce energy. Canal et al. in [2] also applies significance compression to reduce power consumption in all stages of the pipeline. Villa et al. [8] describe a Dynamic Zero Compression (DZC) scheme that compresses if a given entire byte is zero.

In this paper we propose *CoolPression*, a hybrid significance compression technique by using Villa's DZC as a basis along with a novel technique called *CoolCount*, and then dynamically determine the more energy-efficient way to minimize the number of instances of driving data bitlines. The CoolPression circuitry monitors all accesses to the cache and compresses/encodes any data written to the cache, using either of the two compression schemes according to the compressibility of the given data. An extra bit is used to indicate which compression scheme was used. For every read it decodes the data before sending it. The CoolCount technique uses a novel priority encoder and XOR gates, and can count both leading ones and leading zeroes. The novelty of the counting technique lies in the fact that this scheme compresses information[1] in the granularity of bits, while all prior schemes applied compression in the granularity of bytes at best, losing the saving opportunities across byte boundaries. In addition, CoolCount is able to exploit data with leading ones, primarily the small negative integer numbers. Reusing the most significant byte for book-keeping purposes also avoids the area overheads, thus no extra dynamic and leakage energy is consumed. As shown in our experiments, our compression scheme can save an averaged 35% of the cache energy against a baseline cache. As the cache size keeps increasing, the CoolPression will demonstrate even more benefits.

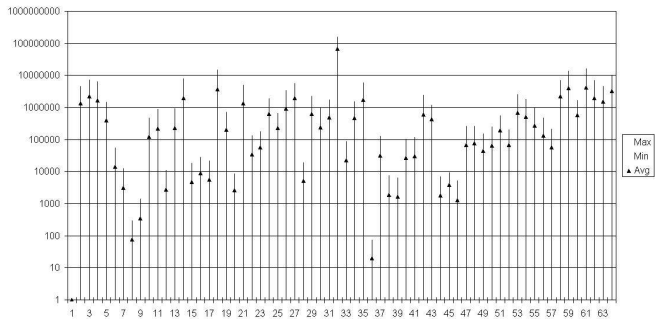
The rest of the paper is organized as follows. Section 2 provides the motivation of adopting a hybrid scheme. Section 3 explains the enabling technique for CoolPression. Section 4 discusses the power and delay of the CoolCount circuit from the CoolPression Cache. Section 5 analyzes our simulated results. Section 6 concludes the paper.

[1]Information hereafter represents both instructions and data in general.

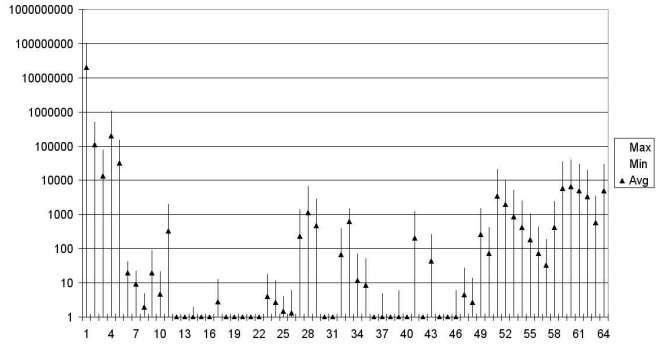
2. MOTIVATION

It has been shown in [3] that more than 70% of bits read from or written to the cache are all zeroes. Also more than 75% of the values used are rather small, having a large number of leading ones or zeroes.

As shown in Figure 1, we conducted a study on the data accesses for SPECint2000 benchmark to profile how many data accesses to the Dcache has "x" ($1 <$ "x" < 64) number of leading zeroes and ones. In the two figures, Y-axis plots the number of leading zeroes (or ones) from 1 to 64 while the X-axis shows the number of instances. The triangle in the plots represents the average number of instances. Each vertical bar shows the range of the number of instances from the 8 SPEC2000 integer benchmark programs. As shown in Figure 1(a), the average number of times we access a piece of data which has "x" number of leading zeroes is quite uniform across the board, for $1<x<64$. A similar trend is also observed for the number of leading ones as shown in Figure 1(b). The encoding techniques proposed in [5, 8] are unable to adequately capture instances in which leading zeroes are not in multiples of 8. Therefore, we will be losing a lot of energy saving opportunities if we only consider compressing data in the granularity of bytes rather than in bits. Based on this analysis, we introduce a compression scheme where we count the leading bits, and keep the count instead of all the bits. We would need 6 counting bits for counting 64 bits of data. We also need another bit indicating whether we counted leading ones or zeroes. If we add 7 bits for every 64 bits of data in the cache, our area overhead would become prohibitively large. We thus propose to reuse the most significant byte of the data to keep the count. Using this method we would need to have only one extra bit for every 64 bits to indicate whether we have employed the counting scheme. If the data being accessed has "count" number of leading zeroes or ones, we need to enable only 64 minus count bitlines to read or write the actual data and append them with lead-



