ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.13, No.14, 1–10

Exploring new features of high-bandwidth memory for GPUs

Bingchao Li[1], Choungki Song[2], Jizeng Wei[1], Jung Ho Ahn[3a)], and Nam Sung Kim[4b)]

[1] *School of Computer Science and Technology, Tianjin University*

[2] *Department of Electrical and Computer Engineering, University of Wisconsin-Madison*

[3] *Graduate School of Convergence Science and Technology, Seoul National University*

[4] *Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign*

a) *gajh@snu.ac.kr*

b) *nskim@illinois.edu*

Abstract: Due to the off-chip I/O pin and power constraints of GDDR5, HBM has been proposed to provide higher bandwidth and lower power consumption for GPUs. In this paper, we first provide detailed comparison between HBM and GDDR5 and expose two unique features of HBM: dual-command and pseudo channel mode. Second, we analyze the effectiveness of these two features and show that neither notably contributes to performance. However, by combining pseudo channel mode with cache architecture supporting fine-grained cache-line management such as *Amoeba* cache, we achieve high effciency for applications with irregular memory requests. Our experiment demonstrates that compared with *Amoeba* caches with legacy mode, *Amoeba* cache with pseudo channel mode improves GPU performance by 25% and reduces HBM energy consumption by 15%.

Keywords: GPU, HBM, DRAM, Amoeba cache, energy

Classification: Integrated circuits

References

[1] JEDEC: Graphic Double Data Rate 5 (GDDR5) Specification (2009) https://www.jedec.org/standards-documents/results/taxonomy%3A4229.

[2] JEDEC: High Bandwidth Memory (HBM) DRAM (2013) https://www.jedec.org/standards-documents/results/jesd235.

[3] S. Kumar, *et al.*: "Amoeba-cache: Adaptive blocks for eliminating waste in the memory hierarchy," MICRO (2012) 484 (DOI: 10.1109/MICRO.2012.42).

[4] B. He, *et al.*: "Mars: A mapreduce framework on graphics processors," PACT (2008) 260 (DOI: 10.1145/1454115.1454152).

[5] K. P. M. Burtscher and R. Nasre: "A quantitative study of irregular programs on gpus," IISWC (2012) 141 (DOI: 10.1109/IISWC.2012.6402918).

[6] Q. Xu, *et al.*: "Graph processing on gpus: Where are the bottlenecks?" IISWC (2014) 140 (DOI: 10.1109/IISWC.2014.6983053).

DOI: 10.1587/elex.13.20160527
Received May 24, 2016
Accepted June 1, 2016
Publicized June 28, 2016
Copyedited July 25, 2016

[7] Hynix: http://www.hynix.com/datasheet/pdf/graphics/H5GQ1H24AFR(Rev1.0).pdf.
[8] Micron: SDRAM power calculator https://www.micron.com/~/media/documents/products/power-calculator/sdram_power_calc_10.xls.
[9] B. Wang, *et al.*: "Exploring hybrid memory for gpu energy efficiency through software-hardware co-design," PACT (2013) 93.
[10] J. Ahn, *et al.*: "Multicore dimm: an energy effcient memory module with independently controlled drams," IEEE Comput. Archit. Lett. **8** (2009) 5 (DOI: 10.1109/L-CA.2008.13).
[11] Y. H. Son, *et al.*: "Microbank: Architecting through-silicon interposer-based main memory systems," SC (2014) 1059 (DOI: 10.1109/SC.2014.91).
[12] T. Zhang, *et al.*: "Half-dram: A high-bandwidth and low-power dram architecture from the rethinking of fine-grained activation," ISCA (2014) 349 (DOI: 10.1109/ISCA.2014.6853217).
[13] J. Ahn, *et al.*: "Improving system energy effciency with memory rank subsetting," ACM Trans. Archit. Code Optim. **9** (2012) 4 (DOI: 10.1145/2133382.2133386).
[14] M. Rhu, *et al.*: "A locality-aware memory hierarchy for energy-effcient GPU architectures," MICRO (2013) 86 (DOI: 10.1145/2540708.25407170).
[15] N. D. Gulur, *et al.*: "Multiple sub-row buffers in dram: Unlocking performance and energy improvement opportunities," ICS (2012) 257 (DOI: 10.1145/2304576.2304613).
[16] L. Barroso, *et al.*: "Piranha: a scalable architecture based on single-chip multiprocessing," ISCA (2000) 282 (DOI: 10.1145/339647.339696).
[17] D. Kaseridis, *et al.*: "Minimalist open-page: A dram page-mode scheduling policy for the many-core era," MICRO (2011) 24 (DOI: 10.1145/2155620.2155624).

