Modeling HBM2 Memory Controller

Ayaz Akram*, Maryam Babaie*, Wendy Elsasser†, and Jason Lowe-Power*
*University of California, Davis, and †Rambus Inc.

gem5 Users' Workshop associated with ISCA 2022

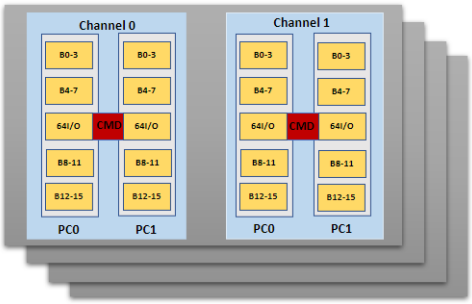

Fig. 1: 4H HBM2 stack (4 DRAM dies). CMD: shared command bus, PC0 & PC1: 2 pseudo channels.

High bandwidth memory (HBM) provides much higher bandwidth at lower power than standard DDR memories (e.g., DDR4) and is thus considered necessary for emerging applications. HBM2 is the latest standard of HBM with actual market products (the JEDEC standard of HBM3 is final).

HBM2 [1] comes as a 4H (4 DRAM dies) or 8H (8 DRAM dies) stack, where each DRAM die has two physical channels and a capacity of 1GB (8Gb). Each physical channel of the DRAM die can act as a 128-bit interface (in HBM legacy mode) or divide into two pseudo channels (in pseudo-channel mode). HBM2 supports an independent address and command bus, shared among both pseudo-channels. However, both pseudo-channels have separate data buses. A 4H stack of HBM2 (shown in Fig. 1) can provide 256GB/s bandwidth, whereas a single physical channel contributes 32GB/s.

A. Limitations of current gem5 memory modeling

Currently, gem5 [1] provides minimal HBM support. The existing HBM interface in gem5 [1] follows HBM1 specifications. Moreover, the interface does not offer asymmetric timing parameters for reads and writes (the case in HBM2).

The most critical limitation of gem5 is that it supports only a single DRAM interface per memory controller, i.e., it cannot support HBM2 in pseudo-channel mode. Using an independent controller per pseudo channel can lose the impact of the shared command bus and any impact of the shared queue for both pseudo channels (as in some real HBM2 devices). The default memory controller of gem5 allows using DRAM and NVM interfaces together, but they have to share a data bus.

[1]https://www.anandtech.com/show/9969/jedec-publishes-hbm2-specification

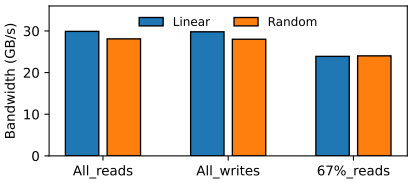

Fig. 2: Bandwidth of a physical channel (two pseudo channels) of HBM2 controlled by a single gem5 memory controller

B. Enabling HBM2 support in gem5

We tried to overcome the above-mentioned issues to enable pseudo channel mode HBM2 in gem5. We relied on JEDEC documentation and other simulation tools to configure an HBM2 interface in gem5. We updated the DRAM interface to support separate tRCD and tCL for reads and writes.

Most importantly, we added an HBM-specific memory controller capable of controlling two DRAM interfaces and supporting the use of a shared or partitioned (among pseudo channels) read and write queue. Our address mapping policy interleaves the memory requests across pseudo channels at a granularity of 64B (two 32B atoms for a 64B cache line go to the same pseudo channel). gem5 had added the support for multi-cycle commands and checks for command bandwidth in the gem5-20 release. We extended this support to check row and column command bandwidth separately. We fixed some minor but essential bugs on bandwidth verification checks.

We observed an extra hit on bandwidth in the read/write traffic mix and, therefore, implemented a *min_reads_per_switch* similar to *min_writes_per_switch* (already implemented in gem5) to ensure a lower number of read/write turnarounds.

C. Evaluation

We performed traffic generator-based studies using different linear/random read/write traffic combinations to validate the HBM2 pseudo channel model. Figure 2 presents the bandwidth numbers for a single physical HBM2 channel (made of two pseudo-channels), controlled by an HBM controller (peak theoretical bandwidth is 32GB/s). We also compared these performance numbers with the bandwidth numbers from another memory simulator *DRAMSys* and found that our numbers are comparable to that of *DRAMSys*. We will discuss our changes in detail at the gem5 workshop and plan on presenting more detailed results which validate the newly added HBM model.

D. Refactoring gem5 memory controller

We have refactored gem5's default memory controller to enable its easy extension to different types of memory controllers. The baseline default memory controller *MemCtrl* supports single memory interface. *HBMCtrl* extends the *MemCtrl* and implement the above mentioned features with two memory interfaces. *MemCtrl* can also serve as a base to implement other types of memory controllers.

REFERENCES

[1] J. Lowe-Power, A. M. Ahmad, A. Akram, M. Alian, R. Amslinger, M. Andreozzi, A. Armejach, N. Asmussen, B. Beckmann, S. Bharadwaj *et al.*, "The gem5 simulator: Version 20.0+," *arXiv preprint arXiv:2007.03152*, 2020.