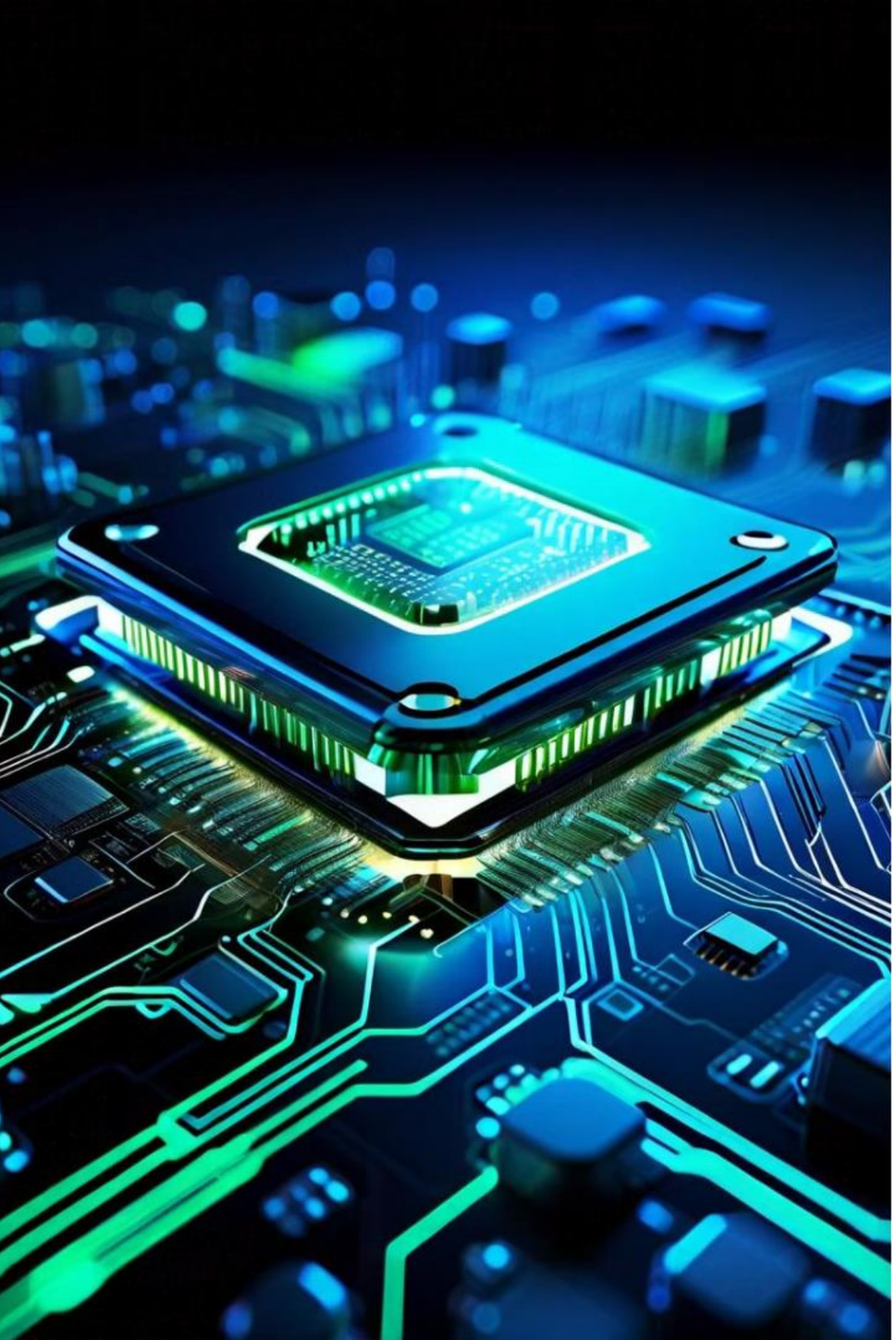


Base-Delta-Immediate Compression: Practical Data Compression for On-Chip Caches

BΔI is a cache compression technique that increases capacity and performance in computer memory systems. It uses simple algorithms and has low hardware overhead. BΔI could improve overall system performance, making it a valuable contribution to computer architecture.

By: Shabbir Aglodiya 21BEC006
Sujal Bhojani 21BEC016

Motivation and Background

1

Redundant Data Patterns

Real-world applications often exhibit common data patterns that can be exploited for exploited for effective compression, such as the prevalence of zero values, repeated repeated values, and narrow value ranges. Leveraging these patterns is crucial for for improving the efficiency of on-chip caches.