1 Introduction

Modern graphics processing units (GPUs) are widely used for high performance computing. As GPU architecture evolves to simultaneously run more threads, they require higher bandwidth and larger capacity to supply necessary data for those threads. For instance, the memory bandwidth of a previous generation GPU (GTX680) is 192 GB/s, whereas that of a current generation GPU (GTX980) is 224 GB/s. GDDR5 [1], a contemporary GPU-DRAM interface, transfers data at a high data rate for high bandwidth. However, it does so at the cost of high power consumption and it is challenging to further increase a data transfer rate due to rapidly deteriorating signal integrity and package power/pin constraint. Besides, GDDR5 employs a point-to-point connection (i.e., one memory module per channel) for a high data transfer rate. This makes hard to offer larger capacity per memory module as we will soon face the end of technology scaling. To further increase memory bandwidth, capacity, or both, more memory channels can be provided, but such an approach is not scalable under package power/pin constraint.

Responding to such challenges, DRAM manufacturers proposed HBM [2], an emerging memory technology that 3D-stacks multiple DRAM dies using through silicon vias and 2.5-stacks memory modules and a GPU using silicon interposers instead of off-chip interconnects. Consequently, HBM can provide higher bandwidth, lower power, and larger capacity than GDDR5.

DOI: 10.1587/elex.13.20160527
Received May 24, 2016
Accepted June 1, 2016
Publicized June 28, 2016
Copyedited July 25, 2016

Emerging GPGPU applications often have irregular data structures, entailing branch and memory divergences. A branch divergence occurs when threads within a warp jump to divergent paths. A memory divergence arises when threads of a warp access different cache lines. Due to branch and memory divergences, GPUs frequently place many irregular small memory requests that cannot be coalesced into a single memory request to a single large cache-line containing data from consecutive addresses.

In this paper, we first provide detailed comparison between HBM and GDDR5, and expose two unique features of HBM: dual-command and pseudo channel mode. Each HBM channel supports separate row and column command buses and this dual-command feature allows the memory controller to simultaneously issue row and column commands for two different banks. Each 128-bit (legacy) HBM channel can be split into two 64-bit (pseudo) channels sharing the row and column command buses, and HBM in pseudo channel mode doubles the number of banks, each of which has halved size of pages (row buffers) compared with HBM in legacy mode.

Second, analyzing the effectiveness of these two features, we show that HBM with these two features do not notably contribute to improving performance, compared with HBM without these two features. More specifically, we observe that the dual-command bus does not improve the performance because the percentage of simultaneously issuable row and column commands is not high enough to benefit from it. Besides, the halved size of row buffers negates the benefit of doubled channels in pseudo channel mode, often entailing marginal improvement (or even degradation) of performance.

Lastly, considering the aforementioned characteristics of emerging GPGPU applications, we propose to combine the pseudo channel mode with on-chip cache architecture that also supports fine-grained cache-line management and accesses such as *Amoeba* cache. The advantage of pseudo channel mode is utilized by this combination. Our experiment shows that compared with a GPU with *Amoeba* caches and legacy channel mode, a GPU with *Amoeba* cache and pseudo channel mode can improve performance and reduce energy consumption of HBM by 25% and 15%, respectively.

2 Architectural comparison between GDDR5 and HBM

Compared with GDDR5, HBM supports two modes: legacy and pseudo channel modes. An HBM device in legacy mode offers eight 128-bit channels, whereas a ×32 GDDR5 device provides one 32-bit channel. A 128-bit channel in HBM is comprised of two 64-bit (pseudo) channels, each being connected to 8 physical banks (Fig. 1). Two physical banks from two 64-bit (pseudo) channels (e.g., `B0` and `B0'` from `PC0` and `PC1` in Fig. 1) constitute one logical bank and they are simultaneously accessed to read or write 256-bit data in legacy mode. A 128-bit channel in HBM transfers 32 bytes over a cycle (BL of 2), whereas a 32-bit channel in GDDR does so over four cycles (BL of 8). In pseudo channel mode, two 64-bit channels forming a 128-bit channel and its 8 logical banks in legacy mode become