(a) Number of Leading Zeroes



(b) Number of Leading Ones

Figure 1: Leading 0's and 1's for SPECint2000

ing zeroes or ones. We detail our approach in the following section.

3. COOLPRESSION CACHE

The CoolPression cache is illustrated in Figure 2. It employs two compression schemes — Dynamic zero compression to capture the zero bytes, and a new CoolCount technique to exploit compression opportunities in bit-level granularity. To explain our approach we would first explain each one individually before we demonstrate the hybrid approach.

3.1 Dynamic Zero Compression (DZC)

The DZC in [8] uses an extra bit, known as the Zero Indicator Bit (ZIB), for each data byte in the cache. On every data write, it is checked whether any of the eight data bits being written are all 0's, if so, the ZIB is enabled and the write for the eight bits is disabled. If the data bits are not all zeroes then the ZIB is cleared and the data is written to the cache as normal. On a cache read, if the ZIB for a byte is enabled, the corresponding bit-lines are gated off and a zero byte is emitted, through a bank of NOR gates. If the ZIB is zero for a byte then a normal cache read operation occurs.

3.2 CoolCount

Our proposed technique, *CoolCount*, counts the number of leading 0's or 1's and reuses the most significant byte to record the count. On a cache write the CoolCount circuit counts the number of leading 0's or 1's. If the count

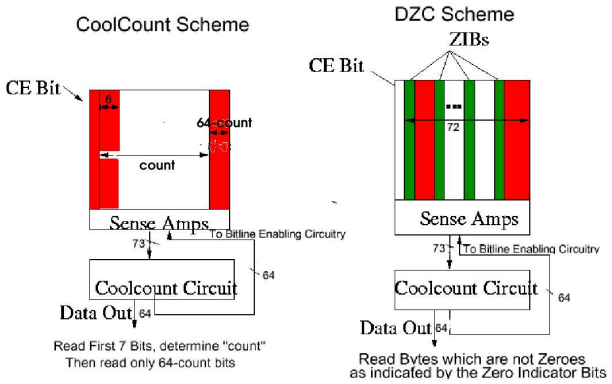


Figure 2: CoolPression Cache

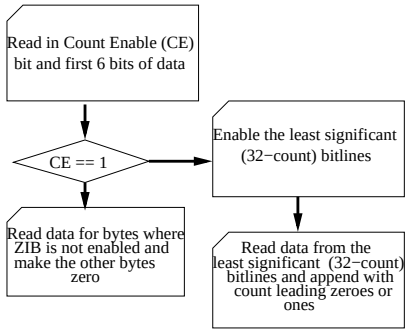


Figure 3: Read Data from Caches

is more than eight, it asserts the Count Enable (CE) bit high. The most significant bit is used to store whether we counted leading 0's or 1's. The next six bits from the most significant byte are used to keep the count. Along with this, (64 - count) least significant bitlines are enabled to store the actual data. If the number of leading 0's or 1's is lower than eight, the cache performs a normal write. The cache read is illustrated in Figure 3. Reading is a two step but pipelined process. On a cache read if the CE bit is enabled, the most significant 6 bits are read to get the number of leading 0's or 1's. Next the least significant bit-lines are enabled to get the actual data appended with the leading bits to obtain the final data. However if the CE bit is zero, the CoolCount cache behaves exactly like a normal cache for a read.