2

Limitations of Existing Techniques

While previous cache compression methods have shown promise, they often sacrifice one aspect for another, such as achieving a low compression ratio or incurring high hardware complexity and decompression latency. Addressing these tradeoffs is the key challenge that BΔI aims to overcome.

3

Potential Benefits of Compression

Successful cache compression can provide significant benefits, including increased effective cache capacity, reduced off-chip memory accesses, and improved overall system performance. Developing a practical and efficient compression technique is crucial for realizing these advantages in modern computer systems.

The BΔI Compression Approach

Exploiting Limited Value Ranges

The core idea behind BΔI compression is that many cache lines contain data with a limited dynamic range of values. By representing these cache lines using a common base value and an array of relative differences (deltas), BΔI can achieve significant compression without sacrificing decompression performance.

Dual Base Compression

To further improve compression efficiency, BΔI employs two base values: one arbitrary base and one fixed at zero. This dual-base approach allows BΔI to effectively compress cache lines containing a mix of distinct value ranges, such as pointer values and small integer values.

Low-Complexity Design

The BΔI compression and decompression algorithms rely on simple vector operations, such as addition, subtraction, and comparison. This design ensures low hardware overhead and fast decompression, which are crucial for practical implementation in commercial microprocessors.

BΔI Compression Algorithm

Compression

The BΔI compression algorithm first assesses whether a cache line can be effectively compressed by considering the dynamic range of its values. It then selects the optimal base value and determines the size of the delta values, aiming to minimize the overall compressed size. The compressed cache line consists of the base value and the array of deltas, which can be decompressed efficiently.

Decompression

Decompressing a BΔI-compressed cache line is a straightforward process involving a simple vector addition operation. The decompression logic adds the base value to the corresponding delta values, reconstructing the original cache line data. This low-latency decompression is a key advantage of the BΔI approach over more complex compression techniques.

Hardware Design

The BΔI compression and decompression logic is designed with simplicity and efficiency in mind. The compression units operate in parallel, while the decompression logic utilizes a SIMD-style vector adder to enable fast decompression. The overall hardware overhead is relatively modest, making BΔI a practical solution for on-chip cache compression.

BΔI Cache Organization

1

Compressed Cache Structure

The BΔI cache design introduces modifications to a conventional cache to efficiently leverage the benefits of compression. By doubling the number of tags, the cache can store twice as many compressed cache lines in the same data storage area, effectively increasing the overall cache capacity without significant hardware overhead.