DOI: 10.1587/elex.13.20160527
Received May 24, 2016
Accepted June 1, 2016
Publicized June 28, 2016
Copyedited July 25, 2016

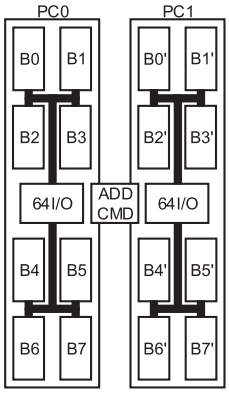
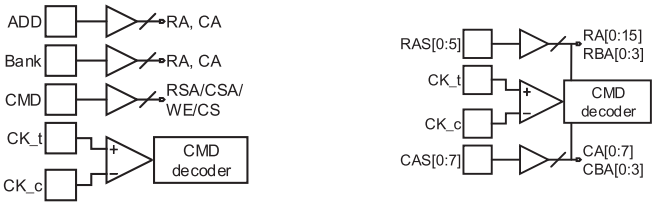


Fig. 1. HBM overview.



(a) Single-command. (b) Dual-command.

Fig. 2. Single-command versus dual-command.

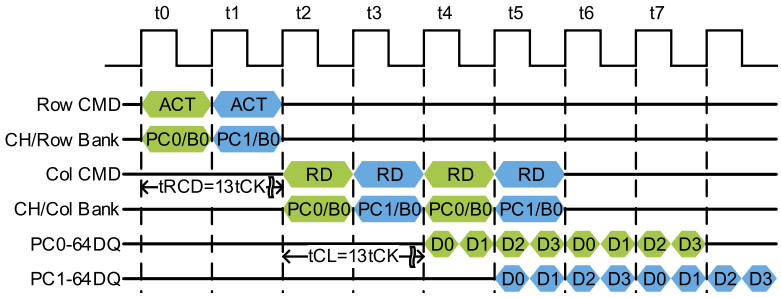


Fig. 3. Timing diagram of pseudo channel mode

independent (i.e., 8 banks per 64-bit pseudo channel), the page size of each bank is reduced to 1 KB, the BL is increased to 4 to transfer 32 bytes (Fig. 3).

HBM has separate pins for row and column commands, whereas the conventional GDDR5 shares the pins for row and column commands, as illustrated in Fig. 2(a) and (b). This dual command feature allows HBM to simultaneously issue row and column commands to two different banks in both legacy and pseudo channel modes. For example, as shown in Fig. 3, an `ACT` to `PC1` and a `RD` to `PC0` can be issued together at `t2` in pseudo channel mode. `PC0` cannot issue column command at `t3` due to `tCCD` timing constraint, whereas a `RD` to `PC1` can be issued. At `t4`, `PC0` and `PC1` transfer their data simultaneously. The intuition behind introducing the dual-command feature is that HBM is expected to consume higher command bandwidth as it has shorter BL than GDDR5. Lastly, the pseudo channel mode defines `tEAW` (eight activate window) instead of `tFAW` (four activate window)

DOI: 10.1587/elex.13.20160527
Received May 24, 2016
Accepted June 1, 2016
Publicized June 28, 2016
Copyedited July 25, 2016

of legacy mode, which can issue more activations during tEAW. The architectural features of GDDR5 and HBM are summarized in Table I.

3 Adapting GPU cache architecture to exploit HBM

In NVIDIA GPUs 32 threads within a warp may generate up to 32 4-byte memory requests for a load or store instruction. Such memory requests can be coalesced into a single memory request when they access the same 128-byte address space in a 128-byte cache line. However, not all 4-byte words within a 128-byte cache line are accessed by threads in particular with irregular memory requests. When categorizing memory requests into 16-, 32-, 64-, and 128-byte ones, we observe that most memory requests are 16-byte ones (83% on average) in applications with irregular memory requests (Fig. 4); see Section 4 for our detailed experimental methodology.

As demonstrated, emerging GPGPU applications mostly place 16-byte memory requests. While HBM can service 32 bytes per transaction, it always ends up servicing 128 bytes per request (i.e., four 32-byte transactions per 128-byte request). This is because each miss of conventional GPU cache leads to a 128-byte memory request (i.e., the size of a cache line).

