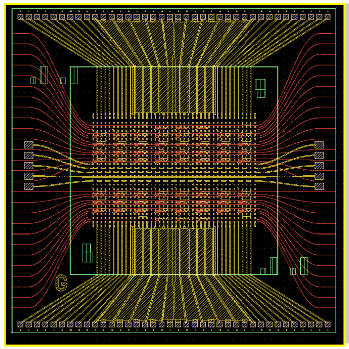
Exceptional service in the national interest

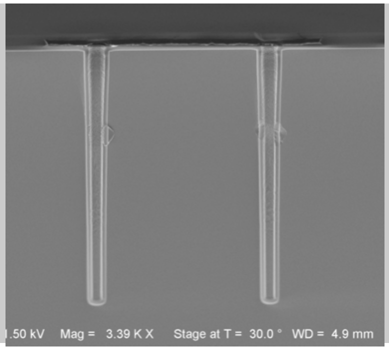
New Resilience Capabilities with Micron's HMC

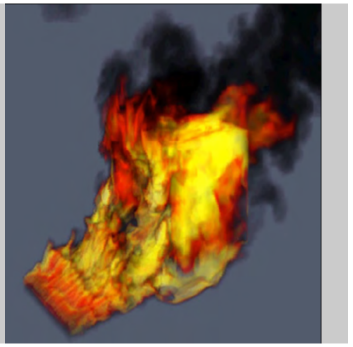
Use Micron's **H**ybrid **M**emory **C**ubes to enable full recovery from chip and module failures

