Enhancing Practicality of Memory Compression for GPUs with High-Throughput Simplifications

Manuel Renz
Technische Universität Hamburg, Germany
manuel.renz@tuhh.de

Sohan Lal
Technische Universität Hamburg, Germany
sohan.lal@tuhh.de

ABSTRACT

Memory-bound Graphics Processing Unit (GPU) applications are limited by memory bandwidth, as the rapid growth in computational power has outpaced the slower increase in memory bandwidth. Consequently, approaches such as memory compression, are gaining prominence to synthetically enhance memory bandwidth and accelerate bandwidth-limited applications. Traditionally, compression techniques have been tailored towards achieving high compression ratios without considering the high bandwidth of modern GPU memory systems which makes hardware integration costly and impractical. We analyze several state-of-the-art memory compression techniques and finds that the throughput of Bit-Plane Compression (BPC) and Frequent Pattern Compression (FPC) is limited by Zero Run-Length Encoding (ZRLE), which efficiently compresses zero blocks, however, GPUs often do not benefit as even heavily compressed blocks require the transfer of a full Memory-Access Granularity (MAG). We propose simplifying the BPC and FPC techniques by removing ZRLE and introducing a fixed-size tag section. Together with higher word-level parallelism, our simplifications increase compressor throughput by 14.0× and decompressor throughput by 13.5× without any loss in the effective compression ratio. Additionally, the area required for hardware integration is significantly reduced; for instance, the area cost of BPC is decreased by 3.6×, and power consumption by 1.8×, making hardware integration of memory compression more practical and cost-effective.

CCS CONCEPTS

• **Computer systems organization** → **Single instruction, multiple data**; • **Theory of computation** → **Data compression**.

KEYWORDS

Memory compression, GPUs, throughput

1 INTRODUCTION

Memory bandwidth remains a fundamental bottleneck in GPU architectures as computational power continues to grow faster than the off-chip memory bandwidth [8, 10]. As conventional strategies to increase memory bandwidth have limitations due to power consumption and physical constraints [7], alternative approaches such as memory compression have been proposed to mitigate the memory bandwidth bottleneck [4–6, 11]. Integrating a compressor and decompressor into the GPU's Memory Controller (MC) enable data transfers to and from the off-chip memory in a compressed format, reducing the effective amount of data transferred.

However, for a practical integration of memory compression, the throughput of a compression technique has to keep up with the high memory bandwidth of GPUs. Otherwise, the (de)compressors themselves can become a bottleneck. Figure 1 shows the memory bandwidth of three modern GPUs and the compressor throughput of four state-of-the-art memory compression techniques: BPC [4], Cache-Packer (C-PACK) [2], FPC [1], and MAG-Aware Base Delta-Immediate (M-BDI) [6] Compression. Firstly, we observe that the throughput of all techniques is significantly lower than the memory bandwidth of the GPUs. Secondly, we observe that M-BDI achieves a higher throughput (78.01 GB/s) than the other techniques (BPC/FPC: 26.01 GB/s, C-PACK: 9.18 GB/s). This is because the (de)compressor design of M-BDI is comparatively simple.

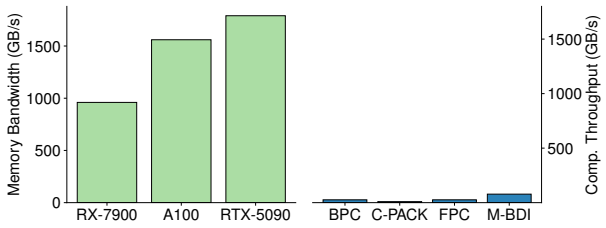


Figure 1: Memory bandwidth of GPUs and compressor throughput of state-of-the-art techniques [9].

While M-BDI provides high throughput, its compression capability is limited to simpler data patterns, as it requires similar values across a cache line. Our results show that the effective Compression Ratio (CR) of M-BDI is 10% lower compared to BPC (see Section 5.3). In addition, it is crucial to select appropriate compression techniques based on workloads and system requirements as hardware-based compression techniques are fixed for the processor's lifetime. Furthermore, offering multiple compression options also enables the development of multi-compression frameworks [3].