To effectively utilize cache capacity and memory bandwidth for these emerging GPGPU applications, we propose to combine the pseudo channel mode with *Amoeba* cache [3] that can support both fine- and coarse-grained (32-, 64-, and 128-byte) cache-line sizes for GPU's L1 and L2 caches and accesses. *Amoeba* cache is originally proposed to support variable cache-line sizes, and it stores extra tags in the cache data-array and uses a tag-bitmap to indicate which words in the data array represent the tags. Tags are used to index fixed-size regions within cache lines and the tag field of each region not only contains tag bits of this region but

Table I. Parameters of GDDR5 and HBM

	GDDR5	HBM	
		Legacy mode	Pseudo channel
Channel/chip	0.5	2	4
Page size	2 KB	2 KB	1 KB
Prefetch	8n	2n	4n
Burst Length	8	2	4
DQ/channel	32	128	64
Banks/Bank Group	4	4	4
Banks/channel	16	8	8
CMD interface	once at a time	semi independent	
CMD input	1 cycle	ACT needs 2 cycles, others need 1 cycle	
Per bank Refresh	X	O	O
DBI	DBI-DC	DBI-AC	DBI-AC
Reliability	CRC	ECC (check by host)	

DOI: 10.1587/elex.13.20160527
Received May 24, 2016
Accepted June 1, 2016
Publicized June 28, 2016
Copyedited July 25, 2016

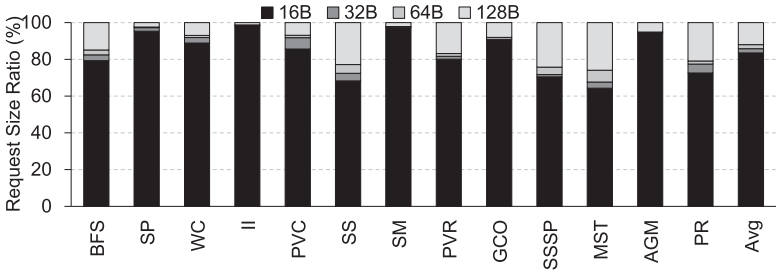


Fig. 4. Request size breakdown.

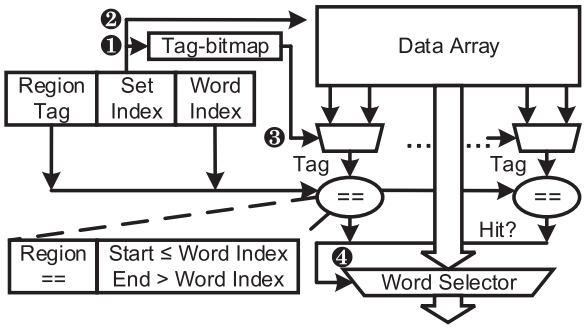


Fig. 5. *Amoeba* cache architecture.

also *start* and *end* bits to specify small cache lines within this region (Fig. 5). When *Amoeba* cache services a request, its set index is used as the address of the tag-bitmap (Fig. 5❶) and data array (Fig. 5❷). Because adjacent words cannot be both tags, $N/2$ multiplexers controlled by the tag-bitmap signal (Fig. 5❸) are required to route one of the adjacent words to the comparators, where N is the number of words in one set. Then the hit signal generated by the comparators is used as the word selector. In Section 5, we will analyze why *Amoeba* cache with pseudo channel mode performs far better than *Amoeba* cache with legacy channel mode.

4 Methodology

4.1 GPU performance

We use GPGPU-Sim version 3.2.2 for our evaluation. The key configuration parameters are tabulated in Table II. In this paper we evaluate emerging memory-intensive GPGPU benchmarks such as graph applications. More specifically, we evaluate benchmarks chosen from *Mars* [4] (Page View Count (`PVC`), Page View Rank (`PVR`), Similarity Score (`SS`), Word Count (`WC`), Inverted Index (`II`), and String Match (`SM`)); *Lonestar* [5] (Breadth First Search (`BFS`), Survey Propagation (`SP`), Minimum Spanning Tree (`MST`), and Single Source Shortest Path (`SSSP`)); and a graph benchmark suite [6] (Approximate Graph Matching (`AGM`), Graph Coloring (`GC`), and Page Rank (`PR`)).