Dave Resnick
Sandia National Laboratories

SAND 2015-2835 C







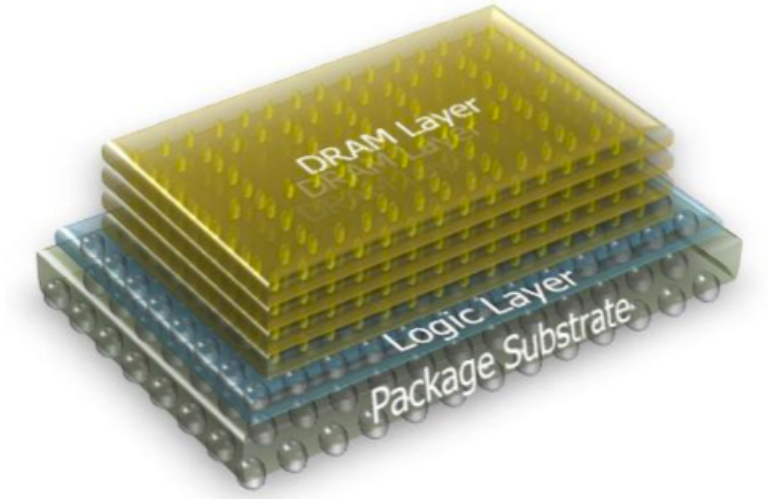
Goals for Memory

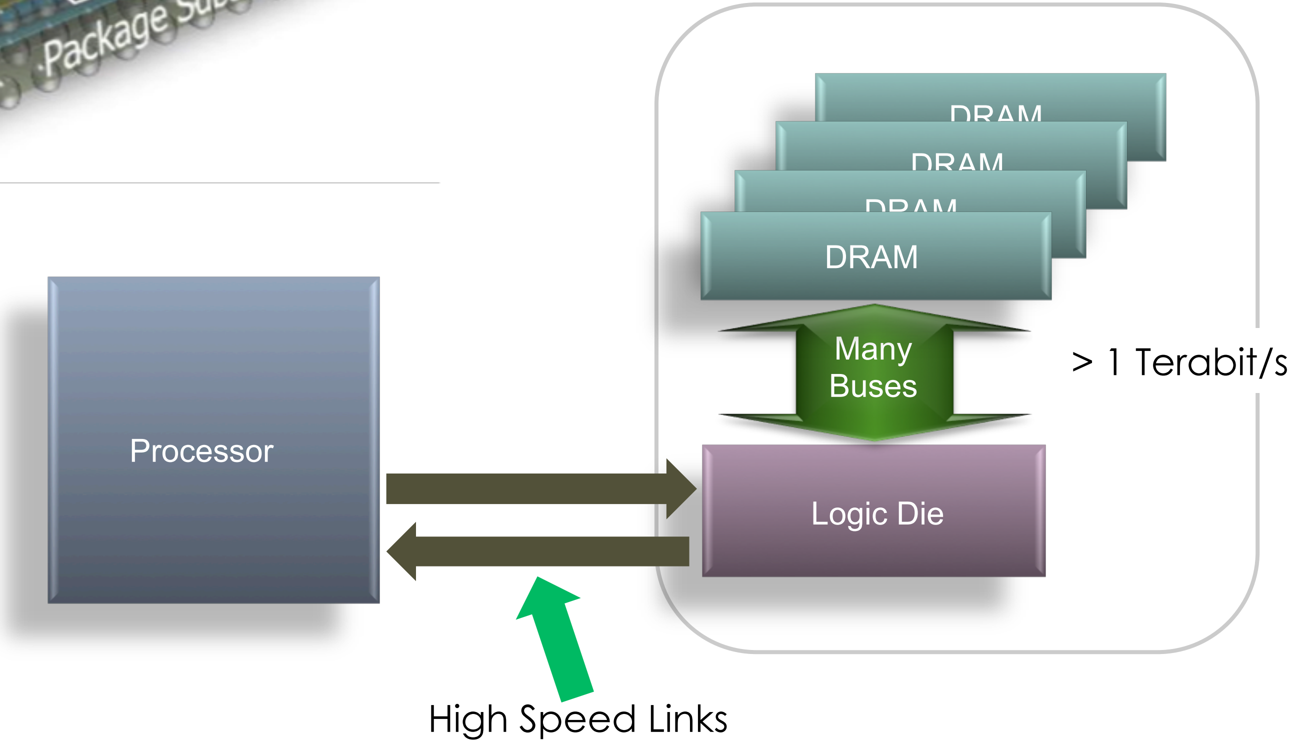
- HPC Objectives
  - Breakthrough low energy
  - Very high to extremely high bandwidths
  - Extensive scalability: Very big to *amazing!* memory sizes :-)
  - Capability of extreme resilience without being a burden
- An architecture that can provide different amounts of memory with a constant bandwidth for the different sizes
- An architecture that can provide different bandwidths for multiple customer requirements
- Provide a base for putting significant accelerant capability in the memory and system network if/as that is worthwhile
- Make a structure that can also support integrated non-volatile memory in a node's main memory system

About HMC

- Stacked die—4 or 8 high—with a CMOS logic chip at the base of the stack that has multiple memory controllers and IO
- Current parts (shipping)
  - 2 and 4 Gbytes
  - 128 (2GB), 256 (4GB) memory banks
  - To 240 GB/sec bandwidth
- Next generation (announced, likely 2017?)
  - 4 and 8 Gbytes
  - 256 (4GB), 512 (8 GB) memory banks
  - To 320 GB/sec bandwidth
- Use about 1/6th the energy per bit than current DRAMs (and future versions will further improve this)
- Bit serial (SERDES) interface for reduced IO count

High Level View of HMC





HMC Features

- HMC parts enable features and functions that no other memory technology can match
  - Multiple ports that access all of the cube
  - High speed SerDes with an abstract protocol and interface
  - Chaining: Multiple parts in series with fairly small latency impact
- Using chaining, along with other features can get:
  - Low energy levels and high memory system bandwidths for small servers to future supercomputers
  - Memory scaling that achieves the memory size and performance goals of exascale systems, while being excellent for smaller systems
  - Error detection and recovery that can make memory systems reliable enough for exascale (10 million memory parts!) while not being a requirement or burden for systems where that level of reliability is not needed. (But see the next slide)

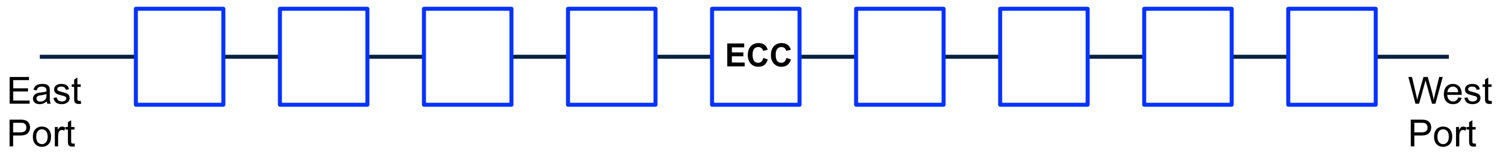
More on HMC Features

- The HMC stack's logic base enables/provides additional capabilities above current memory parts
  - Atomic operations
  - Abstract interface reduces logic in the interface and enables higher efficiency, like a single request for 128 bytes of data rather than 2 requests, or a request that returns 16 bytes of data rather than 64
  - Local (in-package) error detection and recovery for 4-bit symbols
  - In-field failure recovery for things like stuck bits and bad rows (maintenance without replacement)

The resiliency features shown here are proposals only. Micron has made no commitment to implement them.

