Tutorial T2:

Fundamentals of Memory Subsystem Design for HPC and AI

Kyu-Hyoun (KH) Kim
IBM
kimk@us.ibm.com

Live Q&A Session:
Feb. 13, 2020, 7:20-7:40am, PST
Self Introduction

- PhD degree from KAIST, Korea in 1997
- Was with Samsung Electronics 1998-2006
  - DRAM design, CIS design
- Have been with IBM since 2006
  - Memory subsystem design for IBM HPC systems (BlueGene/Q and CORAL), P/Z servers, and AI machines
  - High speed I/O circuit design for IBM P and Z processors
  - Memory standardization (vice-chair of JEDEC DDR4 and DDR5 TGs)
  - AI H/W development
- Focusing on exploratory AI H/W and algorithms
Outline

- Overview
  - Memory metrics
- Memory technologies
  - DRAM and NVM
- Memory bandwidth
- Memory capacity
  - Interface and Modules
  - Advanced packaging
- RAS (Reliability, Accessibility, Serviceability)
- Memory controller
- System Design Examples
  - HPC and AI
- Summary
Overview
Goal of this talk

- Understand the memory subsystem
  - Building blocks
  - Architecture
  - Metrics
  - Design variables
  - Constraints
  - (Feeling of) how to design a system
Metrics

- Bandwidth
- Latency
- Capacity
- Power
- Cost
- Reliability
(Maximum) Bandwidth

- Amount of data that memory can read or write per second (e.g. 100GB/s)
  - Could be different for read : write ratio
- Peak BW vs. sustained BW
  - Effective BW, BW utilization, bus utilization
- BW per module, BW per processor, BW per socket, BW per system
Latency

- Read latency: time delay from read request to data
  - Write latency: delay from write request to write data, or from write data to internal write completion
- Point of measurement: core (load to use) vs. mem. controller (command to data)
- Idle (unloaded) latency [no other traffic] vs. loaded latency [heavy traffic]
- Random access latency vs. sequential access latency (for memories having pages [page hit] – e.g. DRAM)
- May not be critical for parallel workloads (HPC, AI [training], etc.)
- Critical for sequential workload (pointer chasing, Graph analytics, latency sensitive AI inference, etc.)
Memory

- Memory chips use Gb
- Memory modules use GB
- 1GB = $2^{30}$ Bytes (or called 1GiB)

Number of processor cores and threads keep increasing

- Memory capacity per processor thread is roughly constant (e.g. 32-64GB/thread)
Power

- Why is power consumption important?
  - Server: power = cooling
  - Mobile: power = battery

- Memory is not the highest power consumer (per component), but has lower max. temperature limit
  - e.g. 85°C in DRAM
    - 95°C with performance degradation (and more refresh power)
  - Cooled by air flow (air flow speed matters)
    - Water cooling is more effective but very expensive
  - Higher memory power
    - Faster air flow (bigger fan, higher fan noise), OR
    - Reduced Performance (throttle), OR
    - Higher cooling cost (room A/C, water cool)
Cost

- **Cost / bit [$/GB] (as of Oct. 2020)**
  - DRAM ~ $4.2/GB
  - NAND ~ $0.11/GB

- **Assume**
  - Intel i9 (8 cores, 2 threads / core)
  - 32GB / thread (→ 512GB)
  - CPU cost = $380
  - DRAM cost = $2,150

- **Memory may be the major cost contributor**
  - Lower cost memories are available (e.g. NAND), but with much lower performance (BW & latency)

- **Memory cost**
  - 5 – 20% in client systems
  - 20 – 50% in server systems
Design variables - Memory technology

- **DRAM**
  - DDRx (DDR4, DDR5, [DDR6])
  - HBMx (HBM2/2E, HBM3, [HBM4])
  - LPDDRx (LPDDR5/5X, [LPDDR6])
  - GDDRx (GDDR6/6X, [GDDR7])

- **Non-Volatile Memory (NVM)**
  - NAND
  - PCM (including 3DXP)
  - RRAM
  - MRAM
  - Etc.

Latency, BW, reliability

Capacity, cost

BW, capacity, and cost tradeoff

Capacity, latency
Design variables - Architecture

- Homogeneous
  Direct attach

Homogeneous:
- Raw memory interface
- Latency
- Capacity
- BW
- Cost
- Power

Attach through memory buffer:
- High speed interface
- BW
- Cost
- Power

Diagram:
- Host
- Mem
- Hub
Design variables – Architecture (cont’d)

- Heterogeneous
  - Tiered
    - Fast (near)
    - Slow (far)

- Hierarchical
  - S/W overhead
  - BW overhead
  - H/W overhead (tag)

Diagram:

- Host
  - DRAM
  - NVM

- Host
  - DRAM
  - NVM

- Host
  - cache
  - DRAM
  - NVM
Design variables – Form factor

- **DRAM**
  - Processor and memory in a same package
  - Solder down memory to the mother board
  - Module (DIMM: Dual In-line Memory Module)
  - Riser card

- **NVM**
  - M.2 card
  - PCIe card (HHHL/FHHL)
  - 2.5-inch
Overview

Memory technologies

- Memory bandwidth
- Memory capacity
- RAS (Reliability, Accessibility, Serviceability)
- Memory controller
- System Design Examples
- Summary
Memory Technologies

- Memory media
  - Conventional: SRAM, DRAM, NAND
  - Emerging: PRAM(PCM/3D XP), RRAM(ReRAM), MRAM(STT-MRAM), etc.

- Standalone vs. embedded

- Standard vs. custom
  - SRAM, DRAM and NAND are standardized in JEDEC
  - JEDEC: JEDEC Solid State Technology Association (www.jedec.org)
    - Memory components, modules, interface, form factor, reliability etc.
  - Why standard memories?
DRAM technologies

- Standard DRAM families
  - Main memory (commodity) DRAMs
    - DDR/DDR2/DDR3/DDR4/DDR5
  - Mobile DRAMs
    - LPDDR/LPDDR1/LPDDR2/LPDDR3/LPDDR4/LPDDR5
    - WIO/WIO2
  - Graphics DRAMs
    - GDDR/GDDR2/GDDR3/GDDR4/GDDR5/GDDR6
  - HPC DRAMs
    - HBM/HBM2/HBM2E/HBM3
Double Data Rate (x\textsuperscript{th} gen.) Synchronous DRAM

One of the two most popular DRAMs
- Server, desktop, notebook, etc.