2

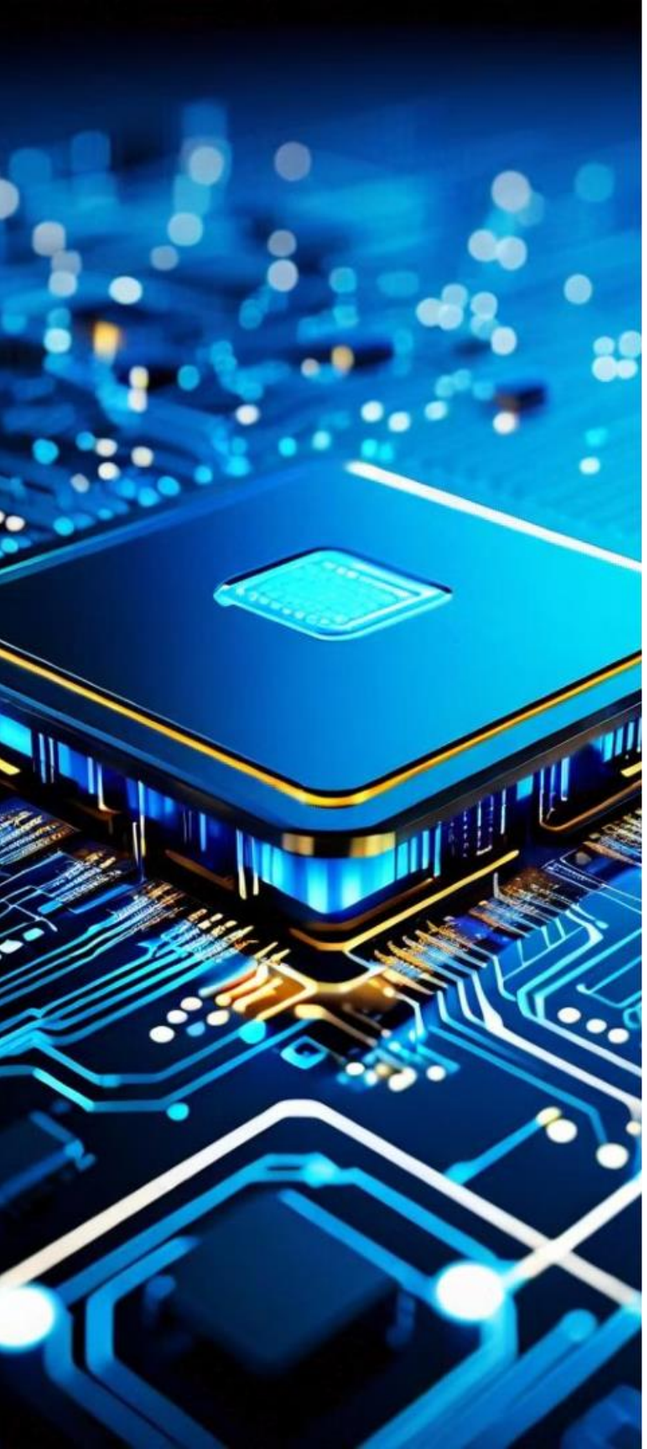
Compression and Decompression

When a cache line is accessed, it is first decompressed using the BΔI decompression logic before being transferred to the L1 cache. Conversely, when a new cache line is inserted or an existing one is modified, it undergoes compression and is stored in the compressed L2 cache.

3

Eviction Policy

To handle the variable size of compressed cache lines, the BΔI cache employs a modified LRU eviction policy that can evict multiple cache lines if necessary to create enough space for the incoming or modified line. This ensures efficient utilization of the compressed cache capacity.



Evaluation and Results

1 Compression Ratio

Extensive evaluations demonstrate that BΔI compression achieves a higher average compression ratio compared to three state-of-the-art cache compression techniques, including Frequent Value Compression (FVC) and Frequent Pattern Compression (FPC).

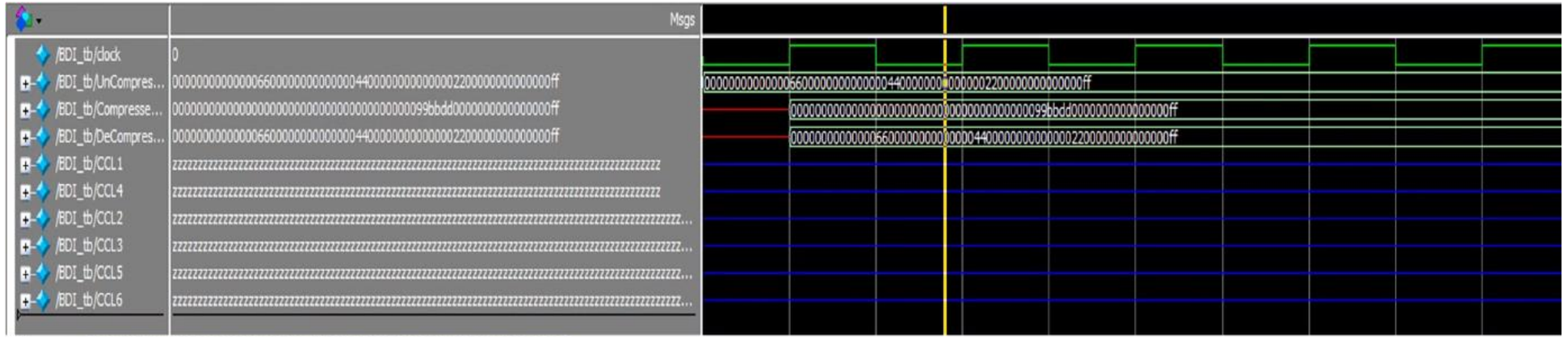
2 Performance Gains

The performance improvements enabled by BΔI compression are significant, with an 8.1% average speedup for single-core workloads and 9.5% and 11.2% average speedups for two-core and four-core multi-core workloads, respectively.

3 Comparison to Larger Caches

In many scenarios, the performance benefits of using BΔI compression closely match those of doubling the uncompressed cache size, highlighting its effectiveness in increasing the effective cache capacity without the drawbacks of a larger physical cache.

