Let Software Decide:
Matching Application Diversity with One-Size-Fits-All Memory

Mattan Erez

The University of Texas at Austin

2010 Workshop on Architecting Memory Systems
March 14, 2010
Biggest Problems with Memory Systems?

- Never enough bandwidth
- Never enough capacity

- Cost
  - Optimize capacity/$
  - Optimize bandwidth/$?
  - Avoid niche products

- Power
  - Moving data takes energy (on-chip and off-chip)
  - Pins
Need to Work Within Commodity Constraints

- Devices and interfaces optimized for generic mass market applications
  1. Maximize capacity per unit cost
  2. Meet reasonable TDP
  3. Have reasonable reliability
  4. Maximize peak bandwidth
- Almost “one size fits all”
  - One memory type for each level of hierarchy
- Reliability a continued concern
- This is probably independent of underlying tech
- Add flexibility through architecture
  - Let software tune and optimize
Resulting Trends (detailed problems)

- Deeper hierarchies
  - Specialized storage per hierarchy level
    - SRAM, eDRAM, DRAM, NV-RAM, ...

- Increasing granularity of access
  - Increase capacity/$
  - Easy to “maximize” bandwidth/$
  - Cheaper reliability
  - Granularity goes up with capacity and distance

- Important tradeoffs between speed, BW, power, and reliability

- Have to optimize power/throughput/reliability

- Can’t do it without software involvement
• **Optical interconnect**
  - Nice to have eventually
  - Electronics still getting more competitive
  - Advantages not as big when all accounted for

• **3D Stacking**
  - Momentary relief, can’t satisfy capacity for large systems
  - Pushes the problem a little bit

• **PCM**
  - Early technology phase (for a very long time)
  - DRAM isn’t dead yet
  - DRAM can and should be re-optimized
Level Specialization

- Large storage is slow
  - Not necessarily inherent to the technology
  - Needed to keep cost and power reasonable

- Bandwidth drops quickly with distance from core
  - Energy of data transfer
  - Cost of switches and wires

- Inherent reliability of large storage is poor
  - Wearout
  - Soft errors
Granularity Examples

FIT numbers are rough estimates of mid-life devices assuming no ECC

<table>
<thead>
<tr>
<th>Device Type</th>
<th>FIT/bit</th>
<th>FIT/core</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>$10^5$</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>DRAM</td>
<td>$10^8$</td>
<td>$10^{-2}$</td>
</tr>
<tr>
<td>DRAM+Chipkill</td>
<td>$10^9$</td>
<td>$10^{-2}$</td>
</tr>
<tr>
<td>FLASH (small)</td>
<td>$10^6$</td>
<td>$10^{-6}$</td>
</tr>
<tr>
<td>FLASH (large)</td>
<td>$10^{10}$</td>
<td>$10^{-4}$</td>
</tr>
</tbody>
</table>

- Reads
- Writes
Coarse Access Granularity Tradeoffs

- Wasted bandwidth and power
  - Random/irregular accesses to large arrays
  - Potentially entire block for single record
  - Can be disastrous for many applications
  - Evident in many cache systems

- Lower redundancy for coding
- Simple management and often just works
  - If accesses are not too sparse
  - If cache at closer level is large enough
Impact of Granularity on Effective Throughput

- Application (even data) specific
  - Can redesign data structures and algorithm
  - Often lose efficiency at this level (but can gain overall)

Reduced throughput because of extra redundancy
Power-Related Tradeoffs

• Volatile memory:
  - Lower power if more errors allowed
    • Low supply voltage increases SRAM errors
    • Longer refreshes increase potential DRAM errors
  - Lower power if higher latency allowed
  - Lower power if lower bandwidth
    • Fewer banks, fewer activates, ...