DDR4 being currently widely used
- DDR4 is probably the one in your notebook
- DDR5 standardization completed and will appear in the market from YE 2021

DDRx yyyy (component) -> PCx-zzzz (module)
- zzzz = 8 x yyyy
- e.g. DDR4 3200 -> PC4-25600
  - Component: DDR4 3200Mbps (per pin)
  - Module (64 data pins): 3200Mbps x 8B = 25600MB/s
- 1\textsuperscript{st} gen: DDR 200 -> PC-1600
LPDDRx

- Optimized for mobile application
  - Battery life -> low refresh power
    - When your phone is in standby, the memory is just refreshing
    - Low leakage cell (but longer latency), low power circuit techniques for periphery
  - Smaller form factor -> lower capacity, decent BW -> BW/capacity ratio is higher than DDRx
  - Interface is almost same to DDRx with minor change
    - Command & address packet designed for lower pin count
  - Supply voltage is lower than DDRx by one generation

<table>
<thead>
<tr>
<th></th>
<th>Gen. 1</th>
<th>Gen. 2</th>
<th>Gen. 3</th>
<th>Gen. 4</th>
<th>Gen. 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDRx</td>
<td>2.5</td>
<td>1.8</td>
<td>1.5 (1.35)</td>
<td>1.2</td>
<td>1.1</td>
</tr>
<tr>
<td>LPDDRx</td>
<td>1.8</td>
<td>1.2 &amp; 1.8</td>
<td>1.2</td>
<td>1.1</td>
<td>1.05</td>
</tr>
</tbody>
</table>
GDDRx

- Optimized for graphics card
  - Per pin data rate is ~5x higher speed than DDRx
  - Supply voltage is one generation lagging (higher) compared to DDRx
  - Larger die size than DDRx
  - No stacking (limited capacity)
## Comparison (DRAM)

<table>
<thead>
<tr>
<th></th>
<th>DDR4</th>
<th>LPDDR4x</th>
<th>GDDR6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Per-pin speed</strong></td>
<td>3.2Gbps</td>
<td>4.26Gbps</td>
<td>14Gbps</td>
</tr>
<tr>
<td><strong>VDD</strong></td>
<td>1.2V</td>
<td>1.1V</td>
<td>1.35V</td>
</tr>
<tr>
<td><strong>Cost / bit</strong></td>
<td>1x</td>
<td>~1.2x</td>
<td>~2x</td>
</tr>
<tr>
<td><strong>Bus width per PKG</strong></td>
<td>x4/x8/x16</td>
<td>x32/x64</td>
<td>x16/x32</td>
</tr>
<tr>
<td><strong>BW per PKG</strong></td>
<td>1.6-6.4GB/s</td>
<td>17-34GB/s</td>
<td>28-56GB/s</td>
</tr>
<tr>
<td><strong>Max. capacity per PKG</strong></td>
<td>8GB</td>
<td>8GB</td>
<td>2GB</td>
</tr>
<tr>
<td><strong>BW per x64</strong></td>
<td>25.6GB/s 1)</td>
<td>68GB/s 2)</td>
<td>112GB/s 3)</td>
</tr>
<tr>
<td><strong>Capacity per x64</strong></td>
<td>128GB 1)</td>
<td>16GB 2)</td>
<td>8GB 3)</td>
</tr>
</tbody>
</table>

1\) Sixteen x4 chips  
2\) Two x32 chips  
3\) Four x16 chips
Emerging NVM

<table>
<thead>
<tr>
<th>Feature</th>
<th>DRAM</th>
<th>STT-MRAM</th>
<th>3DXP</th>
<th>Other PCM</th>
<th>ReRAM</th>
<th>LL-NAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>28ns</td>
<td>&lt; 100ns</td>
<td>&lt; 500ns</td>
<td>~ 3DXP</td>
<td>&lt; 1us</td>
<td>&lt; 5us</td>
</tr>
<tr>
<td>Density</td>
<td>16Gb</td>
<td>1Gb</td>
<td>&gt; 128Gb</td>
<td>&gt; 128Gb</td>
<td>&gt; 128Gb</td>
<td>&gt; 128Gb</td>
</tr>
<tr>
<td>Endurance</td>
<td>&gt; 1E15</td>
<td>1E9</td>
<td>&gt; NAND</td>
<td>~ 3DXP</td>
<td>&gt; NAND</td>
<td>&gt;&gt; TLC NAND</td>
</tr>
<tr>
<td>RBER</td>
<td>&lt; 1E-20</td>
<td>&gt; DRAM</td>
<td>&gt; DRAM</td>
<td>&gt; DRAM</td>
<td>&gt; DRAM</td>
<td>&gt; DRAM</td>
</tr>
<tr>
<td>Availability</td>
<td>Now</td>
<td>Now</td>
<td>Now</td>
<td>2 yrs ?</td>
<td>3 yrs ?</td>
<td>Now</td>
</tr>
<tr>
<td>Cost/Gb</td>
<td>1x</td>
<td>&gt;&gt; 1x</td>
<td>&lt; 0.5x</td>
<td>&lt; 0.5x</td>
<td>&lt; 0.5x</td>
<td>&lt;&lt; 0.5x</td>
</tr>
</tbody>
</table>
Comparison (DRAM & non-DRAM)

<table>
<thead>
<tr>
<th>Capacity per Processor (CPU, GPU or Accelerator)</th>
<th>BW per processor (CPU, GPU or Accelerator ASIC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND</td>
<td>100GB/s</td>
</tr>
<tr>
<td>PCM</td>
<td>100GB/s</td>
</tr>
<tr>
<td>DDR4</td>
<td>1TB/s</td>
</tr>
<tr>
<td>LPDDR4x</td>
<td>1TB/s</td>
</tr>
<tr>
<td>GDDR6</td>
<td>1TB/s</td>
</tr>
<tr>
<td>HBM2E</td>
<td>1TB/s</td>
</tr>
</tbody>
</table>
- Overview
- Memory technologies
- **Memory bandwidth**
- Memory capacity
- RAS (Reliability, Accessibility, Serviceability)
- Memory controller
- System Design Examples
- Summary
Memory bandwidth

