A Mapping of Triangular Block Interleavers to DRAM for Optical Satellite Communication

Lukas Steiner†, Timo Lehnigk-Emden*, Markus Fehrenz* and Norbert Wehn†
*Creonic GmbH, Kaiserslautern, Germany
†Microelectronic Systems Design Research Group, University of Kaiserslautern-Landau, Kaiserslautern, Germany
{timo.lehnigk-emden, markus.fehrenz}@creonic.com, {lukas.steiner, norbert.wehn}@rptu.de

***Abstract*—Communication in optical downlinks of low earth orbit (LEO) satellites requires interleaving to enable reliable data transmission. These interleavers are orders of magnitude larger than conventional interleavers utilized for example in wireless communication. Hence, the capacity of on-chip memories (SRAMs) is insufficient to store all symbols and external memories (DRAMs) must be used. Due to the overall requirement for very high data rates beyond 100 Gbit/s, DRAM bandwidth then quickly becomes a critical bottleneck of the communication system. In this paper, we investigate triangular block interleavers for the aforementioned application and show that the standard mapping of symbols used for SRAMs results in low bandwidth utilization for DRAMs, in some cases below 50 %. As a solution, we present a novel mapping approach that combines different optimizations and achieves over 90 % bandwidth utilization in all tested configurations. Further, the mapping can be applied to any JEDEC-compliant DRAM device.**

***Index Terms*—Optical communication, Interleaving, DRAM**

I. Introduction and Related Work

Interleaving is a key method in communication systems to correct burst errors that occur during transmission. In free-space optical communication from *low earth orbit* (LEO) satellites to earth, huge amounts of data have to be transmitted via lasers in a short time frame. This leads to data rate requirements beyond 100 Gbit/s which have to be decoded in real time. Due to high dynamics in channel quality and a long coherence time of the optical communication channel (> 2 ms), large interleavers are necessary to enable reliable transmission at target code rates. A suitable interleaver type for this application is the *triangular block interleaver* where the symbols of multiple consecutive code words are first *written* to a triangular (upper left half of a square) storage array in *row-wise* order and afterwards *read* from the array in *column-wise* order. These interleavers are usually implemented in SRAM by corresponding memory addressing. As their size exceeds the storage capacity of SRAM in the present application scenario, DRAM must be used instead. Unlike SRAM, however, the achievable transfer bandwidth of DRAM depends strongly on the access pattern. Patterns with few bit toggles between consecutive addresses (e.g., sequential or strided) enable high bandwidth utilization, while patterns with many toggles (e.g., random) decrease bandwidth utilization significantly due to frequent page misses. If the two-dimensional index space of a triangular block interleaver is mapped to linear DRAM storage in row-major order as in the case of an SRAM implementation, the write phase will result in a sequential pattern (high bandwidth utilization), but the read phase will result in frequent page misses (low bandwidth utilization). Since the maximum possible throughput of the interleaver is not determined by the average, but by the minimum DRAM bandwidth utilization across both access phases, the theoretical maximum bandwidth of a DRAM configuration must be largely oversized (faster speed grade or wider data bus) to match the requirements of the communication system. This leads to higher costs and additional energy consumption.

In previous research, several optimized mappings of two-dimensional index spaces to DRAM for alternate row- and column-wise accesses have been proposed [1]–[9]. However, none of the works considered the impact of bank groups, which have been introduced in all recent JEDEC DRAM standards (e.g., DDR4/5, LPDDR5, HBM).

In this work, we present a novel general approach to map the two-dimensional index space of triangular block interleavers to any JEDEC-compliant DRAM device. By combining three optimizations, we eliminate the major limiting factors for DRAM bandwidth utilization in both access directions. We prove the effectiveness of our mapping with cycle-accurate simulations of five different DRAM standards and two different speed grades per standard.