3.3 Hybrid Compression Scheme

Our evaluation indicated that both techniques can lead a significant energy saving when operated independently. However, there are certain cases where the CoolCount scheme outperforms the DZC and in some cases the DZC does better. For instance, CoolCount can capture energy saving opportunities at finer granularity while DZC can exploit the hidden opportunities where zero bytes are embedded in the middle of a data word. Based on this observation, we propose a hybrid compression scheme which employs both DZC and CoolCount and exploits all possible opportunities in a dynamic manner.

The operations occurring in the CoolPression cache for reads and writes are illustrated by the flow-charts in Figure 3 and Figure 4. On each cache write, one circuit will count the leading 0's or 1's, while another circuit counts how many bytes of the data are zero. These circuits and their energy impact are detailed in Section 4. The results are compared to determine which scheme is better. If the CoolCount is better, the CE bit is turned on, all the ZIB's

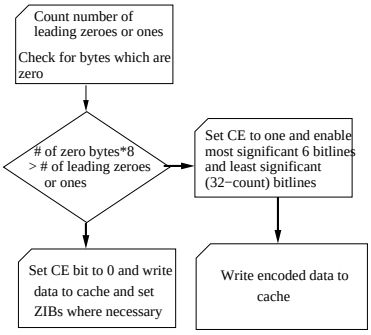


Figure 4: Write Data to Cache

are cleared and the CoolCount scheme is followed. If DZC is found better, the CE bit is cleared, the corresponding ZIB's are enabled and the DZC is used. For reading data from the cache, after address decoding is completed, the CE bit, the most significant 6 bits and the ZIBs are read in the first cycle of the cache read. The CE bit enables the Coolcount decoder shown in Figure 5 which uses the *count* bits to precharge the least significant *64 - count* bits before the start of the second cycle. If the CE bit is disabled, the ZIBs are used to precharge only the relevant *"byte"* lines. The precharged bit lines are read in after the end of the second cycle and properly appended with zero or one bits to form the final data.

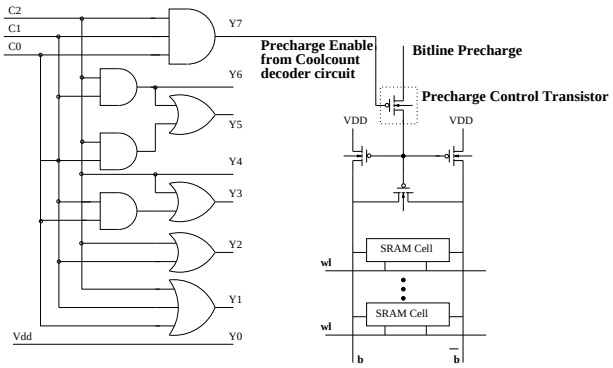
4. COOLPRESSION IMPLEMENTATION

Figure 5 illustrates the schematic of our logic circuits for an efficient implementation for counting the leading 0's or 1's. For illustration purposes, the circuit shown in Figure 5(b) is a stripped-down version using a 8-to-3-line priority encoder for an 8-bit data set. Depending on the data size supported by the targeted ISA, for example, the CoolCount circuit will have a 64-to-6-line priority encoder for a 64-bit data set. Note that as shown in the schematic, the adjacent bits of the data are XORed together and fed into the priority encoder. This design is to determine when the data first changes from 0 to 1 or from 1 to 0, as we scan the bits from the most significant bit. The position of this bit is reported by the priority encoder. The number of leading zeroes or ones is obtained by negating the above result.

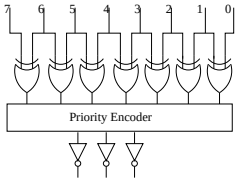
Figure 5(a) illustrates the decoding circuits to enable only the required bitlines depending on the number of leading 0's or 1's. The circuit illustrated is again for 8-bit data only. The CoolCount circuits will be scaled for a 64 bit data set. The output of the decoding circuit are used to disable the Bitline Precharge signal using the Precharge Control Transistor as shown in the Figure 5(a). This mechanism allows only the required bitlines from precharging when reading or writing compressed lines to the cache. We explain the overheads in terms of delay and extra power consumed for the CoolPression circuit in the following subsection.

4.1 Overheads

Delay Overheads: To consider the effect of delay overheads we note that the decoding circuit Figure 5(a) is extremely simple, a tiny overhead compared to the cache array, and the delay associated with it is minimal. Since reading is a two stage process as shown in Figure 3, we assume that a L1 cache read would take 2 cycles instead of one, even though we could have headroom to customize the entire read into one cycle for a lower frequency processor. The 2 cycle latency can be easily pipelined by dividing the cache access into address decoding and data transfer stage.