- Two bottlenecks
  - I/O bandwidth
    - How quickly data can be transferred (transmitted and received) between chips (memory chip and processor chip)
    - This limits the peak bandwidth (because BW cannot be greater than the I/O BW)
  - Core bandwidth
    - How quickly the core can deliver data from the cells to the I/O
    - This determines the sustained bandwidth (because the best-case core bandwidth cannot be higher than the peak I/O BW)
Core bandwidth

- Single cell BW (random access)
  - Cycle time $\approx 50\text{ns} (=t_{RC})$ -> BW = 20Mbps

- Read 64 cells simultaneously
  - BW = 64x20Mbps = 1280Mbps = 160MB/s

- Interleave 16 banks (and read 64 cells per bank)
  - BW = 16x160MB/s = 2.56GB/s
I/O bandwidth

- Target I/O BW = 2.56GB/s
- Assuming 1-pin (x1 DRAM)
  - Per-pin speed should be 20.48Gbps
    - 1UI = 48.83ps, 64UI = 3.125ns
    - One data chunk (64bit) is transmitted as 64 burst (BL=64)

- Assuming N-pins

<table>
<thead>
<tr>
<th>Bus width</th>
<th>Data-Rate</th>
<th>1UI</th>
<th>BL</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>20.48Gbps</td>
<td>48.8ps</td>
<td>64</td>
</tr>
<tr>
<td>x2</td>
<td>10.24Gbps</td>
<td>97.7ps</td>
<td>32</td>
</tr>
<tr>
<td>x4</td>
<td>5.12Gbps</td>
<td>195.5ps</td>
<td>16</td>
</tr>
<tr>
<td>x8</td>
<td>2.56Gbps</td>
<td>391ps</td>
<td>8</td>
</tr>
<tr>
<td>x16</td>
<td>1.28Gbps</td>
<td>781ps</td>
<td>4</td>
</tr>
<tr>
<td>x32</td>
<td>640Mbps</td>
<td>1.563ns</td>
<td>2</td>
</tr>
<tr>
<td>x64</td>
<td>320Mbps</td>
<td>3.125ns</td>
<td>1</td>
</tr>
</tbody>
</table>
DRAM model so far

Bank cycle time = 50ns

Bank 0

Bank 1

Bank 15

Control

Command & Address

64bits every 50ns

64bits every 3.125ns

Data path cycle time = 3.125ns

I/O (Serializer)

8bits every 390ps

UI = 390ps

Data (BW = 2.56GB/s)

But this is too low!
Increasing BW – 1) increase on-chip parallelism

Bank cycle time = 50ns

64bits every 50ns

Bank cycle time = 50ns

Data path cycle time = 1.56ns

8bits every 195ps x BL8
or 16bits every 390ps x BL4

Data (BW = 5.12GB/s)
Increasing BW – 2) increase fetch size per chip

- Bank cycle time = 50ns
- Data path cycle time = 3.125ns
- BW = 5.12 GB/s

128 bits every 50ns
128 bits every 3.125ns
16 bits every 390ps x BL8
Increasing BW – 3) use many chips in parallel

N x 8 bits every 390ps -> N x 2.56 GB/s

UI=390ps (2.56Gbps per pin)
Using many chips in parallel

<table>
<thead>
<tr>
<th>Chips</th>
<th>BW</th>
<th>BL</th>
<th>Total bus width</th>
<th>Data size per read</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.56 GB/s</td>
<td>8</td>
<td>1B</td>
<td>8B</td>
</tr>
<tr>
<td>2</td>
<td>5.12 GB/s</td>
<td>8</td>
<td>2B</td>
<td>16B</td>
</tr>
<tr>
<td>4</td>
<td>10.24 GB/s</td>
<td>8</td>
<td>4B</td>
<td>32B</td>
</tr>
<tr>
<td>8</td>
<td>20.48 GB/s</td>
<td>8</td>
<td>8B</td>
<td>64B</td>
</tr>
<tr>
<td>16</td>
<td>40.96 GB/s</td>
<td>8</td>
<td>16B</td>
<td>128B</td>
</tr>
<tr>
<td>32</td>
<td>81.92 GB/s</td>
<td>8</td>
<td>32B</td>
<td>256B</td>
</tr>
<tr>
<td>64</td>
<td>163.8 GB/s</td>
<td>8</td>
<td>64B</td>
<td>512B</td>
</tr>
<tr>
<td>512</td>
<td>1.31 TB/s</td>
<td>8</td>
<td>512B</td>
<td>4KB</td>
</tr>
</tbody>
</table>

- Data size becomes greater than what processor needs (cache line size)
  - Cache line size: 32B, 64B (Intel), 128B (IBM)
How can we achieve > 100GB/s then?

### Chips BW BL Bus width Data size
<table>
<thead>
<tr>
<th>Chips</th>
<th>BW</th>
<th>BL</th>
<th>Bus width</th>
<th>Data size</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>20.48GB/s</td>
<td>8</td>
<td>8B</td>
<td>64B</td>
</tr>
</tbody>
</table>

- **Group 0**: CA0 20.5GB/s
- **Group 1**: CA1 20.5GB/s
- **Group 2**: CA2 20.5GB/s
- **Group 3**: CA3 20.5GB/s
- **Group 4**: CA4 20.5GB/s
- **Group 5**: CA5 20.5GB/s
- **Group 6**: CA6 20.5GB/s
- **Group 7**: CA7 20.5GB/s

Total 164GB/s

Group = Channel
This one exists

<table>
<thead>
<tr>
<th>Chips</th>
<th>BW</th>
<th>BL</th>
<th>Bus width</th>
<th>Data size</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>20.48GB/s</td>
<td>8</td>
<td>8B</td>
<td>64B</td>
</tr>
</tbody>
</table>

(source: amazon.com)

Samsung 16GB DDR4 PC4-21300, 2666MHZ, 288 PIN DIMM, 1.2V, CL 19 desktop ram memory module

4 stars

$58.00

Prime FREE Delivery Mon, Nov 2
Only 4 left in stock - order soon.
Overview

Memory technologies

Memory bandwidth

Memory capacity

- **Interface and Modules**
  - Advanced Packaging Options

- RAS (Reliability, Accessibility, Serviceability)

- Memory controller

- System Design Examples

- Summary
DIMM

- Dual In-line Memory Module

Modules plugged in vertical connectors provide highest memory density (compare to soldering down)

source: amazon.com
Memory Interface

- **Bus architecture**
  - Bidirectional (half duplex) vs. unidirectional (dual simplex)

![Diagram showing bidirectional and unidirectional bus architectures.](attachment:diagram.png)

- **Bidirectional (half duplex) bus architecture**
  - Command/Address (Cmd/Addr) and Write-Data/Read-Data (Wr-data/Rd-data) are transmitted in both directions.
  - Write Bandwidth (Wr BW) + Read Bandwidth (Rd BW) ≤ Bandwidth peak (BW peak)
  - Full Bandwidth for any Read-to-Write (R:W) ratio.