Simulation




```
# input = 0000000000000066000000000000004400000000000000220000000000000ff
#
#  8_1: del1= 00000000000000dd, del2 =00000000000000bb, del3 = 0000000000000099
#
# CoN1 = 1, CCL1 = 99bbdd00000000000000ff
# CoN2 = 1, CCL2 = 009900bb00dd00000000000000000ff
# CoN3 = 1, CCL3 = 00000099000000bb000000dd0000000000000000000000ff
#  BASE 4: del1= 000000ff, del2 =000000dd, del3 = 000000ff del4 = 000000bb del5 = 000000ff del6 = 00000099 del7 = 000000ff
#
# CoN4 = 1, CCL4 = ff99ffbbffddff00000000ff
# CoN5 = 1, CCL5 = 00ff009900ff00bb00ff00dd00ff0000000000ff
#  BASE 2: del1= 00ff, del2 =00ff, del3 = 00ff del4 = 00dd del5 = 00ff del6 = 00ff del7 = 00bb  del8= 00ff, del9 =00ff, del10 = 00ff del11 = 00ff del12 = 0099 del13 = 00ff del14 = 00ff del7 = 00ff
#
# CoN6 = 1, CCL6 = ffffff99ffffffffbbffffddffffff0000ff
# input = 0000000000000066000000000000004400000000000000220000000000000ff
#
#  8_1: del1= 00000000000000dd, del2 =00000000000000bb, del3 = 0000000000000099
#
# CoN1 = 1, CCL1 = 99bbdd00000000000000ff
# CoN2 = 1, CCL2 = 009900bb00dd00000000000000000ff
# CoN3 = 1, CCL3 = 00000099000000bb000000dd0000000000000000000000ff
#  BASE 4: del1= 000000ff, del2 =000000dd, del3 = 000000ff del4 = 000000bb del5 = 000000ff del6 = 00000099 del7 = 000000ff
#
# CoN4 = 1, CCL4 = ff99ffbbffddff00000000ff
# CoN5 = 1, CCL5 = 00ff009900ff00bb00ff00dd00ff0000000000ff
#  BASE 2: del1= 00ff, del2 =00ff, del3 = 00ff del4 = 00dd del5 = 00ff del6 = 00ff del7 = 00bb  del8= 00ff, del9 =00ff, del10 = 00ff del11 = 00ff del12 = 0099 del13 = 00ff del14 = 00ff del7 = 00ff
#
# CoN6 = 1, CCL6 = ffffff99ffffffffbbffffddffffff0000ff
```

Key Innovations of BΔI

High Compression Ratio

BΔI effectively leverages common data patterns in cache lines to achieve a high compression ratio, outperforming state-of-the-art techniques.

Low Decompression Latency

The simple vector-based decompression algorithm of BΔI ensures low-latency decompression, crucial for maintaining system performance.

Low Hardware Complexity

The BΔI design minimizes hardware overhead, making it a practical solution for implementation in commercial microprocessors.

Significant Performance Gains

BΔI compression can provide performance improvements comparable to doubling the uncompressed cache size, without the associated drawbacks.

Broader Impact and Applications

Memory Hierarchy Optimization

While this research focuses on applying BΔI compression to on-chip caches, the technique could potentially be extended to other levels of the memory hierarchy, such as main memory and storage systems, to further improve overall system efficiency and performance.

Embedded and Mobile Devices

The low-complexity and high-efficiency characteristics of BΔI make it an attractive option for deployment in resource-constrained embedded and mobile devices, where maximizing the effective memory capacity is crucial for performance and power efficiency.

Specialized Hardware Accelerators

The BΔI compression and decompression algorithms, with their reliance on simple vector operations, could be easily integrated into specialized hardware accelerators, further enhancing the performance and energy efficiency of various computing systems.

Future Research Directions

Adaptive Compression Policies

Exploring more sophisticated cache eviction policies that consider the compressed size of cache lines could further optimize the utilization of the compressed cache capacity and improve overall system performance.