This motivated us to analyze memory compression techniques with respect to throughput. We find that both BPC and FPC compression are limited by the special optimization ZRLE which efficiently compresses zero blocks but increases complexity and latency. Therefore, we propose to remove the ZRLE component. Our results show that this simplification 1) maintains or achieves a higher effective CR due to the large MAG, 2) allows for a fixed-size tag section, greatly simplifying the decompressor design, 3) enables higher clock speed for the (de)compressor, and 4) boosts compressor (14.0×) and decompressor (13.5×) throughput by several times. In summary, we make the following main contributions:

- We propose simplifying the BPC and FPC techniques by removing ZRLE and introducing a fixed-size tag section. This modification boosts compressor and decompressor throughput by 14.0× and 13.5× respectively.
- We implement the memory compression techniques in VHDL and synthesize the designs. The optimized BPC/FPC have 3.6×/8.2× lower area cost, while power consumption is decreased by 1.8×/5.0×, respectively.

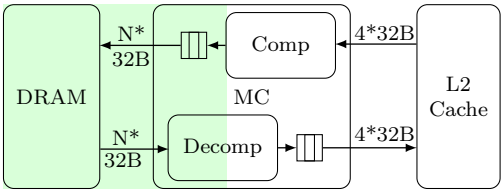


Figure 2: Architectural changes for memory compression.

- The simplification can be realized without any reduction in the effective CR compared to original designs.

This paper is structured as follows: Section 2 reviews related work. Section 3 provides background. In Section 4, we propose throughput optimizations. Section 5 outlines experimental methodology and presents results. Finally, Section 6 offers conclusion.

2 RELATED WORK

Various hardware compression techniques have been proposed for GPUs [1, 4, 6]. Prioritizing high compression ratios, these publications often overlook throughput as performance metric. For instance, FPC [1] uses 8 parallel word (de)compressors, while BPC [4] uses 9, both suggest using more word (de)compressors to increase throughput but lack quantitative evaluation. C-PACK [2] uses 2 parallel word (de)compressors and discusses the possibility to scale the number of parallel word (de)compressors but also notes that this would be extremely costly. Entropy Encoding Based Memory Compression (E^2MC) [5] evaluates (de)compressor latency for processing 1, 2, 4, and 8 words in parallel. While the approach is sound, it lacks a thorough comparison of throughput with other techniques. M-BDI [6] works on the whole cache line in parallel. Finally, Renz et al. [9] analyze the throughput of hardware compression techniques for GPUs. They find that several techniques have insufficient throughput and are impractical for a hardware integration into modern GPUs. The findings are insightful, yet, the work lacks concrete proposals for throughput optimization.

3 BACKGROUND

Memory Compression: Figure 2 shows a GPU architecture with memory compression in which the MC is augmented with a compressor and decompressor. Data is compressed before writing to off-chip memory and decompressed after reading [11]. GPUs access off-chip memory in coarse-granular chunks, so-called MAG, which means that the effective amount of data transferred is higher than the raw compressed block sizes. For instance, assuming 32B MAG size, if a block is compressed to any size smaller than 32B, say 7 bits by ZRLE, it will still result in a 32B data transfer.

FPC and BPC: FPC is a compression technique which checks for each word in a block whether it falls into one of the predefined frequently occurring patterns. It then stores the word in compressed form along with the corresponding tag. BPC is aims to compress homogeneously-typed memory blocks, common in workloads for many-core systems. It applies the so-called Delta-BitPlane-XOR (DBX) transformation which increases the data compressibility.

ZRLE: ZRLE is a special case in compression which encodes consecutive zero words into a few bits. BPC and FPC employ ZRLE to further compress the tags in a compressed block, see Figure 3.

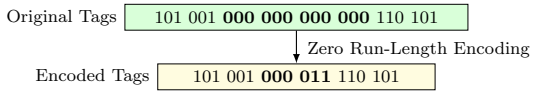


Figure 3: ZRLE applied to 3-bit tags. Consecutive zero tags are replaced by one leading tag and the run-length.

