Performance Implications of NoCs on 3D-Stacked Memories: Insights from the Hybrid Memory Cube (HMC)

Ramyad Hadidi, Bahar Asgari, Jeffrey Young, Burhan Ahmad Mudassar, Kartikay Garg, Tushar Krishna, and Hyesoon Kim
Introduction to HMC

Hybrid Memory Cube (HMC) vs High-Bandwidth Memory (HBM)

- HMC: Serial, packet-based interface
- HBM: Wide bus, standard DRAM protocol
  - Found in high-end GPUs and Intel’s Knight’s Landing

Illustration credits: AMD and Micron
Why is HMC Interesting?

- Serialized, high-speed link addresses pin limitation issues with DRAM and HBM
- Abstracted packet interface provides opportunities for novel memories and addressing opportunities
  - Can be used with DRAM, PCM, STT-RAM, NVM, etc.
- Memory controller sits on top of a “routing” layer
  - Allows for more interesting connections between processors and memory elements
  - This study addresses the impacts of the network on chip (NOC) for architects/application developers

Illustration credits: Micron
This Study’s Contributions

We examine the NoC of the HMC using an FPGA-based prototype to answer the following:

1) How does the NoC behave under low- and high-load conditions?
2) Can we relate QoS concepts to 3D stacked memories?
3) How does the NoC affect latency within the HMC?
4) What potential bottlenecks are there and how can we avoid them?
Hybrid Memory Cube (HMC)

HMC 1.1 (Gen2): 4GB size
Hybrid Memory Cube (HMC)

HMC 1.1 (Gen2): 4GB size

16 Banks/Vault
Total Number of Banks = 256
Size of Each Bank = 16 MB

- Logic Layer
- Vault Controller
- DRAM Layer
HMC Memory Addressing

Closed-page policy

Page Size = 256 B

Low-order-interleaving address mapping policy

34-bit address field:

Block Address	Vault ID in a Quadrant
Bank ID	Quadrant ID
Ignored

4K OS Page
HMC Communication I

Follows a serialized **packet-switched** protocol
Partitioned into 16-byte *flit*
Each transfer incurs 1 flit of overhead

<table>
<thead>
<tr>
<th>Type</th>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read</td>
<td>Write</td>
</tr>
<tr>
<td>Data Size</td>
<td>Empty</td>
<td>1~8 Flits</td>
</tr>
<tr>
<td>Overhead</td>
<td>1 Flit</td>
<td>1 Flit</td>
</tr>
<tr>
<td>Total Size</td>
<td>1 Flit</td>
<td>2~9 Flits</td>
</tr>
</tbody>
</table>
HMC Communication II

(a) Flow Control

(b) Request/Response
Micron’s AC-510 module contains a Xilinx Kintex FPGA and HMC 1.1 4 GB part
-2 half-width links for a total of 60 GB/s of bandwidth
-Host SW communicates over PCIe to FPGA-based queues
Methodology (GUPS)
Methodology (multi-port stream)
Experiments

[1] High-Contention Latency Analysis *(GUPS design)*

[2] Low-Contention Latency Analysis *(Multi-port stream)*

[3] Quality of Service Analysis *(Multi-port)*


[5] Requested and Response Bandwidth Analysis *(GUPS)*
[1] Read-only Latency vs. Bandwidth

![Graph showing read-only latency vs. bandwidth for different sizes of data (16B, 32B, 64B, 128B) and varying number of vaults and banks.](image)

- **Latency (μs)** vs. **Bandwidth (GB/s)**
  - **Size 16B**
  - **Size 32B**
  - **Size 64B**
  - **Size 128B**

- Key labels:
  - 1 bank
  - 2 banks
  - 4 banks
  - 8 banks
  - 1 vault
  - 2 vaults
  - 4 vaults
  - 8 vaults
  - 16 vaults

---

ISPASS 2018
Average Latency vs Requests

Latency (μs) vs Number of Read Requests

- ▲ 16B
- ▲ 32B
- ○ 64B
- ○ 128B

Graph shows the increase in latency (μs) as the number of read requests increases, with different line styles and markers representing various data point sizes.
Average Latency vs Requests II

Latency (μs) vs Number of Read Requests

- 16B
- 32B
- 64B
- 128B

Linear Increment
[3] QoS for 4 Vaults

![Graph showing Maximum Latency (μs) vs Vault Number for different data sizes: 16B, 32B, 64B, 128B.](image-url)
[4] Latency vs. Request Size

Latency (ns)

Vault Number

16B

32B

64B

128B
[4] Latency vs. Request Size

Latency vs. Request Size

16B

Latency vs. Request Size

32B

Latency vs. Request Size

64B

Latency vs. Request Size

128B

Average Latency ($\mu s$) vs. Request Size

- 16B
- 32B
- 64B
- 128B

Average
Standard Deviation

Latency Standard Deviation (\(\sigma\)) (ns)

0 40 80 120 160 200
[5] GUPS – Bandwidth vs. Active Ports

(a) 16B
(b) 32B
GUPS – Bandwidth vs. Active Ports II

---

(c) 64B

(d) 128B
[6] GUPS – Outstanding Requests

The diagram shows the number of outstanding requests for different request sizes and the number of banks. Three request sizes are presented: 16, 32, 64, and 128 bytes, and an average is also shown. Two scenarios are considered: 2 banks and 4 banks. The number of outstanding requests is depicted on the y-axis, and the request size (in bytes) is on the x-axis.
Takeaways

Large and small requests allow tuning for bandwidth- or latency-optimized applications better than DRAM

- Vault- and bank-level parallelism are key to achieving higher BW

Vault latencies are more correlated with access patterns and traffic than with physical vault location

Queuing delays will continue to be a concern with NOCs in the HMC

- Address via host-side queuing/scheduling or by distributing accesses across vaults (data structures or compiler passes)

The HMC’s NoC complicates QoS due to variability

- However, trade-offs in packet size and ”private” vaults can improve QoS
Questions?

Thanks to Micron for helping to support our HMC testbed!