II. Optimized Mapping

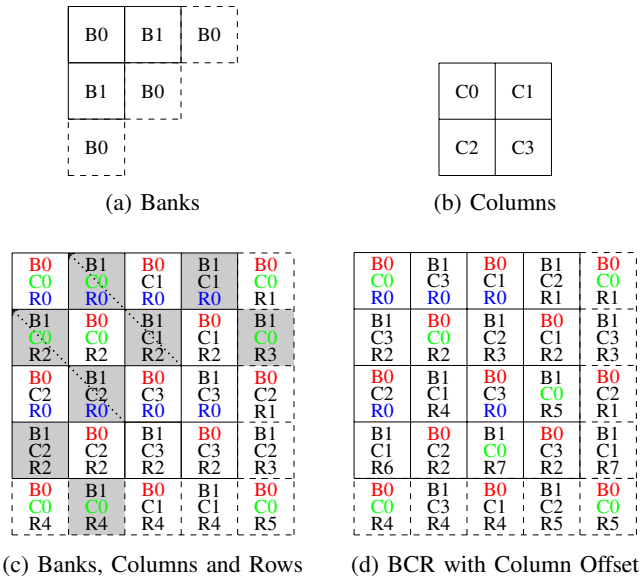
It is important to note that the number of bits accessed with a single DRAM burst is much larger than one symbol (e.g., 512 bits vs. 3 bits). In our application, interleaving is therefore divided into two stages, and a small SRAM block interleaver first ensures that symbols within a DRAM burst belong to different code words. The optimized mapping assigns each position of the two-dimensional index space of the interleaver to a DRAM address composed of a bank, column and row. In order to handle standards both with and without bank groups using the same mapping, it is presumed that the lower bank address bits always denote the bank group.

For a DRAM standard with bank groups, the full bandwidth can only be utilized if the **bank group is switched with each access** in a round-robin order. This is achieved by incrementing the bank index by one with each access in both directions, which leads to the diagonal pattern shown in Fig. 1a exemplarily for two banks.

Irrespective of the DRAM standard, **page misses should be minimized** for high bandwidth utilization. To divide them

This work was partly funded by the German Federal Ministry of Education and Research (BMBF) under grant 16KIS1608 (FACTOR).

979-8-3503-4859-0/DATE24/© 2024 EDAA



(a) Banks (b) Columns

(c) Banks, Columns and Rows (d) BCR with Column Offset

Fig. 1: Optimized Mapping Schemes

equally between both access directions, the two-dimensional index space is partitioned into small rectangles, each of which is assigned to a DRAM page. This mapping is shown in Fig. 1b for a page with four columns.

Combining both optimizations and adding DRAM rows leads to the mapping shown in Fig. 1c. For reasons of space, only the top left quarter of the index space is presented. Bank 0, column 0 and row 0 are colored for a better understanding. One final problem is that the remaining page misses occur almost simultaneously on all banks, which prevents a page miss on one bank from being effectively masked by page hits on other banks. As a solution, **page misses on different banks are staggered** by introducing a bank-dependent column offset. Simply put, all positions of the index space are shifted to the top left by an offset that depends on the bank they are mapped to. In Fig. 1c, all positions mapped to bank 1 are shifted by two (offset of one column in both directions, dotted arrows), while the offset is zero for positions mapped to bank 0. In this way, the first page miss on bank 1 occurs one access earlier than on bank 0. The shifting is done circular, i.e., positions with a gray background are moved to the bottom and right border, respectively. This leads to the final mapping shown in Fig. 1d.[1] Due to space limitations, the mapping rules are not elaborated further, but they only consist of additions, logical shifts and bitwise operations, which enables a hardware implementation with low complexity.

III. Experimental Results

To quantify the improvements, multiple configurations are simulated with the cycle-accurate DRAM simulator DRAMSys [10]. Table I shows the achieved bandwidth utilizations for a triangular block interleaver with 12.5 M elements.

[1]Figs. 1c and 1d show a rectangular index space, which would result in a storage overhead of more than 50 % for a triangular block interleaver. In reality, the number of DRAM rows decreases from the top to the bottom of the index space, which allows a more storage-efficient mapping.