Table 1: Updated BPC tags for DBP/DBX symbol encoding.

DBP/DBX Pattern	Length	Code
0 word	3-bit	3'b000
All 1s	3-bit	3'b001
DBX!=0 & DBP=0	3-bit	3'b010
Single 1	8-bit	3'b011, PosSingleOne[4:0]
Consecutive two 1's	8-bit	3'b100, StartPosTwoOnes[4:0]
Two Single 1's	13-bit	3'b101, StartPosFirstOne[4:0], StartPosSecondOne[4:0]
Single 0	8-bit	3'b110, PosSingleZero[4:0]
Uncompressed	35-bit	3'b111, UncompressedData[30:0]

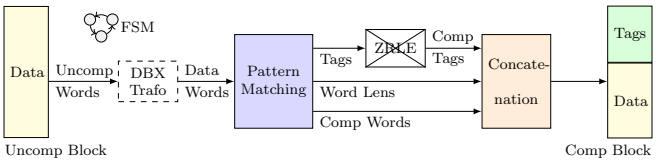
4 SYSTEM DESIGN

In this section, we present the optimized (de)compressors for BPC and FPC. A straightforward attempt to increase throughput would be to simply deploy 16 or 32 parallel word (de)compressors without removing ZRLE. However, this approach is costly and provides little throughput gain as the variable length of the tag section forces the (de)compressor to sequentially process all tags before concatenation / parallel word decompression. Moreover, our synthesis results indicate that a sequential ZRLE implementation limits the maximum clock frequency in a pipelined design. A parallel ZRLE implementation, which exhaustively lists all possible combinations of zero/non-zero tags, is even more expensive according to our results, as the run-length depends on all previous tags.

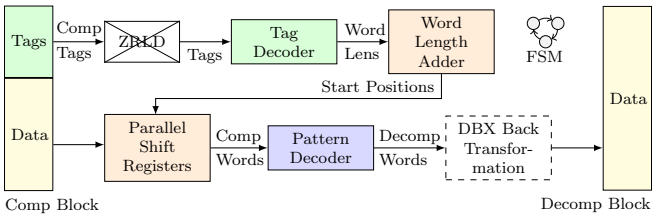
Therefore, to enable high-throughput (de)compression, we propose the following simplifications for FPC: the ZRLE component is removed from the compressor and decompressor, while the number of word (de)compressors per (de)compression unit is increased to 16 or 32. No changes to FPC's tags are required as they are already static. The simplification of BPC involves a few additional steps: first, as for FPC, the ZRLE is removed from the designs. Secondly, the BPC tags are updated to 3 bits length to enable a statically-sized tag section, as depicted in Table 1. The first five patterns and the uncompressed case remain unchanged. Since zero runs are no longer required, this frees two entries in the table which can be assigned to other commonly occurring patterns (bold). Finally, the number of word (de)compressors is increased to 16/32 units.

Figure 4 shows the optimized (de)compressor designs of BPC and FPC, applying the optimizations discussed in the previous paragraph. The designs are pipelined, however, the components need to be reused over multiple cycles. The BPC-OPT-32 and FPC-OPT-32 compressors could be fully pipelined.

Compressor: The optimized FPC compressor consists of only two steps: first, the 16/32 pattern matchers check for frequently occurring patterns in parallel. Secondly, the compressed words are concatenated using the word lengths and data words. BPC additionally transforms the uncompressed words into the bitplane before pattern matching to increase data compressibility. The concatenation stage and shift registers can be simplified as well.



(a) Optimized BPC/FPC compressor design



(b) Optimized BPC/FPC decompressor design

Figure 4: Optimized (de)compressor designs: ZRLE removal allows a fixed-size tag section, enabling parallel decompression. DBX transformations are only applied to BPC.