DOI: 10.1587/elex.13.20160527
Received May 24, 2016
Accepted June 1, 2016
Publicized June 28, 2016
Copyedited July 25, 2016

Table II. GPGPU-Sim configuration

Number of SM	15
Maximum Threads per SM	1536
L1 Data Cache per SM	16 KB
Number of Memory Channels	8
L2 Cache per Memory Channel	128 KB
Compute Core Clock	1000 MHz
Interconnect Clock	1000 MHz
Memory Clock	1000 MHz
Memory Controller	FRFCFS
Warp Scheduling Policy	Greedy then oldest
HBM Memory Timing	$t_{CL} = 13$ $t_{RP} = 11$ $t_{RC} = 40$ $t_{RAS} = 29$ $t_{RCD} = 13$ $t_{RRD} = 5$

4.2 HBM energy

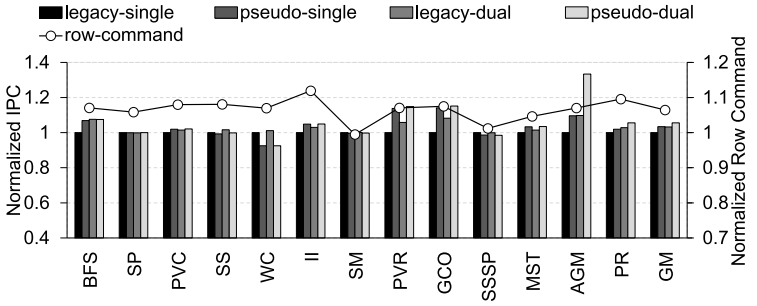
Similar to traditional DRAM, HBM has four power components often denoted by `IDD0`, `IDD2`, `IDD4R`, and `IDD4W`. Each component addresses how much current is consumed when specific operation sequences are performed. As `IDD0` is proportional to page size and HBM in legacy and pseudo channel modes has 2× and 4× smaller page size than GDDR, we assume HBM consumes 2 and 4× smaller `IDD0`, respectively. `IDD2` is proportional to capacity and HBM has 8× smaller capacity per channel than GDDR5, so we assume HBM consumes 8× less `IDD2`. `IDDR4R` and `IDDR4W` are associated with read and write operations, respectively, and proportional to the number of transferred bytes per burst. Thus, we assume HBM per channel consumes the same `IDDR4R` and `IDDR4W` as a GDDR5 ×32 device [7], as one HBM channel and a GDDR5 ×32 device transfer the same number of bytes per busrt (i.e., 32 bytes). Lastly, we calculate the total energy consumption of HBM after we plug the estimated `IDD` values and memory command traces from GPGPU-Sim into an SDRAM power calculator [8].

5 Experimental results

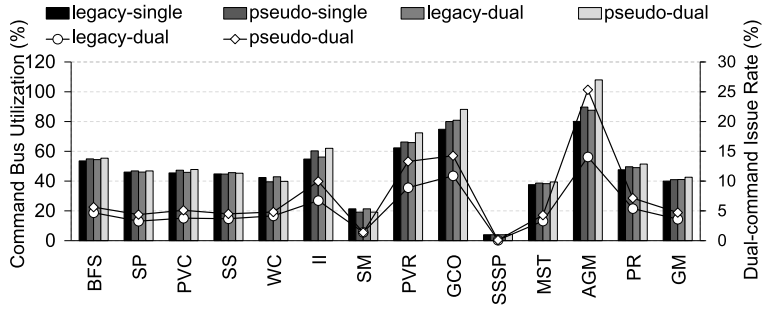
5.1 GPU performance

Fig. 6(a) plots the instructions per cycle (IPC) of a GPU with four HBM configurations: `legacy-single`, `pseudo-single`, `legacy-dual`, and `pseudo-dual`, where `legacy` (`pseudo`) and `single` (`dual`) denote legacy (pseudo) channel mode and single-command (dual-command) channel, respectively, normalized to that of `legacy-single`; we use *standard cache architecture* for all four HBM configurations for Fig. 6(a). First, `legacy-dual` and `pseudo-dual` offer only 3% and 2% higher geo-mean performance than `legacy-single` and `pseudo-single`, as consuming only 41% and 43% of command bus bandwidth and exhibiting only