- **Unidirectional (dual simplex) bus architecture**
  - Command/Address (Cmd/Addr) + Write-Data (Wr-data) and Read-Data (Rd-data) are transmitted in opposite directions.
  - Write Bandwidth (Wr BW) ≤ Write Bandwidth peak.
  - Read Bandwidth (Rd BW) ≤ Read Bandwidth peak.
  - Full Bandwidth only when Read-to-Write (R:W) ratio is equal to Read peak to Write peak (Rd_peak : Wr_peak).
## Memory Interface

### Bidirectional vs. unidirectional

<table>
<thead>
<tr>
<th></th>
<th>Bidirectional</th>
<th>Unidirectional</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R:W ratio</strong></td>
<td>Full BW for any R:W ratio</td>
<td>Full BW for a predetermined R:W ratio</td>
</tr>
<tr>
<td><strong>Utilization</strong></td>
<td>Limited by R &lt;-&gt; W turnaround</td>
<td>Close to 100% (for the optimum R:W ratio)</td>
</tr>
<tr>
<td><strong>Bus topology</strong></td>
<td>Multi-drop (multi-drivers) possible</td>
<td>Point to point only</td>
</tr>
<tr>
<td><strong>Capitance</strong></td>
<td>Higher</td>
<td>Lower</td>
</tr>
<tr>
<td><strong>Signaling</strong></td>
<td>Limited due to burst transfer</td>
<td>Continuous transfer -&gt; various I/O techniques can be applied (coding, DFE, etc.)</td>
</tr>
<tr>
<td><strong>Advantage</strong></td>
<td>Higher capacity</td>
<td>Higher speed</td>
</tr>
</tbody>
</table>

- **Bus topology**
  - Multi-drop (multi-drivers) possible (Bidirectional)
  - Point to point only (Unidirectional)

- **R:W ratio**
  - Full BW for any R:W ratio (Bidirectional)
  - Full BW for a predetermined R:W ratio (Unidirectional)

- **Utilization**
  - Limited by R <-> W turnaround (Bidirectional)
  - Close to 100% (for the optimum R:W ratio) (Unidirectional)

- **Capitance**
  - Higher (Bidirectional)
  - Lower (Unidirectional)

- **Signaling**
  - Limited due to burst transfer (Bidirectional)
  - Continuous transfer -> various I/O techniques can be applied (coding, DFE, etc.) (Unidirectional)

- **Advantage**
  - Higher capacity (Bidirectional)
  - Higher speed (Unidirectional)
Memory Interface

- **Signaling**
  - Single-ended vs. Differential (or parallel vs. serial)
    - Differential can achieve much higher speed, but has 2x pin count penalty, consumes more power, and bidirectional bus switching is difficult -> more suitable for high-speed serial link type of interface
    - Single-ended & low/med-speed & wide & parallel vs. differential & high-speed & narrow & serial
  - All standard memory (component and module) interfaces have been single-ended
    - Except two memory modules (Fully Buffered DIMM, Differential DIMM)
    - Some critical signals (clock and strobe) are differential
    - There are custom memories with differential high-speed serial interface
      - Micron HMC, IBM SNDIMM / CDIMM
Memory Interface

- **Clocking**
  - Currently, all standard memory (component and module) interfaces are source synchronous
    - because all those are basically the same Bi-directional strobed DDR interface
    - Usually host memory controller PHY takes all the burden to de-skewing for both directions (write and read)
      - keeps memory component as simple as possible (for low power)
      - There is one host chip, whereas there are many memory chips
      - Transistor performance in memory process is much worse than in ASIC process
      - It is easier for the host to schedule and control all the training
Memory Interface

- Bus topology
  - Point-to-point vs. multi-drop vs. cascading
    - Multi-dropping (dotting) is the easiest way to increase memory capacity independently to bandwidth, only if signal integrity is met
    - Cascading adds complexity and needs 2x memory pins

![Memory Interface Diagram](image-url)
DDR Interface

- Bidirectional Strobed Double Data Rate
  - Bidirectional bus
  - Single-Ended signaling
  - Source synchronous clocking (with data strobe)
    - Strobe instead of clock because of multi-drop

![Diagram showing DDR interface with Host, Mem 0, and Mem 1 connected with WCK, DQ, RCK signals.]
DDR Interface

- **Host** to **Mem** connection with x8 data width.
- **DQ** and **DQS** signals for data and command lines, respectively.
- Example timing diagram:
  - **Write Rank0**
  - **Read Rank0**
  - **Write Rank1**

- **Write** and **Read** operations shown with **preamble** and **postamble** regions.
- **Write** is center aligned, **Read** is edge aligned (MC responsible for 90° shift).