Decompressor: The decompression with FPC is a four-stage process: 1) 16/32 tags are decoded in parallel to generate the word lengths. 2) the word lengths are added to generate the starting positions of compressed words. 3) the compressed words are generated with the parallel shift registers. 4) using the data tags, the pattern decoder outputs the uncompressed data words. For BPC, an additional fifth step is the back-transformation from the DBX plane into the data plane. The optimization of concatenation logic and shift registers is only partly applicable, as compressed word lengths are not multiple of 4s, but rather 5/10/31 bits.

Table 2: Simulator configuration (based on RTX 3070).

Parameter	Value	Parameter	Value
#SMs	46	L1 $ size/SM	128 KB
SM freq	1500 MHz	L2 $ size	4 MB
Max #Threads per SM	1536	# Memory controllers	16
Max #CTA per SM	32	Memory type	GDDR6
Max CTA size	512	Memory clock	1750 MHz
#FUs per SM	64	Memory bandwidth	448.0 GB/s
#Registers/SM	65 K	Burst length	16
Shared memory/SM	102 KB	Bus width	256 bits

5 EXPERIMENTAL RESULTS

To estimate the delay, throughput, area, and power consumption, we implemented VHDL models of the various techniques. We then synthesized the designs using Synopsys Design Compiler V-2023.12-SP2 with 14nm EDK libraries and an operating voltage of 0.72V. For simulation, we modify gpgpu-sim 4.0 to include the (de)compressors. Table 2 shows the main configuration parameters. We use a block size of 128B and a MAG size of 32B. In results, the suffixes indicate the number of parallel word (de)compressors used, e.g. BPC-OPT-32 denotes an optimized BPC design with 32 parallel word (de)compressors. The SSSP benchmark was simulated for 10 billion instructions due to long runtime.

Table 3: Results for single compressor/decompressor units.

	Technique	Delay (cycles)	Max Clk (GHz)	Max TP (GB/s)	Area (μm^2)	Power (mW)
Comp	BPC-8	7	2.5	45.7	25050	2.96
	BPC-OPT-16	3	10.0	426.7	98585	15.17
	BPC-OPT-32	2	10.0	640.0	156738	12.26
	FPC-8	7	2.5	45.7	25593	3.86
	FPC-OPT-16	3	10.0	426.7	52386	7.66
	FPC-OPT-32	2	10.0	640.0	71694	3.09
	M-BDI	2	10.0	640.0	19817	18.91
Decomp	BPC-8	9	3.3	47.4	35541	0.93
	BPC-OPT-16	3	10.0	426.7	180259	17.64
	BPC-OPT-32	2	10.0	640.0	268039	119.37
	FPC-8	8	3.3	53.3	24119	0.42
	FPC-OPT-16	3	10.0	426.7	38526	4.51
	FPC-OPT-32	2	10.0	640.0	62693	4.45
	M-BDI	1	10.0	1280.0	7556	6.60

5.1 Compressor and Decompressor Analysis

Table 3 shows the synthesis results for the single compressor and decompressor units of both the optimized and unoptimized designs, including delay, maximum clock frequency, throughput, area, and power consumption.

Compressor Delay: The delay for BPC-8 and FPC-8 is 7 cycles: 4 cycles for pattern matching/word compression, 1 cycle for ZRLE, and 2 cycles for concatenation. The optimized and scaled techniques BPC-OPT-16 and FPC-OPT-16 have 3 cycles delay : 2 cycles for pattern matching and 1 cycle for concatenation. In case of BPC-OPT-32 and FPC-OPT-32, the delay is further reduced to 2 cycles, with 1 cycle for pattern matching and 1 cycle for concatenation.

Compressor Frequency & Throughput: The maximum clock frequency is 2.5 GHz for BPC-8/FPC-8 and primarily limited by ZRLE according to our results. It is increased to 10 GHz for BPC-OPT/FPC-OPT. The maximum throughput is calculated by dividing the block size (128B) by the delay and then multiplying by the maximum frequency. The resulting maximum throughput is: 45.7GB/s for BPC-8 and BPC-8 426.7GB/s for BPC-OPT-16 and FPC-OPT-16, and 640.0GB/s for BPC-OPT-32 and FPC-OPT-32.