TABLE I: DRAM Bandwidth Utilizations

DRAM Configuration	Row-Major Mapping		Optimized Mapping	
	Write	Read	Write	Read
DDR3-800	**95.99 %**	96.03 %	**95.99 %**	96.26 %
DDR3-1600	95.75 %	**64.16 %**	**95.91 %**	96.16 %
DDR4-1600	92.02 %	**73.92 %**	**92.01 %**	92.37 %
DDR4-3200	91.83 %	**43.50 %**	**91.86 %**	92.15 %
DDR5-3200	100.00 %	**96.37 %**	**100.00 %**	100.00 %
DDR5-6400	99.90 %	**88.95 %**	**99.83 %**	99.97 %
LPDDR4-2133	99.02 %	**66.00 %**	99.41 %	**98.30 %**
LPDDR4-4266	98.03 %	**35.77 %**	**99.67 %**	99.72 %
LPDDR5-4267	99.39 %	**55.87 %**	**99.77 %**	100.00 %
LPDDR5-8533	97.56 %	**47.25 %**	**99.14 %**	99.66 %

Results for other interleaver dimensions are omitted for space reasons because they differ only slightly. The bandwidth utilization is separated into write (row-wise) and read (column-wise) phase since the minimum of both (bold) determines the maximum throughput of the interleaver. Small variations within each phase are balanced by buffers in the memory controller.

The optimized mapping achieves significant improvements in bandwidth utilization over the row-major mapping for the majority of configurations, in some cases from below 50 % to over 90 %. Especially the faster speed grade device of each standard and LPDDR devices benefit greatly. The missing percentages of the optimized mapping are caused by refresh commands. When refresh is disabled, which is legal as long as the maximum lifetime of the interleaver data is smaller than the DRAM refresh period (between 32 ms and 64 ms), a bandwidth utilization of over 99 % is consistently achieved. As a side note, the proposed optimizations can also be applied to different applications with similar DRAM access patterns.

References

[1] L. Si and C. Chen, "Implementation of time-domain interleaver based on FPGA in DTTB," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, 2012, pp. 3284–3287.

[2] J. Zhou, Y. Dou, Y. Lei, and Y. Dong, "Window memory accesses method in alternate row/column matrix access systems," in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 3, 2010, pp. V3–201–V3–205.

[3] M. Garrido and P. Pirsch, "Continuous-flow matrix transposition using memories," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 9, pp. 3035–3046, 2020.

[4] B. Akin, F. Franchetti, and J. C. Hoe, "Data reorganization in memory using 3D-stacked DRAM," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, 2015, pp. 131–143.

[5] H. Kim and I.-C. Park, "Array address translation for SDRAM-based video processing applications," in *Visual Communications and Image Processing 2000*, vol. 4067. SPIE, 2000, pp. 922 – 931.

[6] B. Akin, P. A. Milder, F. Franchetti, and J. C. Hoe, "Memory bandwidth efficient two-dimensional fast fourier transform algorithm and implementation for large problem sizes," in *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, 2012, pp. 188–191.

[7] S. Langemeyer, P. Pirsch, and H. Blume, "Using SDRAM memories for high-performance accesses to two-dimensional matrices without transpose," *International Journal of Parallel Programming*, vol. 41, 04 2012.

[8] V. Prasanna, "DRAM row activation energy optimization for stride memory access on FPGA-based systems," 04 2015, pp. 349–356.

[9] S. Singapura and V. Prasanna, "Optimal dynamic data layouts for 2D FFT on 3D memory integrated FPGA," vol. 9251, 08 2015, pp. 338–348.

[10] L. Steiner, M. Jung, F. S. Prado, K. Bykov, and N. Wehn, "DRAMSys4.0: A fast and cycle-accurate SystemC/TLM-based DRAM simulator," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Springer International Publishing, 2020, pp. 110–126.