---

KH Kim

ISSCC 2021 Tutorial
DIMM architecture

8 x 16Gb = 16GB
Doubling capacity

32GB

Rank0

Rank1

CK, CMD, ADDR

DQ x8

DQS

DQ x8

DQ x8

DQ x8

DQ x8

DQ x8

DQ x8
Doubling capacity again

64GB UDIMM

Rank0

Rank1

CK, CMD, ADDR

DQ x4
DQS
Buffering CK,C/A

RCD: register clock drive

64GB RDIMM

KH Kim

ISSCC 2021 Tutorial

50 of 105
Doubling capacity further: Dual Die PKG (DDP)

128GB RDimm
Buffering DQ: Load Reduced DIMM (LRDIMM)

128GB LRDIMM
Stacking technologies

QDP  Quad Die PKG

DDP  Dual Die PKG

4-high 3-dimensional Stack

TSV  Through Silicon Via

Single Die PKG

SDP (Mono)

4H 3DS

U. Kang, ISSCC 2009
3DS DRAM (Single-load stack)

RDIMM with 4H 3DS

4H 3DS

256GB RDIMM
RDIMM with 4H 3DS

128GB

9 columns

256GB

9 columns

Why 9 not 8?
Overview
Memory technologies
Memory bandwidth
Memory capacity
  - Interface and Modules
  - Advanced Packaging Options
RAS (Reliability, Accessibility, Serviceability)
Memory controller
System Design Examples
Summary
System using RDIMMs

8 channels per socket, 1DIMM/channel

Capacity per socket: Max. 256GB x 8 = 2TB

BW per socket: 8 x 3.2Gbps x 8B = 204.8GB/s (peak)
Processor pin count

- Processor socket pin counts
  - IBM POWER9 SO: 2601
  - Intel Cascade Lake: 2066
  - AMD EPYC: 4094

- Processor pins for 1 RDIMM channel ~ 200 (inc. GND)

<table>
<thead>
<tr>
<th>Channels</th>
<th>Bandwidth [GB/s]</th>
<th>Min. capacity</th>
<th>Max. Capacity</th>
<th>Pins for memory</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>102</td>
<td>64GB</td>
<td>1TB</td>
<td>800</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>154</td>
<td>96GB</td>
<td>1.5TB</td>
<td>1200</td>
<td>Cascade Lake</td>
</tr>
<tr>
<td>8</td>
<td>205</td>
<td>128GB</td>
<td>2TB</td>
<td>1600</td>
<td>POWER9, EPYC</td>
</tr>
<tr>
<td>12</td>
<td>307</td>
<td>192GB</td>
<td>3TB</td>
<td>2400</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>410</td>
<td>256GB</td>
<td>4TB</td>
<td>3200</td>
<td></td>
</tr>
</tbody>
</table>
Advanced PKG option-1: Differential DIMM

- DDR5 has two DIMM interfaces
  - Single-ended: RDIMM/LRDIMM
  - Differential: DDIMM
Differential DIMM

- Processor pins for 1 DDIMM channel ~ 50 (inc. GND)
- BW per processor pin
  - RDIMM: 25.6GB/s per 200 pins
  - DDIMM: 64GB/s per 50 pins

<table>
<thead>
<tr>
<th>Channels</th>
<th>BW</th>
<th>Min. capacity</th>
<th>Max. Capacity</th>
<th>Pins for memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>2TB/s</td>
<td>512GB</td>
<td>8TB</td>
<td>1600</td>
</tr>
</tbody>
</table>

Buffer

32GB/s  32GB/s
Discussion initiated to develop next gen. Graphics memory (post GDDR5)

Two ways to achieve ultra high BW
- Ultra high data rate
- Ultra wide bus (chosen)
  - Like the Wide I/O 2 (leverages TSV, unbuffered, post LPDDR4)

Positioned as the mainstream HPC memory due to wide usage of GPGPU in HPC
What’s HBM?

- **DRAM**
  - Same technology, cell and array with DDR4
  - However, there are several uniqueness

- **Major differences from existing DRAM**
  1. Ultra wide: two orders of magnitude larger number of I/Os (1K DQ’s)
  2. Micro bump: one order smaller ball pitch (55um) → needs High Density Interconnect (e.g., silicon interposed)
  3. Large number of TSV’s (HBM ~2K, 3DS ~300)
  4. 8 independent channels
  5. 8H stack is the main volume
  6. Ultra short reach channel (~5mm)
  7. Unterminated net (because it’s ultra short. Otherwise, huge power)

- **It’s a same DRAM, but almost like a different animal**
Deriving HBM (step-1)

- 8x form factor saving
- 8x activation power saving
- 8x larger prefetch (internal data lines) [chip size overhead]
Deriving HBM (step-2)

- 8x parallelism
- 8x bandwidth
- 1x capacity

- 128Gb, x128
- 1 channel per stack
- 8 channel per stack
- 128Gb, x1024
HBM

- HBM2E being used. HBM3 to appear in 2021
  - HBM2E (2.4-3.2Gbps/pin): 307-410GB/s (peak) per PKG
  - HBM3 (4-6.4Gbps/pin): 512-819GB/s
- 8 channels or 16 pseudo channels (doubled in HBM3)
  - Minimum data size = 32B
- 8/16Gb x 4/8/12H (8-24GB per stack) [32GB or 48GB in the future]
HBM

- One stack has about 1600 signal pins (micro bumps)
  - Micro bump pitch is 55um compared to 150um conventional C4 (relaxed to 110um in HBM3)
  - About 8x denser -> One HBM stack is equivalent to 1 RDIMM in terms of processor pin count (or PKG area)

- HBM is about 3x-4x more expensive ($/bit) than DDR memories (expected to go below 2x with wider adoption)

- Packaging cost is higher than DDR memories due to High Density Interconnect (e.g., silicon interposer)
  - Industry working hard to enable lower cost interposer (organic interposer, etc.)
Overview
Memory technologies
Memory bandwidth
Memory capacity
RAS (Reliability, Accessibility, Serviceability)
Memory controller
System Design Examples
Summary
RAS

- **Reliability**
  - A system's ability to operate continuously without failures
  - Or the probability that a system will produce correct outputs up to some given time $t$
  - MTBF (Mean Time Between Failure). Reliability = $\exp(-t/\text{MTBF})$

- **Availability**
  - Probability that a system is operational at a given time
  - Availability features allow the system to stay operational even when faults do occur. A highly available system would disable the malfunctioning portion and continue operating at a reduced capacity

- **Serviceability (or maintainability)**
  - Simplicity and speed with which a system can be repaired or maintained; if the time to repair a failed system increases, then availability will decrease
  - Serviceability includes various methods of easily diagnosing the system when problems arise. Early detection of faults can decrease or avoid system downtime

Ref. [9]
Example RAS feature examples

- **Reliability**
  - ECC (Error Correction Code), RAID (Redundant Array of Inexpensive Disk) / RAIM (redundant Array of Inexpensive Memory), memory mirroring,

- **Availability**
  - H/W downgrade mode, S/W dynamic page deallocation

- **Serviceability**
  - Memory health monitoring, automatically call deferred maintenance
Codes