• Non-volatile memory:
  - Lower power if lower bandwidth
  - Lower power (higher bandwidth) if longer bursts
  - Others?
    • Magic technology is rare
Example: DRAM

<table>
<thead>
<tr>
<th>SDRAM</th>
<th>clock</th>
<th>data</th>
<th>8B</th>
<th>Peak BW</th>
<th>8B random BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>200MHz</td>
<td></td>
<td></td>
<td></td>
<td>1.6GB/s</td>
<td>1.6GB/s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DDR</th>
<th>clock</th>
<th>data</th>
<th>16B</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>400MHz</td>
<td></td>
<td></td>
<td></td>
<td>3.2GB/s</td>
<td>1.6GB/s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DDR2</th>
<th>clock</th>
<th>data</th>
<th>32B</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>800MHz</td>
<td></td>
<td></td>
<td></td>
<td>6.4GB/s</td>
<td>1.6GB/s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DDR3</th>
<th>clock</th>
<th>data</th>
<th>64B</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1600MHz</td>
<td></td>
<td></td>
<td></td>
<td>12.8GB/s</td>
<td>1.6GB/s</td>
</tr>
</tbody>
</table>
Example: DRAM

- Data Bus Sequential Peak
- Data Bus Random Peak
- Address Bus Limit (3 cmds/req)
- Address Bus Limit (2 cmds/req)

Bandwidth [GB/s]

SDRAM-200, DDR-400, DDR2-800, DDR3-1600
Fine-grained access with ECC:
Cray Black Widow (SEC-DED)

32bit (40bit) data bus – 16B min access granularity

Need many DRAM channels for higher BW
ECC overhead = 25%, but only SEC-DED
Pin increase – more A/C per data

Cray BW vector multiprocessor systems

BW0  BW1  BW2  BW3

...  ...  ...  ...

Weaver0  ...  Weaver15

...
Fine-grained access with ECC: Convey S/G DIMM (SEC-DED)

S/G DIMM

72-bit wide data bus

FPGA

Memory Controller

Custom A/C bus 4x to 8x faster than normal A/C bus

WAMT 2010

©Mattan Erez
To support 8B access granularity with DDR3
Bandwidth and Capacity are too Valuable to Waste - Expose Tradeoffs to Applications

- Need to maximize capacity & throughput per $
  - This is what we started with!

- Solution: let the application decide on granularity and redundancy
  - Granularity, redundancy, and power go hand in hand

- Flexible and dynamic granularity (and ECC)
  - Potentially at all levels

- Explicitly managed memories with bulk transfers
  - “Generalized Streaming” software managed asynchronous gathers and scatters
  - Sometimes, software knows best – requires a hierarchy of control – not fine-grained loads and stores
  - Amortize cost of control and maximize throughput
  - Memory aggregators
Flexible Granularity/ECC – Observations

- **Error detection (and minor correction) – common**
  - Need low cost, low overhead error detection

- **Strong error correction – uncommon**
  - Correction can be slow
  - But, still need to maintain error correction info somewhere

- Memory hierarchy provides inherent redundancy
  - Only dirty data needs error correcting codes
- Some data doesn’t need stringent protection
- Part of storage arrays more/less reliable
Solution: Tiered Error Protection – Works at all Levels

• Tier-1 Error Code (T1EC)
  - Light-weight error code (simpler)
  - Uniform error protection
  - Relaxes design constraints

• Tier-2 Error Code (T2EC)
  - Strong error codes only for dirty data
  - Corrects T1EC Detected but Uncorrected Errors (DUE)

• Flexibility in storing T2EC Information
  - Can be offloaded to cheaper levels of hierarchy

• Flexibility in choosing the codes
  - Potentially based on individual memory regions or pages
  - Minimal impact on performance
Flexible ECC for Caches

ECC increases area, leakage/dynamic power
Flexible ECC for Caches (Memory-Mapped ECC)

8 ways

T2EC is memory mapped to cached DRAM namespace
Managing T2EC information

A physical cache line is associated with a T2EC in DRAM.

T2EC Read is avoided when all these lines are clean.