The descriptions of Micron HMC capabilities in the following slides are covered by Micron patents. The implementation of these capabilities will benefit from demand from the HPC community to influence Micron's plan of record.

One Memory Chain



Nine cubes in 4 + 1 + 4 configuration

- Each end of the chain functions independently except for error recovery operations.

  Double bandwidth from a single link
- Access at each end means shorter read and write paths: further reduce power and latency
- Read operations only read the cube holding the referenced data unless there is an error.
- Write operations send recovery data to the ECC cube (2nd write)
- East and West ports, on each cube and at the module level, are a port-pair. This has functional implications explained below

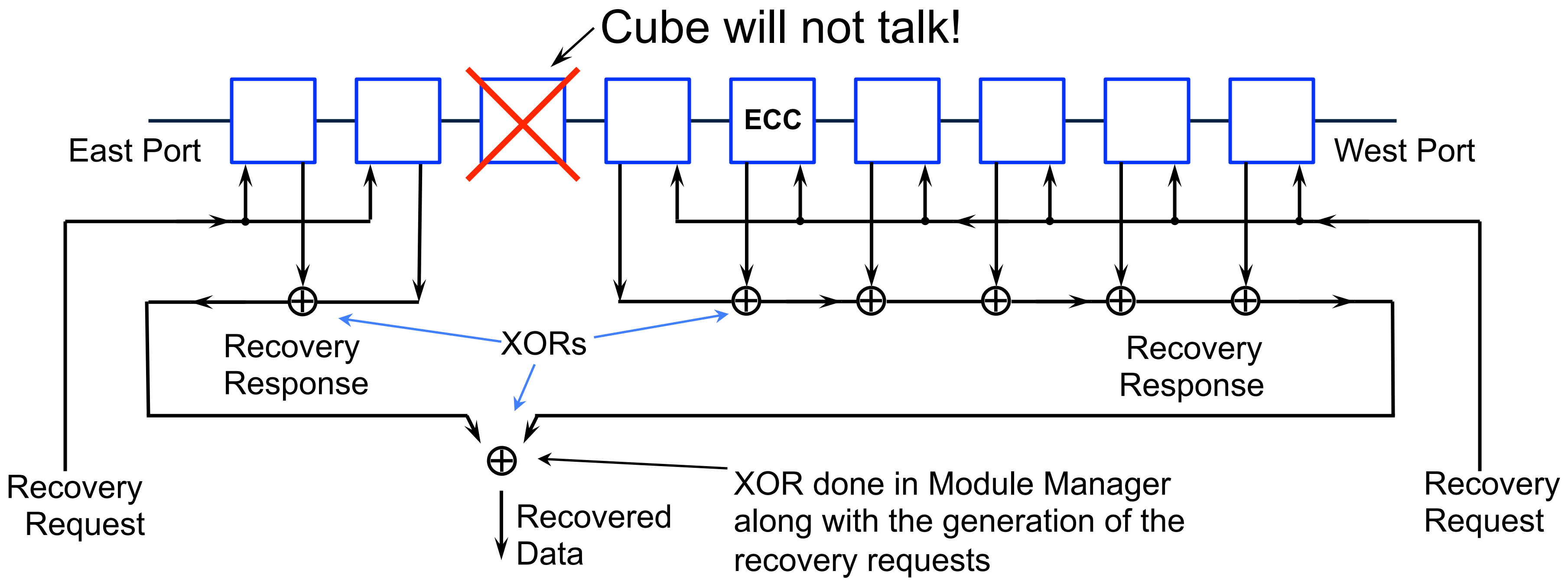
On Writing in Chains

- A write to a cube in a chain that supports this ECC does two write operations, one to the data cube being addressed and one to the ECC cube. The ECC update is the difference between new & old data and is the result of a write packet generated by the data cube
- When a read or a write is done to a data cube, the reference request is removed from the chain. Cubes further down a chain see a reduced number of requests—less chain bandwidth used. Request traffic 'tapers' between the requesting port and the ECC cube
- The ECC update write thus has chain bandwidth to do the update without creating 'additional' chain conflicts. Write traffic is uniformly distributed along the chain and is the same basic traffic pattern and bandwidth as if all writes go to the ECC cube.