- Codes without redundancy
  - \{0, 1\}
  - \{00, 01, 10, 11\}
  - \{000, 001, 010, 011, 100, 101, 110, 111\}

- Codes with redundancy
  - \{0, 1\} -> \{00, 11\}, \{000, 111\}
  - \{00, 01, 10, 11\} -> \{000, 011, 101, 110\}

- \{000, 111\}
  - Valid code words = \{000, 111\}
  - Invalid code words = \{001, 010, 011, 100, 101, 110\}
ECC (Error Correction Code) capability

- If distance between any two valid code words are:
  - $\geq 1$: nothing
  - $\geq 2$: single error detection
  - $\geq 3$: single error correction
  - $\geq 4$: single error correction & double error detection
  - $\geq 5$: double error correction
  - $\geq 6$: double error correction & triple error detection

SEC-DED

DEC-TED
Hamming code

- Distance between any two valid code words is 3
  - Single Error Correction
  - For ‘r’ number of check bits, message length (number of data bits) should be $\leq 2^r - r - 1$
  - For example:
    - $r=3$: data bits $\leq 4$
    - $r=4$: data bits $\leq 11$ (e.g. 8)
    - $r=5$: data bits $\leq 26$ (e.g. 16)
    - $r=6$: data bits $\leq 57$ (e.g. 32)
    - $r=7$: data bits $\leq 120$ (e.g. 64)
    - $r=8$: data bits $\leq 247$ (e.g. 128)
    - $r=9$: data bits $\leq 512$ (e.g. 256)
Extended Hamming code

- Add one more check bit to the Hamming code
  - Distance between any two valid code words is 4
  - Single Error Correction and Double Error Detection (SECDED)
  - For ‘r’ number of check bits, message length (number of data bits) should be \( \leq 2^{(r-1)} - r \)
  - For example:
    - \( r=4 \): data bits \( \leq 4 \)
    - \( r=5 \): data bits \( \leq 11 \) (e.g. 8)
    - \( r=6 \): data bits \( \leq 26 \) (e.g. 16)
    - \( r=7 \): data bits \( \leq 57 \) (e.g. 32)
    - \( r=8 \): data bits \( \leq 120 \) (e.g. 64)
    - \( r=9 \): data bits \( \leq 247 \) (e.g. 128)
    - \( r=10 \): data bits \( \leq 502 \) (e.g. 256)
Some of example codes

<table>
<thead>
<tr>
<th>Capability</th>
<th>SEC</th>
<th>SECDED</th>
<th>SBCDBD (Ext. Reed-Solomon)</th>
<th>DECTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data bits</td>
<td>Type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hamming</td>
<td>Extended Hamming</td>
<td>b=2</td>
<td>b=4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>32</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>64</td>
<td>7</td>
<td>8(^{(72,64)})</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>128</td>
<td>8</td>
<td>9</td>
<td>12</td>
<td>16(^{(144,128)})</td>
</tr>
<tr>
<td>256</td>
<td>9</td>
<td>10</td>
<td>14</td>
<td>20</td>
</tr>
<tr>
<td>512</td>
<td>10</td>
<td>11</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>1024</td>
<td>11</td>
<td>12</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>2048</td>
<td>12</td>
<td>13</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>4096</td>
<td>13</td>
<td>14</td>
<td>20</td>
<td>28</td>
</tr>
</tbody>
</table>

Ref. [8]
Chipkill

- Ability to keep data integrity even when a whole memory chip becomes bad
  - and to keep error correction capability using the remaining chips to keep running the system
Example-1: using (72,64) SECDED code

- Total 72 x4 DRAM chips required (64 for data, 8 for check bits) = 32B wide bus
- Works up to BL2 for 64B cache line processors, and BL4 for 128B cache line processors
Example-2: using (144,128) SBCDBD (b=4) code

- 36 x4 DRAM chips (32 chips for data and 4 chips for check bits)
- One x4 DRAM = 1 Byte

- 16B wide bus
- Works up to BL4 for 64B cache line processors, and BL8 for 128B cache line processors
Other RAS enhancement schemes

- CRC (Cyclic Redundancy Check) – detects transfer errors to retry
- Memory scrubbing
  - Periodically goes through all the memory slowly (reads and rewrites) to correct errors and avoid error accumulation
- Memory mirroring (2x provisioning)
  - Can apply to very critical memory region only (e.g. hypervisor)
- RAIM (Redundant Array of Inexpensive memory)
  - Adopt the same idea used for disks (RAID: Redundant Array of Inexpensive Disk)
  - Distribute data across 5 disks (DIMMs), then 1 disk failure (DIMM) can be totally recovered
- Sparing (have one more DRAM)
- Dynamic page deallocation
Overview
Memory technologies
Memory bandwidth
Memory capacity
RAS (Reliability, Accessibility, Serviceability)
Memory controller
System Design Examples
Summary
Memory controller

- A macro in a processor (CPU, GPU, or Accelerator) that communicates with off-chip memories (chips or modules)

- Required functions
  - Logically translates write/read requests into raw memory command protocols
  - Physically translates command, address and data into electrical signals (PHY)
  - To issue technology specific commands (refresh, etc.)
  - To guarantee mem. spec. timings (delays between commands)
  - To reorder commands to maximize throughput
Block diagram

L3 or DMA

WDATAin

READY

CMDin

Request queue (reordering queue)

Scheduler

Timers (per bank)

Dispatch queue

Rdata buffer

RDATAout

WDATAout

Wdata buffer

CMDout

RDATAin

PHY

To Mem

To Mem

KH Kim

ISSCC 2021 Tutorial

82 of 105
Timing restrictions

- There are tens of timing restrictions (delays) to be met between commands to:
  - Same bank
    - ACT to WR/RD, WR to PRE, ACT to PRE, PRE to ACT
  - Same bank group (same or different banks)
    - RD to RD, WR to WR
  - Different bank group
    - RD to RD, WR to WR
  - Same rank (same of different bank groups)
    - ACT to ACT, WR to RD, Four ACTs window, REF to ACT/REF
  - Different rank
    - WR to RD, RD to WR, RD to RD, WR to WR