Last Level Cache

Data + T1EC (64B+1B)

T2EC Write

T2EC Read

T2EC Write

T2EC (8B)

T2EC in DRAM

WAMT2010
Two-Tiered Error Correction Examples

- S - SEC-DED
- D - DEC-TED
- PS - Parity T1, SEC-DED T2 (both interleaved)
- PN - Parity T1, nibble-based T2 (both interleaved)
- PB - Parity T1, byte-based T2 (both interleaved)
- SD - SEC-DED T1, DEC-TED T2 (both interleaved)
Two-Tiered Error Correction Examples (zoomed)

- **S** - SEC-DED
- **D** - DEC-TED
- **PS** - Parity T1, SEC-DED T2 (both interleaved)
- **PN** - Parity T1, nibble-based T2 (both interleaved)
- **PB** - Parity T1, byte-based T2 (both interleaved)
- **SD** - SEC-DED T1, DEC-TED T2 (both interleaved)
Two-Tiered Error Detection Examples

- S - SEC-DED
- D - DEC-TED
- PS - Parity T1, SEC-DED T2 (both interleaved)
- PN - Parity T1, nibble-based T2 (both interleaved)
- PB - Parity T1, byte-based T2 (both interleaved)
- SD - SEC-DED T1, DEC-TED T2 (both interleaved)
Flexible ECC for DRAM

Virtual Memory

VA

VPN

offset

VPN

offset

Virtual Memory

PA

PFN

offset

Page Frame

PA

Physical Memory

Data

ECC
Flexible ECC for DRAM (Virtualized ECC)

Virtual Memory

VA

VPN

offset

PA

PFN

offset

OS manages PFN to EPN translation

Scale according to T2EC size

ECC Address

ECC page number

offset

Physical Memory

Page Frame

Data

T1EC

T2EC

EA

ECC page
Flexible ECC for DRAM

Virtual Address space

- Chipkill-Detect
  - Virtual page – i

- Chipkill-Correct
  - Virtual page – j

- Double Chipkill-Correct
  - Virtual page – k

VPN to PFN mapping

- Page frame – i
- Page frame – j
- Page frame – k

PFN to EPN mapping

- ECC page – j
- ECC page – k

T2EC for Chipkill

T2EC for Double Chipkill

Page frame

Data

T1EC

27
Flexible Granularity Access & ECC

- **64B granularity**
  - Data: 64B
  - Virtualized ECC: 8B
  - Data Density: 0.89

- **32B granularity**
  - Data: 32B (2x16B)
  - Virtualized ECC: 8B (2x4B)
  - Data Density: 0.8

- **16B granularity**
  - Data: 16B (4x4B)
  - Virtualized ECC: 8B (4x2B)
  - Data Density: 0.67

- **8B granularity**
  - Data: 8B (8x1B)
  - Virtualized ECC: 8B (8x1B)
  - Data Density: 0.5
Flexible granularity accesses:
Threaded – RAMBUS, MCDIMM – HP, …
Flexible granularity accesses: Subranking based DRAM system

ABUS

DBUS – 8 8-bit buses

Double Data Rate bus

Register / Demux

x8 sr 0
x8 sr 1
x8 sr 2
x8 sr 3

Partition 0

x8 sr 4
x8 sr 5
x8 sr 6
x8 sr 7

Partition 1

Quad Data Rate bus

Register / Demux

x8 sr 0
x8 sr 1
x8 sr 2
x8 sr 3

Partition 0

x8 sr 4
x8 sr 5
x8 sr 6
x8 sr 7

Partition 1

Partition 2

Partition 3

WAMT 2010

©Mattan Erez
Random 8B reads

**1-subrank system**

- ABUS utilization [%]
- DBUS utilization [%]
- Throughput [GB/s]

**8-subrank system**

- ABUS utilization [%]
- DBUS utilization [%]
- Throughput [GB/s]

**8-subrank system + 2 ABUS**

- ABUS utilization [%]
- DBUS utilization [%]
- Throughput [GB/s]

**8-subrank system + 4 ABUS**

- ABUS utilization [%]
- DBUS utilization [%]
- Throughput [GB/s]
Conclusions

• Commodity memory is designed for capacity first
  – Leads to compromises, especially granularity
• Coarse-grained accesses are efficient from the memory perspective
  – Simple, low command BW, better ECC
• Fine-grained accesses are good for applications
  – No wasted BW, storage, and power
• Flexible granularity is important
  – Tradeoffs need to be exposed to the application
• Works for DRAM and SRAM
  – Generalized “streams”
    – Virtualized ECC, Memory-mapped ECC
• Should work for future technology as well