Power in Chaining

- A reference, in only chaining through a cube, is very low energy (~1 pJ per bit)
- A single reference to a cube takes only a fraction (~1/4th) of the max power for that cube
- Each cube supports one or two chains. A data cube thus sees one-quarter or -half of the published maximum
- The result is that the power seen by a chain is normally *much* less then max_power * chain length
(~ 2 pass-thrus + 1 cube being referenced)
- A write reference to a chain increases power because a 2nd write is done to the ECC cube

Cube & Link Error Recovery: **CubeKill**

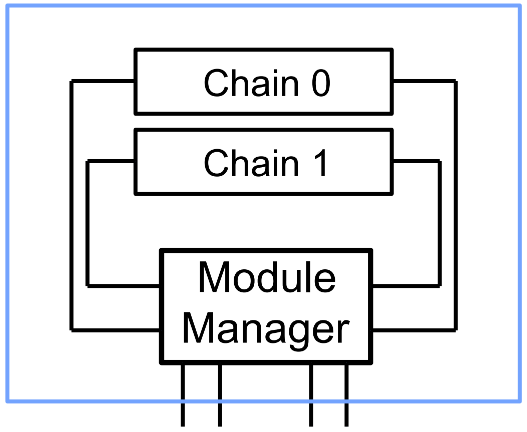


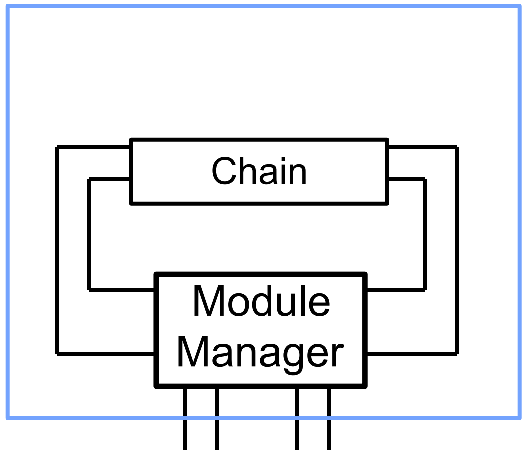
- Send a recovery request down both ends of the chain, indicating the place the chain is broken or data is corrupt
- XOR the responses. The result is the recovered data
- Not described here, but can also 'write' to a failing cube such that the new data can be correctly read

Error Recovery Power

- A recovery reference does reference all the cubes in a chain (but the failing cube), so power is higher
- Each of the referenced cubes sees about 1/4 power so total is about 1/4th of the otherwise expected maximum

