High-Bandwidth, Energy-efficient DRAM Architectures for GPU systems

Mike O’Connor - November 2, 2015
GPUs Demand High DRAM Bandwidth

<table>
<thead>
<tr>
<th>GDDR3</th>
<th>GDDR5</th>
<th>HBM</th>
<th>HBM2</th>
<th>Future DRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2000</td>
<td>4000</td>
<td>8000</td>
<td>16000</td>
</tr>
</tbody>
</table>

Gigabytes per Second
GPUs Demand High DRAM Bandwidth

2008: NVIDIA Tesla GT200
512-bits @ 2.2 Gbps
GPUs Demand High DRAM Bandwidth

2010: NVIDIA Fermi
GF100
384-bits @ 3.7 Gbps
GPUs Demand High DRAM Bandwidth

2013: NVIDIA Kepler
GK110
384-bits @ 6 Gbps
GPUs Demand High DRAM Bandwidth

2013: NVIDIA Maxwell GM200
384-bits @ 7 Gbps
GPUs Demand High DRAM Bandwidth

2013: AMD Fury X
“Fiji”
4096-bits @ 1 Gbps
GPUs Demand High DRAM Bandwidth

Future GPU with 4-stack HBM2
4096-bits @ 2 Gbps
GPUs Demand High DRAM Bandwidth

![Graph showing DRAM bandwidth over time with labels for GDDR3, GDDR5, HBM, HBM2, and Future DRAM. The graph indicates an increasing trend towards higher bandwidth with the 4 TB/sec Exascale Node Target highlighted.](image-url)
GPUs Demand High DRAM Bandwidth

CPUs, Not so Much

Typical PC CPU
2 Channel DDR3-1600
51.2 GB/sec
GPUs Demand High DRAM Bandwidth

CPUs, Not so Much
Why Do GPUs Demand so Much Bandwidth?

*Lots* of compute

- 24 Streaming Multiprocessors
  - each w/ 128 execution units

*Lots* of threads

- 64 warps of 32 threads
  - per SM

→ 49,152 threads executing simultaneously on 3072 execution units

NVIDIA Maxwell GM200
Why do GPUs Demand so Much Bandwidth?

Different Memory Hierarchies

NVIDIA Maxwell GM200

<table>
<thead>
<tr>
<th>Level</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 MB Register File</td>
<td>6 MB</td>
</tr>
<tr>
<td>2.3 MB L1/Scratchpad</td>
<td>2.3 MB</td>
</tr>
<tr>
<td>3 MB L2</td>
<td>3 MB</td>
</tr>
<tr>
<td>25.25K PRF</td>
<td>25.25K</td>
</tr>
<tr>
<td>128K L1</td>
<td>128K</td>
</tr>
<tr>
<td>512K L2</td>
<td>512K</td>
</tr>
<tr>
<td>8 MB L3</td>
<td>8 MB</td>
</tr>
<tr>
<td>Intel Core i7-4790</td>
<td></td>
</tr>
</tbody>
</table>
Why do GPUs Demand so Much Bandwidth?

Radically Different Memory Hierarchies per Thread

NVIDIA Maxwell GM200
49,152 Threads

Intel Core i7-4790
8 Threads
Why do GPUs Demand so Much Bandwidth?

Radically Different Memory Hierarchies per Thread

On-Chip Storage:

240 bytes per thread vs. 1,133,856 bytes per thread

Intel Core i7-4790
8 Threads

NVIDIA Maxwell
49,152 Threads

48B

3.28K Register File (PRF)

1 MB L3

16KB L1

64KB L2

6 MB Register File
(Holds state for many threads)

~2 MB L1 and Scratchpad

1 MB L3

64K L2

154 KB L1

624K L1
Whole Different Ballgame

vs. typical CPU

25x More Bandwidth
Less sensitive to latency
LESS regular access patterns
More sensitive to tRRD/tFAW
I thought GPUs had regular/streaming accesses?

GPU warp = 32 Threads executed in SIMD manner

LD r1, [r2+tid.x]

Nice, Coalesced load of 32 x 4B = 128B
I thought GPUs had regular/streaming accesses?

But a lot is happening in parallel...
I thought GPUs had regular/streaming accesses? But a lot is happening in parallel...

24 SM’s...
64 warps each...
Highly interleaved execution!

HA HA HA! I will bring Chaos to your puny DRAM system!
CPU

Fewer threads, lower chance of bank conflicts

Typical CPU: 8 Threads

→ 4 banks per thread

2 ranks/channel
8 banks per rank
GPU

Many different threads competing

NVIDIA Titan X: 1536 Warps

→ 0.06 banks per warp

6 channels
16 banks per channel
What does this mean for Energy?