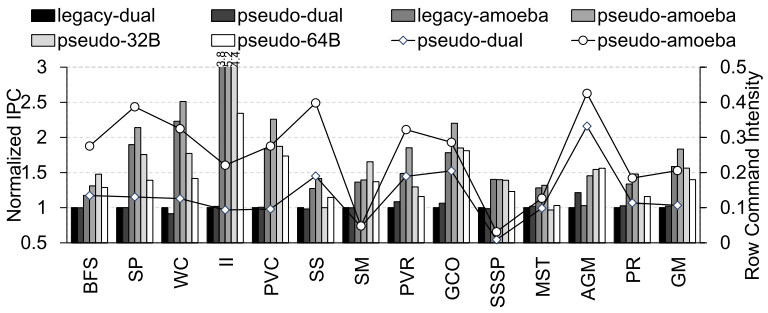
DOI: 10.1587/elex.13.20160527
Received May 24, 2016
Accepted June 1, 2016
Publicized June 28, 2016
Copyedited July 25, 2016



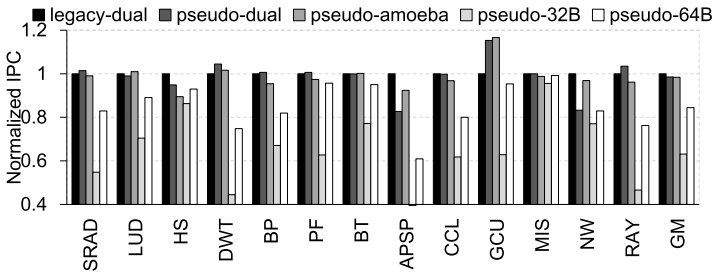
(a) Performance comparison between single- and dual-command feature.



(b) Command bus utilization and dual-command issue rate.



(c) Performance comparison among legacy, pseudo, legacy-amoeba, pseudo-amoeba, pseudo-32B and pseudo-64B configurations.



(d) Performance comparison for regular applications.

Fig. 6. Performance and command bus bandwidth utilization.

3.6% and 4.7% of commands are simultaneously issuable, respectively, as plotted in Fig. 6(b). Furthermore, compared to `*-single`, `*-dual` can reduce the latency by only one cycle for a pair of simultaneously issuable row and column commands, which can be easily absorbed by latency-tolerant nature of GPU architecture. The only exception is `AGM`, where `legacy-single` and `pseudo-single` consume 88% and 108% more command bus bandwidth with 14% and 25% simultaneously issuable commands, respectively.

DOI: 10.1587/elex.13.20160527
Received May 24, 2016
Accepted June 1, 2016
Publicized June 28, 2016
Copyedited July 25, 2016

Second, pseudo-dual provides only 2% higher geo-mean performance than legacy-dual. As stated earlier, the page size of pseudo is only a half of that of legacy (6% more geo-mean row-buffer misses as shown in Fig. 6(a)), negating the benefit of more memory channels (and even degrading performance of WC).

Lastly, Fig. 6(c) plots the IPC of a GPU with six HBM configurations: legacy, pseudo, legacy-amoeba, pseudo-amoeba, pseudo-32B (32-byte cache line size), and pseudo-64B (64-byte cache line size), normalized to that of legacy; we assume the dual-command channel for all six configurations. As *Amoeba* cache improves cache utilization (higher hit rates), legacy-amoeba and pseudo-amoeba give 58% and 81% higher geo-mean performance than legacy and pseudo, respectively. Although pseudo offers geo-mean performance similar to legacy, pseudo-amoeba gives 25% higher geo-mean performance than legacy-amoeba. That is, *Amoeba* cache synergistically interacts with the pseudo channel model, as *Amoeba* cache generates more memory requests (with less spatial locality) at shorter execution time, which in turn leads to more row commands per unit time (i.e., higher intensity of row commands, as shown in Fig. 6(c)). In such a case, the pseudo channel mode, which doubles the number of channels (and halves the size of row buffers), can notably contribute to higher GPU performance.

Note that pseudo-32B and pseudo-64B supporting only fine-grained cache accesses perform 27% and 43% worse than pseudo-amoeba offering both fine- and coarse-grained (i.e., 32- and 128-byte) cache accesses, as even the benchmarks with irregular memory requests have a notable fraction of 128-byte cache accesses (cf. Fig. 4) and smaller cache line sizes incur more cache misses and hence result in more requests in the memory system, which takes up more hardware resources. The drawbacks of smaller cache line can even hurt the performance of benchmarks with regular memory requests. Fig. 6(d) shows that the performance of pseudo-amoeba exhibits only 1% degradation for benchmarks with regular memory requests. However, pseudo-32B and pseudo-64B perform 39% and 15% worse than the baseline, respectively.