(a) Decoder and Precharge Control Circuit



(b) Encoder

Figure 5: Decoding/Encoding Logic Circuits

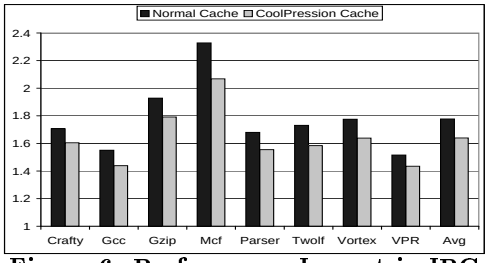


Figure 6: Performance Impact in IPC

This disadvantage is also omnipresent in other cache compression schemes [5, 8]. This is an indispensable overhead in any cache compression since extra time is always needed for ascertaining whether the data were compressed prior to being stored. Figure 6 studies the effect if a 2-cycle pipelined Instruction and Data Cache is to be designed against a single cycle uncompressed cache. It is observed that the average IPC can be degraded by 7.8% for a 35% savings in energy consumption of the cache. The processor architects have to weigh the trade-off when making such a design decision. The delay associated with the Priority Encoder, is in the order of 6 gate delays. Nevertheless, since the delay is only associated with a write and not a read, it will be hidden in practice in the interval between a write and the next read of the same cache line.

Power Overheads: The majority of the delay and energy consumption associated with the CoolCount circuit is caused by the Priority Encoder (PE). The PE design used for this study is an energy efficient high speed PE design taken from [9]. The reported power numbers have been scaled down to the current process technology parameters using standard technology scaling rules [6]. The cache energy consumption numbers were reported using Wattch. All numbers cited are for 0.1μm technology. The Cool-

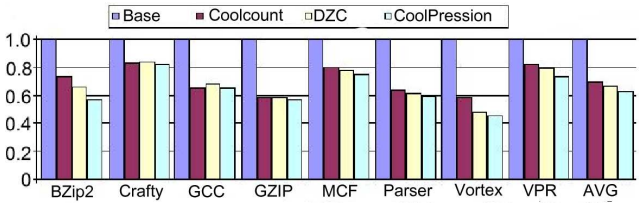


Figure 7: Norm. Energy in a 16KB L1 D-Cache

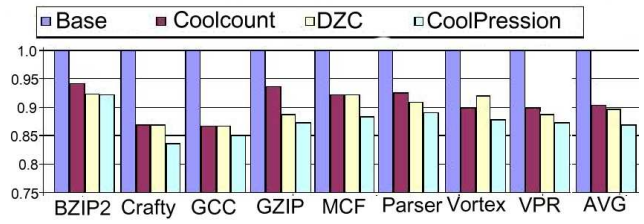


Figure 8: Norm. Energy in a 16KB L1 I-Cache

Pression circuit only consumes 0.1% of the 64KB cache power as reported by Wattch.

5. EXPERIMENTAL RESULTS

We integrated the CoolPression technique into SimpleScalar to quantify the energy savings. The processor model is similar to an Alpha 21264. Wattch and Cacti were used to model total cache energy dissipation in each application and the energy consumed in each cache array. A two level cache was used and our technique was applied to both levels. We simulated SPECint2000, each for 1 billion instructions.

Figure 7 shows the energy consumption in a 16K direct mapped L1 data cache using the CoolCount, DZC and hybrid CoolPression, taking a word size of 64 bits. All the results are normalized to the baseline cache with no compression. We observe that the DZC and CoolCount are on par within 3% savings, and there is no obvious trend as to which one is better. Since the CoolPression can dynamically choose the better scheme, therefore, as expected, a lower energy is dissipated. The CoolPression beats the DZC from 3 to 15%. In overall, the CoolPression provides an energy savings of 36% to the baseline cache case. Figure 8 illustrates a similar analysis on a 16K L1 Instruction cache which shows that the CoolPression generates considerable savings even though it is not as significant as the data cache. One reason for lower energy savings in the Icache is that the data in the Icache consists of instructions whose encodings would have a higher entropy in terms of the distribution of 0's and 1's, while data, mostly, show a large number of small positive and negative numbers, i.e. many consecutive 0's and 1's, for integer programs. Our technique did not show any substantial improvement for the unified L2 cache since there are very few accesses to the L2 cache in the benchmark we studied, and the L2 energy consumption is largely dominated by the leakage energy due to its size.