Low Accesses per Activate

→ Average ~160 Bytes accessed per activate

(5 x 32B)
What does this mean for Energy?

Low Accesses per Activate

→ High Accesses per Activate workloads are typically simpler functions like large data copies
What does this mean for Energy?

Low Accesses per Activate

→ Many of the 1 Access/Activate workloads are doing graph traversals of large graphs
What does this mean for Energy?

Want to reduce row overfetch

DRAM device typically has 1-2KB row
- we only need 160B on average

Wastes ~84-92% of the energy on activate/precharge of a DRAM bank

Multiple devices in parallel (like a DIMM) make this even worse
What does this mean for Performance?

Activate rate is a key

High-bandwidth I/O is nice, but...

Key aspect of performance is rate of activates
  - low tRRD and tFAW
  - high number of channels

Typical 2-channel, 2-rank DDR4 w/ $t_{RRD_{\text{eff}}} = 5.25\text{ns}$: $762 \text{ M ACT/sec}$

HBM2 4-stack, 64-channel (w/ pseudochannels) w/ $t_{RRD_{\text{eff}}} = 4\text{ns}$: $16 \text{ G ACT/sec}$
Small DRAM Atoms

DRAM “atom” is smallest indivisible access
Basically a function of bus-width and burst-length

GPUs extensively use compression of graphics surfaces
Efficiency of compression a function of minimum access size
Efficient partial coverage
What to GPUs need from DRAM?

High Bandwidth (w/ low access / activate)

Energy-efficient (largely because of high bandwidth)

Small minimum burst sizes (e.g. 16-32B)

Not necessarily concerned with:

- Low-latency
- Extremely large capacities
- Lowest possible cost
One Approach: High-Speed Signaling

Start with a commodity DRAM core
Since GPUs don’t need huge capacities, drop multi-rank support
Without multi-drop busses and sockets, push I/O data rates
Beef-up the DRAM core to keep up (reduce tRRD/tFAW if possible)

Basic approach behind GDDR DRAMs
GDDR5

GDDR5 signals at up to 8 Gbps
GDDR5X soon going to 10-12 Gbps
Lots of board challenges
I/O energy efficiency not so great
Challenges with High-speed Signaling

Limits on data rates with inexpensive board & package

High-data rates place demands on the DRAM core
- Cycle the DRAM core arrays faster
  And/Or
- Sub-partition pieces of the DRAM array
  And/Or
- Fetch more data from the array each time
Another Approach: In-package Integration

3D Stacking technologies enable many more I/Os

What if instead of faster, we go wider...
What is High-Bandwidth Memory (HBM)?

Memory standard designed for needs of future GPU and HPC systems:

- Exploit very large number of signals available with die-stacking technologies for very high memory bandwidth
- Reduce I/O energy costs
- Enable higher fraction of peak bandwidth to be exploited by sophisticated memory controllers
- Enable ECC/Resilience Features


Initial work started in 2010
What is High-Bandwidth Memory (HBM)?

Enables systems with extremely high bandwidth requirements like future high-performance GPUs
HBM Overview

Each HBM stack provides 8 independent memory channels

These are completely independent memory interfaces

Independent clocks & timing
Independent commands
Independent memory arrays

In short, nothing one channel does affects another channel
HBM Overview - Bandwidth

Each channel provides a 128-bit data interface
  Data rate of 1 Gbps per signal (500 MHz DDR)
  HBM2 bumps this to 2 Gbps per signal (1 GHz DDR)
  16-32 GB/sec of bandwidth per channel

8 Channels per stack
  128-256 GB/sec of bandwidth per stack

For comparison:
  Highest-end GDDR5-based today (NVIDIA GeForce GTX 980Ti)
    384b wide GDDR5 (12 x32 devices) @ 7 Gbps = 336 GB/s
  AMD Fury X with 4 stacks of HBM
    Four stacks of HBM @ 1 Gbps = 512 GB/s
  Future possible GPU with 4 stacks of HBM2
    Four stacks of HBM2 @ 2 Gbps = 1 TB/s
HBM Overview - Bandwidth

Each channel provides a 128-bit data interface
Data rate of 1 Gbps per signal (500 MHz DDR)
HBM2 bumps this to 2 Gbps per signal
16-32 GB/sec of bandwidth per channel
8 Channels per stack
128-256 GB/sec of bandwidth per stack

For comparison:
Highest-end GDDR5-based today (NVIDIA GeForce GTX 980Ti)
384b wide GDDR5 (12 x32 devices) @ 7 Gbps = 336 GB/s
AMD Fury X with 4 stacks of HBM
Four stacks of HBM @ 1 Gbps = 512 GB/s
Future possible GPU with 4 stacks of HBM2
Four stacks of HBM2 @ 2 Gbps = 1 TB/s