5.2 HBM energy

Fig. 7 plots the HBM energy consumption and its breakdown of legacy, legacy-amoeba, pseudo, and pseudo-amoeba, normalized to those of legacy. Although pseudo did not offer notably higher performance than legacy, it consumes 16% lower energy. This is because the page size of pseudo (and pseudo-amoeba) is a half of that of legacy (and legacy-amoeba), notably reducing ACT and PRE

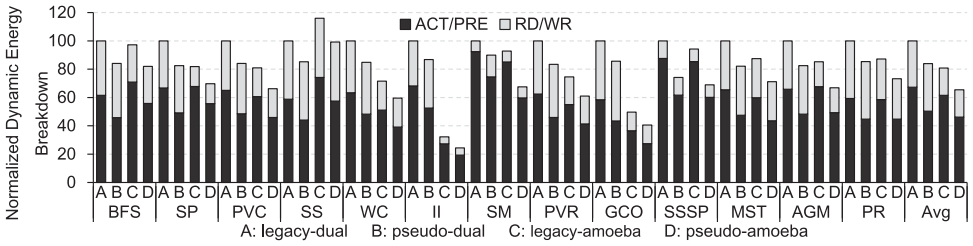


Fig. 7. HBM energy consumption and its breakdown normalized to Legacy

DOI: 10.1587/elex.13.20160527
Received May 24, 2016
Accepted June 1, 2016
Publicized June 28, 2016
Copyedited July 25, 2016

energy consumption (i.e., IDD0). HBM with *Amoeba* cache (i.e., `legacy-amoeba` and `pseudo-amoeba`) consumes 13% less RD and WR geo-mean energy than HBM with conventional cache, as *Amoeba* cache mostly incurs fewer bursts than conventional cache (mostly one burst (32 bytes) versus always four bursts (128 bytes per miss)).

6 Related work

Previous works propose to replace GDDR5 with PCM to reduce the memory power and increase the density [9]. However, PCM suffers from limited write operations and higher write energy. Fine-grained DRAM structures are also proposed to reduce the activate/precharge power by shrinking the width of each bank [10, 11, 12, 13]. They may lead to lower row buffer hit rates due to the narrower row size, but better bank-level parallelism. These works only focus on the internal architecture of DRAM rather than how to further leverage the bank-level parallelism from the prospective of request intensity. In [14], sub-ranked GDDR5 is adopted to support the fine-grained requests that access GDDR5. However, the bank-level parallelism of the sub-ranked GDDR5 is not explored by this work. [15] proposes to improve the row buffer utilization by splitting a large row buffer into multiple sub-row buffers. However, it does not eliminate the timing constrains of row commands among sub-rows. There are also studies devoting to the design of DRAM controllers to determine whether to leave a row buffer open or closed [16, 17].

7 Conclusion

HBM has higher bandwidth and lower power consumption than GDDR5, making it a good candidate for future memory systems for GPUs with extremely high bandwidth requirement. In this paper, we first compare HBM with GDDR5 in detail, analyze two unique features (i.e., dual-command channel and pseudo channel mode), and demonstrate that these features alone are not effective to improve performance. Second, we propose to combine the HBM pseudo channel mode with *Amoeba* cache, offering 83% (25%) higher geo-mean GPU performance and 18% (15%) lower HBM energy consumption than the HBM pseudo channel mode with conventional cache (the HBM legacy channel mode with *Amoeba* cache), respectively.

Acknowledgments

This research was supported in part by grants from NSF (CNS-1217102), Samsung (2016-03853), Natural Science Foundation of China (61402321) and Natural Science Foundation of Tianjin (15JCQNJC00100). Nam Sung Kim has a financial interest in AMD and Samsung Semiconductor. Nam Sung Kim and Jung Ho Ahn are the co-corresponding authors.

DOI: 10.1587/elex.13.20160527
Received May 24, 2016
Accepted June 1, 2016
Publicized June 28, 2016
Copyedited July 25, 2016