- Hundreds of timers are needed to guarantee all the timings
Memory controller can more effectively reorder requests to minimize timing bubble (maximize bandwidth), if the scheduler has visibility over more entries (deeper reordering queue)
More efficient (smarter) scheduler design may achieve higher utilization
For any scheduler, deeper queue achieves higher utilization but with longer latency
Overview
Memory technologies
Memory bandwidth
Memory capacity
RAS (Reliability, Accessibility, Serviceability)
Memory controller
System Design Examples
- HPC
- AI
Summary
Design flow (HPC)

1. System target is given
2. Identify potential node architectures
3. For each node architecture:
   1. Determine number of nodes
   2. Derive memory BW and capacity requirements per node
   3. Choose memory technology(-ies)
   4. Examine feasibility (whether it meets power, cost, reliability, etc.)
4. Compare and select
Target

- System target
  - Compute power = 10 EFLOPS
  - Memory capacity (per system) = 100 PB

- Processors
  - GPU for parallel workloads
    - Compute power = 25 TFLOPS
  - CPU for management and sequential workloads
Node architecture options

- Homogeneous system (one type of node)
  - GPU only system
    - This is not actually practical because of variety of HPC workloads
  - CPU + GPU system
    - CPU:GPU = 1:N

- Heterogeneous system (more than one type of nodes)
  - Compute node
  - Memory node (different drawer)
  - Comp. node : Mem. Node = 1:M
Homogeneous GPU only system

- 1 GPU / node
  - Number of nodes = 10 EFLOPS / 25 TFLOPS = 400K
  - Memory capacity = 100 PB / 400K = 250 GB

- Memory bandwidth
  - Assume L2 cache miss rate = 5%
  - Memory BW = 5% x 25 TFLOPS x 8B = 10 TB/s
  - HBM3 @6.4Gbps has peak BW of 6.4Gbps x 128B = 820 GB/s
    - 12 x 820 GB/s = 9.84 TB/s
  - 16Gb x 12H stack = 24 GB
    - 12 x 24GB = 288 GB

- Meeting the capacity requirement with only HBM is too expensive!
  - HBM has 3~4x higher $/b than conventional (DDR) memories
  - 12H stack (doesn’t even exit today) vs. 8H stack cost ratio is more than 1.5 : 1 due to yield issue
Homogeneous CPU + GPU system

- 1 node = 1 CPU + 4 GPU (total 100K nodes)
- GPU
  - Memory bandwidth = $12 \times 820 \text{ GB/s} = 9.84 \text{ TB/s (12 HBM3)}$
  - Assume minimum HBM3 capacity
    - $8\text{Gb} \times 4H = 4\text{GB}$
  - Memory capacity = $12 \times 4\text{GB} = 48 \text{ GB}$
- CPU
  - Memory capacity = $100 \text{ PB} / 100\text{K} = 1 \text{ TB}$
    - $1\text{TB} = 8$ channels $\times 2 \text{ DIMMs/channel} \times 64\text{GB/DIMM}$
  - Memory bandwidth
    - DDR5-4800 -> $38.4\text{GB/s/channel}$
    - 8 channels $\times 38.4\text{GB/s} = 307 \text{ GB/s}$

- Achieves BW through HBM, capacity through DIMM
- One node is very big → cannot pack many nodes per drawer
Heterogeneous system - I

- Compute node (4 GPU / node) = total 100K nodes
- Memory node = total 12.5K nodes
- Compute node [per GPU]
  - Memory bandwidth = 9.84 TB/s (12 HBM3)
  - Memory capacity = 12 x 4GB = 48 GB
- Memory node
  - Memory capacity = 100 PB / 12.5K = 8 TB
  - 1 CPU + 32 DDIMMs
    - 1 DDIMM = 16Gb x 4H(3DS) x 32 sites = 256GB
    - 32 DDIMM = 8TB
- Smaller node → more nodes per drawer or rack
- Easier or more interconnection among GPUs
Heterogeneous system – II

- Compute node: same as previous
- Memory node: hybrid memory
  - 1 CPU + 4 DDIMMs + 4 NV-DDIMMs
    - 4 DDIMM = 4 x 64GB = 256GB
    - 4 NV-DDIMM = 4 x 2TB = 8TB
      - 1 NV-DDIMM = 1Tb (PCM) x 16 sites = 2TB
  - Alternatively, we can also do:
    1 CPU + 4 RDIMMs + 4 NVDIMM-Ps

KH Kim
ISSCC 2021 Tutorial
## Comparison – cost

<table>
<thead>
<tr>
<th></th>
<th>GPU only</th>
<th>GPU+CPU</th>
<th>Heterogeneous I</th>
<th>Heterogeneous II</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU:CPU</td>
<td>1:0</td>
<td>4:1</td>
<td>32:1</td>
<td>32:1</td>
<td></td>
</tr>
<tr>
<td>GPU count</td>
<td>$3000/EA</td>
<td>400000</td>
<td>400000</td>
<td>400000</td>
<td>GB</td>
</tr>
<tr>
<td>GPU memory</td>
<td>HBM</td>
<td>HBM</td>
<td>HBM</td>
<td>HBM</td>
<td>GB</td>
</tr>
<tr>
<td>GPU mem. cap.</td>
<td>288</td>
<td>48</td>
<td>48</td>
<td>48</td>
<td>GB</td>
</tr>
<tr>
<td>CPU count</td>
<td>$1000/EA</td>
<td>100000</td>
<td>12500</td>
<td>12500</td>
<td>GB</td>
</tr>
<tr>
<td>CPU memory</td>
<td>RDIMM</td>
<td>DDIMM</td>
<td>DDIMM + NV-DDIMM</td>
<td></td>
<td>GB</td>
</tr>
<tr>
<td>CPU mem. cap. (DRAM)</td>
<td></td>
<td>1024</td>
<td>8192</td>
<td>256</td>
<td>GB</td>
</tr>
<tr>
<td>CPU mem. cap. (PCM)</td>
<td></td>
<td></td>
<td></td>
<td>8192</td>
<td>GB</td>
</tr>
<tr>
<td>HBM total</td>
<td>115.2</td>
<td>19.2</td>
<td>19.2</td>
<td>19.2</td>
<td>PB</td>
</tr>
<tr>
<td>DRAM total</td>
<td>0</td>
<td>102.4</td>
<td>102.4</td>
<td>3.2</td>
<td>PB</td>
</tr>
<tr>
<td>PCM total</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>102.4</td>
<td>PB</td>
</tr>
<tr>
<td>HBM total $12/GB (4H, 8H)</td>
<td>2074</td>
<td>230</td>
<td>230</td>
<td>230</td>
<td>$ M</td>
</tr>
<tr>
<td>DRAM total $4/GB</td>
<td>0</td>
<td>410</td>
<td>410</td>
<td>13</td>
<td>$ M</td>
</tr>
<tr>
<td>PCM total $1/GB</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>102</td>
<td>$ M</td>
</tr>
<tr>
<td>Memory total</td>
<td>2074</td>
<td>640</td>
<td>640</td>
<td>346</td>
<td>$ M</td>
</tr>
<tr>
<td>Processor total</td>
<td>1200</td>
<td>1300</td>
<td>1213</td>
<td>1213</td>
<td>$ M</td>
</tr>
<tr>
<td>System total</td>
<td>3274</td>
<td>1940</td>
<td>1853</td>
<td>1558</td>
<td>$ M</td>
</tr>
</tbody>
</table>
## Comparison – other