At lower overall DRAM system power -
~6 pJ/bit vs.
~18 pJ/bit for GDDR5
HBM Overview - Capacity

Per-channel capacities supported from 1-32 Gbit
Stack capacity of 1 to 32GBytes

Nearer-term, at lower-end of range
   HBM: 4 high stack of 2Gb dies = 1GBytes/stack
   HBM2: 4 high stack of 8Gb dies = 4GBytes/stack

8 or 16 banks per channel
   16 banks when > 4Gbit per channel (> 4GBytes/stack)

Not including optional additional ECC bits
   A stack providing ECC storage may have 12.5% more bits
HBM Channel Overview

Each channel is similar to a standard DDR interface

Data interface is bi-directional
  Still requires delay to “turn the bus around” between RD and WR
  Burst-length of 2 (32B per access)

Requires traditional command sequences
  Activates required to open rows before read/write
  Precharges required before another activate
  Traditional dram timings still exist (tRC, tRRD, tRP, tFAW, etc.) - but are entirely per-channel
### HBM Channel Summary

<table>
<thead>
<tr>
<th>Function</th>
<th># of µBumps</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>128</td>
<td>DDR, bi-directional</td>
</tr>
<tr>
<td>Column Command/Addr.</td>
<td>8</td>
<td>DDR</td>
</tr>
<tr>
<td>Row Command/Addr.</td>
<td>6</td>
<td>DDR</td>
</tr>
<tr>
<td>Data Bus Inversion</td>
<td>16</td>
<td>1 for every 8 Data bits, bi-directional</td>
</tr>
<tr>
<td>Data Mask/Check Bits</td>
<td>16</td>
<td>1 for every 8 Data bits, bi-directional</td>
</tr>
<tr>
<td>Strobes</td>
<td>16</td>
<td>Differential RD &amp; WR strobes for every 32 Data bits</td>
</tr>
<tr>
<td>Clock</td>
<td>2</td>
<td>Differential Clock</td>
</tr>
<tr>
<td>Clock Enable</td>
<td>1</td>
<td>Enable low-power mode</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>193</strong></td>
<td></td>
</tr>
</tbody>
</table>
New: Split Command Interfaces

2 semi-independent command interfaces per channel
  “Column Commands” - Read / Write
  “Row Commands” - ACT / PRE / etc.

Key reasons to provide separate row command i/f:
  100% column command bandwidth to saturate the data bus w/ BL=2
  Simplifies memory controller
  Better performance (issue ACT earlier or not delay RD/WR)

Still need to enforce usual ACT→RD/WR→PRE timings
New: Single-Bank Refresh

Current DRAMs require refresh operations
  Refresh commands require all banks to be closed
  ~ 1 refresh command every few µsec
  Can consume 5-10% of potential bandwidth
  Increasing overheads with larger devices

Sophisticated DRAM controllers work hard to overlap ACT/PRE in one bank with traffic to other banks
  Can manage the refresh similarly
  Added “Refresh Single Bank” command
    Like an ACT, but w/ internal per-bank row counter
    Can be issued to any banks in any order
    Memory controller responsible for ensuring all banks get enough refreshes each refresh period
New: RAS Support

HBM standard supports ECC

Optional: Not all stacks required to support it

ECC and non-ECC stacks use same interface

Key insight:
Per-byte data mask signals and ECC not simultaneously useful

Data Mask Signals can carry ECC data
- makes them bi-directional on HBM stacks that support ECC
Other HBM Features

HBM supports Temperature Compensated Self Refresh

Temperature dependent refresh rates with several temperature ranges (e.g. cool/standby, normal, extended, emergency)

Temperature sensor can be read by memory controller to adjust its refresh rates as well

DBIac Data Bus Inversion coding

Reduce number of simultaneously switching signals

No more than 4 of 9 (DQ[0..7], DBI) signals switch

DBI computation maintained across consecutive commands
HBM2 - The next step

Evolution of HBM

Doubles bandwidth of I/O channel

- Requires doubling burst-length and DRAM atom

Break up channel into two pseudo-independent half-wide channels

- Pseudochannels add bank-level parallelism
- Prevent DRAM atom size from increasing
- Reduce DRAM row overfetch by cutting effective row in half
High Bandwidth DRAM Energy Trends
Conclusion

GPUs place significant requirements on the DRAM

Ideal GPU DRAM provides

- energy efficient
- high-bandwidth

...to small quantities of data

Stacked memories like HBM2 are good,
but need new innovations to get to Exascale-class nodes