Compressor Area: The area cost for the single compressor units is 25050μm^2, 98585μm^2, 156738μm^2 for BPC-8, BPC-OPT-16, BPC-OPT-32, and 25593μm^2, 52386μm^2, 71694μm^2 for FPC-8, FPC-OPT-16, FPC-OPT-32, respectively. The area cost is increased for OPT-16 and OPT-32 due to a higher number of word compressors, and the higher number of parallel inputs for the concatenation logic.

Decompressor Delay: FPC-8 decompressor delay is 8 cycles, which spends 4 cycles on ZRLE, and 4 cycles on word decompression. BPC-8 has 9 cycles delay, with the same steps as FPC-8 and an additional cycle for base decompression. BPC-OPT-16 and FPC-OPT-16 decompressors have 3 cycles delay, 1 cycle is used for the starting position calculation, and 2 cycles for parallel word decompression. BPC-OPT-32 and FPC-OPT-32 have a delay of only 2 cycles, spent on starting position calculation and parallel decompression.

Table 4: Estimated area and power needed to meet the bandwidth of a desktop GPU (GeForce RTX 5090, 1.79 TB/s).

Technique	Units (#)	Area (mm^2)	Power (W)	Area (%)	Power (%)
BPC-8	512	15.85	0.93	2.11	0.16
BPC-OPT-16	32	4.46	0.52	0.59	0.09
BPC-OPT-32	32	6.80	2.11	0.91	0.37
FPC-8	480	11.91	0.97	1.59	0.17
FPC-OPT-16	32	1.45	0.19	0.19	0.03
FPC-OPT-32	32	2.15	0.12	0.29	0.02
M-BDI	32	0.44	0.41	0.06	0.07

Decompressor Frequency & Throughput: The maximum circuit frequency is 3.3 GHz for the unoptimized and 10 GHz for the optimized designs. As for the compressors, the frequency of unoptimized BPC/FPC is limited by ZRLE. The maximum decompressor throughput increases substantially due to the simplifications, namely 426.7GB/s for BPC-OPT-16/FPC-OPT-16, and 640.0GB/s for BPC-OPT-32/FPC-OPT-32.

Decompressor Area: The area cost for the single decompressors is 35541µm^2, 180259µm^2, 268039µm^2 for BPC-8, BPC-OPT-16, BPC-OPT-32, and 24119µm^2, 38526µm^2, 62693µm^2 for FPC-8, FPC-OPT-16, FPC-OPT-32, respectively. The higher area cost of BPC-OPT and FPC-OPT is due to longer shift registers and a greater number of parallel shift registers and word decompressors. FPC sees a much smaller increase due to even word lengths, allowing for better circuit optimization.

5.2 Units, Total Area and Total Power

Table 4 shows the units, area, and power analysis when matching the memory bandwidth of a desktop GPU. For BPC-OPT, FPC-OPT, and M-BDI a (de)compressor clock frequency of 10.5 GHz is assumed (6× GPU MC frequency). BPC-8/FPC-8 require 224/224 compressors and 288/224 decompressors to match the GPU bandwidth (512/480 total units), respectively. In contrast, BPC-OPT-16, BPC-OPT-32, FPC-OPT-16, FPC-OPT-32, and M-BDI only require 16 compressors and 16 decompressors (32 total units). The minimum number of (de)compressors is 16 as bandwidth needs to be matched for each MC individually.

The total area cost is 15.85/4.46/6.80/11.91/1.45/2.15/0.44mm^2 for BPC-8/BPC-OPT-16/BPC-OPT-32/FPC-8/FPC-OPT-16/FPC-OPT-32/M-BDI. Overall, the total area cost for BPC-OPT-16/BPC-OPT-32 is 3.6/2.3× smaller compared to BPC-8 and for FPC-OPT-16/FPC-OPT-32, 8.2/5.5× smaller compared to FPC-8, respectively. The total power consumption is 0.93/0.52/2.11/0.97/0.19/0.12/0.41W for BPC-8/BPC-OPT-16/BPC-OPT-32/ FPC-8/FPC-OPT-16/FPC-OPT-32/M-BDI. Our synthesis analysis shows that the parallel shift registers are major contributors to net switching power, leading to the increased power consumption in BPC-OPT-32.