HMC Chained Memory Modules

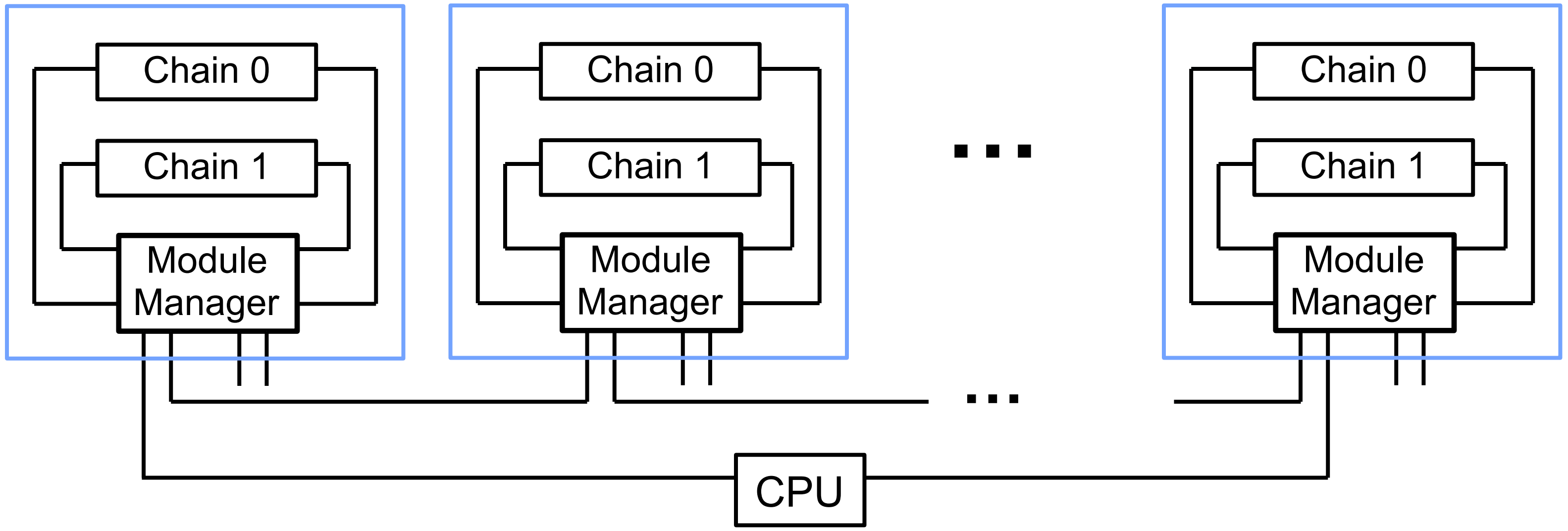




- If a request is not for this module, chain the request out the paired port
- The manager routes a request to indicated chain port
- The manager has logic that generates on-module recovery sequences as needed, implementing CubeKill and supporting ModuleKill
- Many module variations possible varying the number of external link paths and the number and length of internal cube chains

ModSets and ModuleKill

- Make a chain of 4 + 1 (ECC) + 4 modules. This is a "ModSet"
- Connect to both ends of a module chain



- Implement the same error recovery mechanism as done for CubeKill in a module, but with logic in each CPU functioning for error recovery as the Module Manager did: **ModuleKill**

Can Use ModuleKill to Better Recover From a Failing Module

- On a module failure, operations continue with 'recovery' commands to the ModSet
  - Takes one (read) or two (read, write) ModSet recovery references
  - Increases power as all modules in the ModSet are referenced
- While system operation continues, in the background can rebuild data from the failing module into the ECC module
  Details available on how to do this
- Then use the ECC module in place of the failing module
  - No more recovery commands, lower power, full performance
    But of course schedule maintenance, as are then vulnerable to another failure :-)
  - Takes a small increase in logic in Module Managers and additional logic in the interface to each ModSet (remap module IDs, run the rebuild sequence)
- The same idea can be used for CubeKill recovery

Resiliency Summary

- Fully recover from a failing memory part in each local chain; this includes recovering from failing links/memory paths
- Can fully recover from a failing memory module and failing link at the cost of an additional module per ModSet
- Can leave out CubeKill and ModuleKill capabilities if not needed—for lower end and lower cost systems
- Most failures will be individual cubes. Those failures are recovered at the module chain level, so can have multiple failures in a system and system keeps going without impact on users
- If a whole module fails, then each module in a ModSet is referenced which increases power in the Set, but power still low compared to other technologies

Super-Node's Memory System?

Possibility: Four module sets (to 648 cubes @ 16 GB per cube)

Size:	4 * 2 Tbytes	= 8 Tbytes of memory
Bandwidth:	4 * 480 GB/sec	= 1.9 TB/sec bandwidth

- Most any in-module error is recoverable at the Manager level
- Any full module or module link error is recoverable at memory interface
- HMC provides a bridge to enable current applications to see large improvements in performance with minimum changes, as apps are (gradually) upgraded to use new architectural features
- Easy to have versions that can be high volume, and so attractive as a general memory set of products for smaller/lower cost systems

HMC and Exascale Memory

- Believe that HMC technology enables reaching exascale system goals.
  - Memory sizes to 250 Pbytes within power limits
  - Memory bandwidths to 100s of PB/sec
  - Memory that users will not see fail
- Believe that no other memory technology comes close.
  (At least all at the same time, so within the limits of 20 MW power at expected size, etc.)
- HMC is expected to integrate well with local non-volatile memory (whatever that turns out to be)

More on HMC Benefits

- HMC enables features that provide benefits over and above those of previous memory components. This means that the memory is more valuable than the base 'bit-cost' of current DDR memory
  - Much larger and faster runs for memory limited apps
  - Less effort in bending applications to run well on new systems
  - Atomic operations reduce coherency traffic
  - Lots more memory references at the same time for improved execution efficiency
  - Place for more intelligence in a system like multi-dimensional Move operations to improve cache use. Searches in memory,….. and more
- This means higher performance and reduced power with systems that have less total hardware then existing system scaling rules would indicate

If you see solutions to issues here and opportunities in these ideas, lets encourage having the components and modules actually being brought into the market place

Thanks!