Integration with Other Techniques

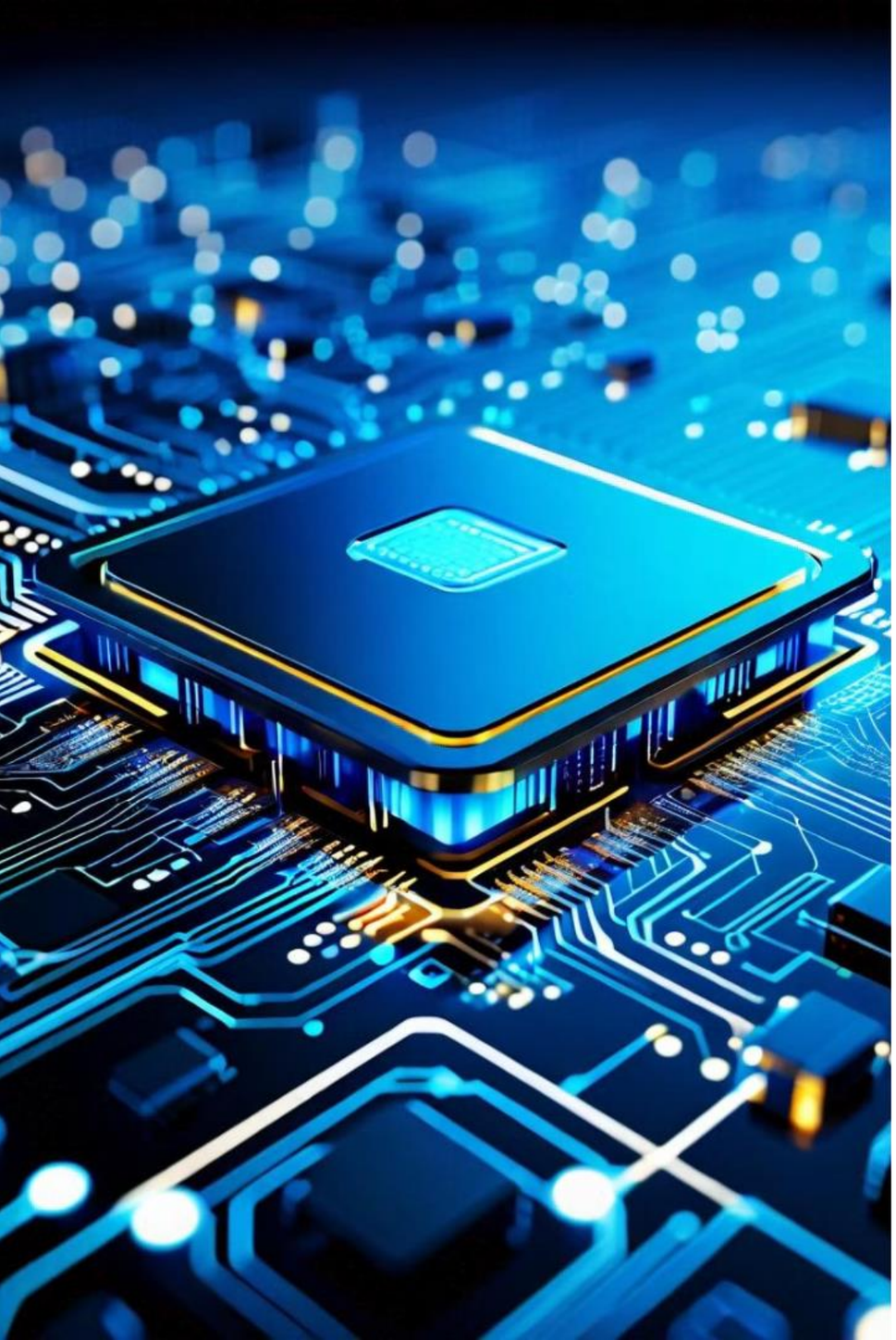
Investigating the potential synergies between BΔI compression and other cache optimization techniques, such as cache bypassing and prefetching, could lead to even greater performance improvements.

Extending to Other Memory Levels

Expanding the application of BΔI compression beyond on-chip caches, to main memory and storage systems, could yield significant benefits in terms of capacity, bandwidth, and energy efficiency for a wide range of computing systems.

Hardware-Software Co-Design

Exploring opportunities for hardware-software co-design, where the compression algorithm and cache organization are tailored to specific application domains, could further enhance the effectiveness of BΔI compression.



Conclusion

BΔI is a cache compression technique that increases capacity and performance in computer memory systems. It uses simple algorithms and has low hardware overhead. BΔI could improve overall system performance, making it a valuable contribution to computer architecture.