5.3 Compression Ratio

Figure 5 shows the effective CR of the various compression techniques for different benchmarks. The effective CR refers to the effective amount of data transferred via the memory bus and is calculated by scaling up the raw compressed block size to the nearest multiple of the MAG size. The average effective CR is 1.48/1.48/2.04/2.11/1.92 for FPC/FPC-OPT/BPC/BPC-OPT/M-BDI, respectively. The results

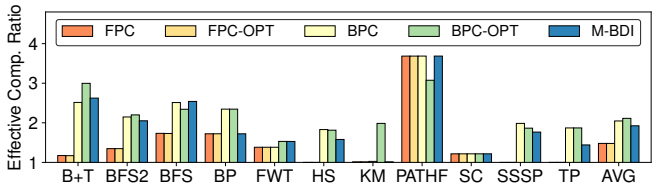


Figure 5: Effective compression ratio of original and optimized FPC/BPC. M-BDI is included for comparison.

highlight that our proposed simplifications do not reduce the effective CR, which is critical for memory bandwidth compression. In case of BPC, the simplifications even increase the CR by 3.4%.

6 CONCLUSIONS

We address the practicality challenges of integrating memory compression in high-throughput GPUs. We analyze the throughput of state-of-the-art techniques like BPC and FPC, identify ZRLE as a bottleneck and propose its removal. The simplification introduces fixed-size tags, increases word-level parallelism, and improves the effective compression ratio of BPC. Our simplifications yield substantial performance gains, enhancing compressor throughput by 14.0× and decompressor throughput by 13.5× without diminishing the effective compression ratio. Moreover, we achieve a significant reduction in hardware requirements, lowering the area cost of BPC by 3.6× and power consumption by 1.8×, making its integration more feasible in high-throughput GPUs. In summary, our simplifications make GPU memory compression more practical and cost-effective for memory-bound applications.

REFERENCES

[1] A. Alameldeen and D. Wood. 2004. Frequent Pattern Compression: A Significance-Based Compression Scheme for L2 Caches. In *University of Wisconsin-Madison*.
[2] X. Chen, L. Yang, R. P. Dick, I. Shang, and H. Lekatsas. 2010. C-PACK: A High-Performance Microprocessor Cache Compression Algorithm. In *IEEE Transactions on VLSI Systems*.
[3] Hoyong Jin, Donghun Jeong, Taewon Park, Jong Hwan Ko, and Jungrae Kim. 2022. Multi-Prediction Compression: An Efficient and Scalable Memory Compression Framework for GP-GPU. *IEEE Computer Architecture Letters* (2022).
[4] J. Kim, M. Sullivan, E. Choukse, and M. Erez. 2016. Bit-Plane Compression: Transforming Data for Better Compression in Many-core Architectures. In *ISCA*.
[5] S. Lal, J. Lucas, and B. Juurlink. 2017. E2MC: Entropy Encoding Based Memory Compression for GPUs. In *IPDPS*.
[6] S. Lal, M. Renz, J. Hartmer, and B. Juurlink. 2022. Memory Access Granularity Aware Lossless Compression for GPUs. In *IPDPS*.
[7] O. Mutlu. 2013. Memory Scaling: A Systems Architecture Perspective. In *IMW*.
[8] O. Mutlu, J. Meza, and L. Subramanian. 2015. The Main Memory System: Challenges and Opportunities. *Carnegie Mellon University* (2015).
[9] M. Renz and S. Lal. 2023. Beyond Compression Ratio: A Throughput Analysis of Memory Compression Techniques for GPUs. In *ICCD*.
[10] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin. 2009. Scaling the Bandwidth Wall: Challenges in and Avenues for CMP Scaling. *SIGARCH Comput. Archit. News* 37, 3 (2009).
[11] V. Sathish, M. J. Schulte, and N. S. Kim. 2012. Lossless and Lossy Memory I/O Link Compression for Improving Performance of GPGPU Workloads. In *PACT*.