<table>
<thead>
<tr>
<th></th>
<th>GPU only</th>
<th>GPU+CPU</th>
<th>Heterogeneous-I</th>
<th>Heterogeneous-II (hybrid)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU to GPU comm.</td>
<td>Good</td>
<td>Limited</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>CPU to GPU comm.</td>
<td>N/A</td>
<td>Short</td>
<td>Long</td>
<td>Long</td>
</tr>
<tr>
<td>Avg. Mem. Latency</td>
<td>1x</td>
<td>&gt; 1x</td>
<td>&gt; 1.1x</td>
<td>&gt; 1.2x</td>
</tr>
<tr>
<td>Avg. Mem. BW</td>
<td>1x</td>
<td>&lt; 1x</td>
<td>&lt; 0.9x</td>
<td>&lt; 0.8x</td>
</tr>
<tr>
<td>Reliability</td>
<td>Limited (HBM has no chipkill)</td>
<td>Good</td>
<td>Good</td>
<td>Limited (PCM has higher RBER)</td>
</tr>
</tbody>
</table>
Overview
Memory technologies
Memory bandwidth
Memory capacity
RAS (Reliability, Accessibility, Serviceability)
Memory controller
System Design Examples
  - HPC
  - AI
Summary
Target

- Universal training and inference server
  - Large memory capacity to store hundreds of models
    - 1TB
  - High memory bandwidth for training
    - 1TB/s
Start with HBM

- HBM3
  - Max. capacity = 24GB (16Gb x 12H)
  - Max. BW = 820 GB/s

- To meet the capacity target, we need 42 HBM3 chips
  - The BW will be 34 TB/s

- To meet the BW target, we need 2 HBM3 chips
  - The capacity will be only 48GB

- The problem is because BW/capacity of HBM is too high
**BW/capacity ratios**

<table>
<thead>
<tr>
<th></th>
<th>Max. BW per PKG [GB/s]</th>
<th>Capacity per PKG [GB]</th>
<th>BW / Capacity [1/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min.</td>
<td>Max.</td>
<td>Min.</td>
</tr>
<tr>
<td>HBM3</td>
<td>820</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>GDDR6 (x32)</td>
<td>36</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>LPDDR5 (x32)</td>
<td>25.6</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>DDR5 (x16)</td>
<td>12.8</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>DDR5 (x8)</td>
<td>6.4</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>DDR5 (x4)</td>
<td>3.2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>PCM (x4)</td>
<td>1.6</td>
<td>8</td>
<td>64</td>
</tr>
</tbody>
</table>

- LPDDR5 or DDR5 has the ratio close to 1
- LPDDR5 -> 64 LPDDR5 chips soldered down to achieve 1.64TB/s and 1TB
- DDR5 -> 20 DIMMs (64GB per DIMM) to achieve 1TB/s and 1.28TB
Idea

- Reading the target again,
  - “Large memory capacity (1TB) to store hundreds of models. High memory bandwidth (1TB/s) for training”

- The 1TB can be a ‘read mostly’ memory (endurance doesn’t matter)

- And not all 1TB needs to be high bandwidth, only if large enough to hold a working set
Solution

- **Hybrid memory**
  - Two 16GB HBM3-4800 chips to achieve: 4.8Gbps x 128B x 2 = 1.23 TB/s
  - Sixteen (eighteen for ECC) 512Gb PCM chips
  - Software to explicitly use split address spaces between training and inference (or model fetching and other jobs)

- **Form factor**
  - Might fit to M.2 SSD form factor
Summary
Summary

- **Memory metrics**
  - Bandwidth, latency, capacity, power, cost, reliability

- **Design variables**
  - Memory technology, architecture, form factor

- **Memory technologies**
  - DRAM (DDR, LPDDR, GDDR, HBM), emerging NVM

- **Memory Bandwidth** – core BW, I/O BW, how to achieve 100GB/s

- **Memory capacity**
  - Interface
  - Modules – 256GB module
  - Advanced packaging – DDIMM, HBM

- **RAS** - ECC, chipkill

- **Memory controller** – latency vs. BW

- **Design examples**
  - HPC
  - AI
Papers to see this year

- **25.1**: A 24Gb/s/pin 8Gb GDDR6 with a Half-Rate Daisy-Chain-Based Clocking Architecture and IO Circuitry for Low-Noise Operation

- **25.2**: A 16Gb Sub-1V 7.14Gb/s/pin LPDDR5 SDRAM Applying a Mosaic Architecture with a Short-Feedback 1-Tap DFE, an FSS Bus with Low-Level Swing and an Adaptively Controlled Body Biasing in a 3rd-Generation 10nm DRAM

- **25.3**: An 8Gb GDDR6X DRAM Achieving 22Gb/s/pin with Single-Ended PAM-4 Signaling

- **25.4**: A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications
References

1. JESD79-4, JEDEC DDR4 SDRAM specification
2. JESD79-5, JEDEC DDR5 SDRAM specification
3. JESD250, JEDEC GDDR6 specification
4. JESD235, JEDEC HBM specification
5. JESD209-5, JEDEC LPDDR5 specification
7. POWER® Processor-Based Systems RAS, IBM Systems Group