In Figure 9, we plot the energy consumption for 32 KB and 64KB L1 Dcache and Icache, showing CoolPression reduces cache energy by as much as 50%. It is observed that the CoolPression cache gives substantial energy savings for the larger caches. In addition, what we do not show due to space contraint is that the CoolPression also consistently provides more saving than the DZC and the CoolCount when applied alone. The trend is similar to what was shown in Figure 7 and Figure 8.

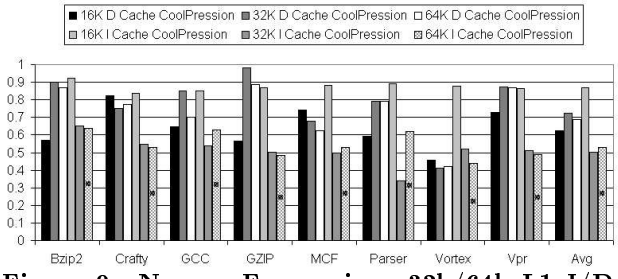


Figure 9: Norm. Energy in a 32k/64k L1 I/D-Cache

6. CONCLUSION

In this paper we present *CoolPression*, a hybrid hardware-based data compression mechanism to reduce energy consumption in caches by eliminating unnecessary bitline switchings. CoolPression consists of a dynamic zero compression technique with a novel CoolCount scheme to exploit both byte and bit-level compressibility. The CoolPression is system transparent in the sense that the rest of the system does not change their interface with the CoolPression cache. The CoolCount circuitry used a small amount of hardware to encode data stored in the cache,and succeeded in reducing the energy consumption by a significant amount. Based on our simulation results using SPECint2000, the CoolPression improves dynamic energy consumption by more than 35% against a baseline cache, while having a saving ranged from 5 to 15% compared to the dynamic zero compression or CoolCount applied individually.

The novel encoding scheme discussed in the paper may be extended to save energy at all places wherever data is being transferred. Some important locations that may be considered are the pipeline latches. We may consider using this technique for data transfer from L2 cache to memory and memory to the disk. Effects of reducing the bus transfer power at certain places where we would encode the data using this scheme and transfer only the required data gating off the other bits may also be considered.

7. ACKNOWLEDGEMENTS

This research was supported by NSF Grants CCF-0326396 and CNS-0325536. We would also like to thank Josh Fryman for his valuable suggestions.

8. REFERENCES

[1] B. Amrutur and M. Horowitz. Techniques to Reduce Power in Fast Wide Memories. In *Proc. of the Int'l Symp. on Low Power Electronics*, 1994.

[2] R. Canal, A. Gonzalez, and J. E. Smith. Very low power pipelines using significance compression. In *MICRO-33*, 2000.

[3] Y.-J. Chang, C.-L. Yang, and F. Lai. A Power-Aware SWDR Cell for Reducing Cache Write Power. In *ISLPED-03*, 2003.

[4] K. Ghose and M. B. Kamble. Reducing power in superscalar processor caches using subbanking, multiple line buffers and bit-line segmentation. In *ISLPED-99*, 1999.

[5] N. S. Kim, T. M. Austin, and T. N. Mudge. Low-Energy Data Cache Using Sign Compression and Cache Line Bisection. In *2nd Workshop on MPI*, 2002.

[6] A. Matsuzawa. RF-SoC - Expectations and Required Conditions. In *IEEE Trans. on Microwave Theory and Techniques*, 2002.

[7] C.-L. Su and A. M. Despain. Cache design trade-offs for power and performance optimization: a case study. In *Proc. of the Int'l Symp. on Low Power Design*, 1995.

[8] L. Villa, M. Zhang, and K. Asanovic. Dynamic zero compression for cache energy reduction. In *MICRO-33*, 2000.

[9] J.-S. Wang and C.-H. Huang. High-speed and low-power cmos priority encoders. *JSCC*, 35(10), 2000.