



The Necessity of Integrated, Abstracted Memory and Storage and Moving Forward

SC14 Birds of a Feather: Abstracted Interface for General Usage

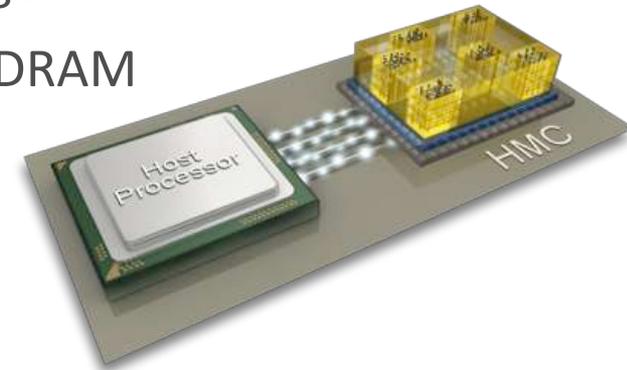
J. Thomas Pawlowski, Chief Technologist, Micron Fellow
jpawlowski@micron.com [LinkedIn jtpawlowski](#)

Memory Abstraction

Hide unnecessary idiosyncrasies and exploit the new-found freedom

Numerous historical abstracted memory and storage

- Micron Zero Bus Turnaround SRAM, 1990's
- PSRAM, 2000's (Micron CellularRAM 1, 2, 3 fully-abstracted DRAM)
- RLDRAM (abstracted row,column,refresh)
- Flash cards, eMMC, UFS, SSD – abstracted storage
- Micron Hybrid Memory Cube – fully abstracted DRAM



Memory Abstraction is not the end goal

- Brings the industry to the new starting gate
- Facilitates a better future balance between memory technology and logic technology devices
- Creates necessary new degrees of freedom

Degrees of Freedom Required in Future DRAM and NAND

Bit, page, block, bank, refresh, erase ... management

Error management

Yield management – 2D but especially 3D

Ensuring performance improvements and continuing cost reduction

Relief from bottlenecks of IO space, storage semantics, sub-optimal PHYs

Scaling issues continue to compound

- New charge coupling issues expected in DRAM with each new node
- Intrinsic and extrinsic NAND issues change with each new node
- Timing parameters of both “want” to change with each new node

Enable real memory innovations

Necessity of Memory Abstraction

To roll out new technologies

- DRAM and NAND replacement paths likely differ between vendors
- Wildly different characteristics from their predecessors
- Potentially new hierarchical elements to be introduced
- Already, NAND implementations have differed significantly
 - Abstracted NAND already dominates the market
- How can customers move all designs in lock-step for new tech?

To synchronize all sides of the supplier and consumer equation

Abstraction allows vendors and users to seek same memory “socket” with more optimal implementations

Compatible Black Boxes, but not identical

- Not all workers have the same skill, allows differentiation and innovation

Future: no memory device is directly controlled by Host

Some Required Building Blocks

Very high efficiency IO

- LPDDR4+, new SERDES, new HS single-ended, new efficient encoded, new optical

Operating System modifications

- Shift some current functions to hardware without breaking systems

Many logic blocks, of course

- New memory controllers, protocol handlers, buffer modules, etc.

Optional accelerator blocks

Protocol

Very high efficiency protocol to minimize overhead

- Micron RIGEL is in draft v7, incorporated internal and external feedback
- More efficient than anything else published
- Modularity to support markets: SmartPhones, Workstations, Server varieties, HPC
- Ability to encapsulate existing protocols when needed
 - Of course with efficiency loss on prior art protocol, negligible RIGEL impact
- Enables many new capabilities

Rigel is well advanced, great framework to incorporate everyone's feedback

Rigel

Protocol Development Method and Goals

Agreement first required on the operating goals and principles

- Details resolve more easily with everyone on the same page

Create a single protocol template once

- À la carte tailoring for specific system types, not just for HPC

Enable any combination of memory and processing

- Single bus systems, Many bus systems, In-memory processing, Accelerator processing, Network devices, Storage appliances, Hierarchical memory, Any abstracted memory, Any to Any comm

Enable future innovations that we can't even anticipate now

Minimize overhead

Support any physical bus choices

- electrical, optical, encoding, lane count, channel count, speed, style, instantiations in system

Numerous Issues

Minimize Energy

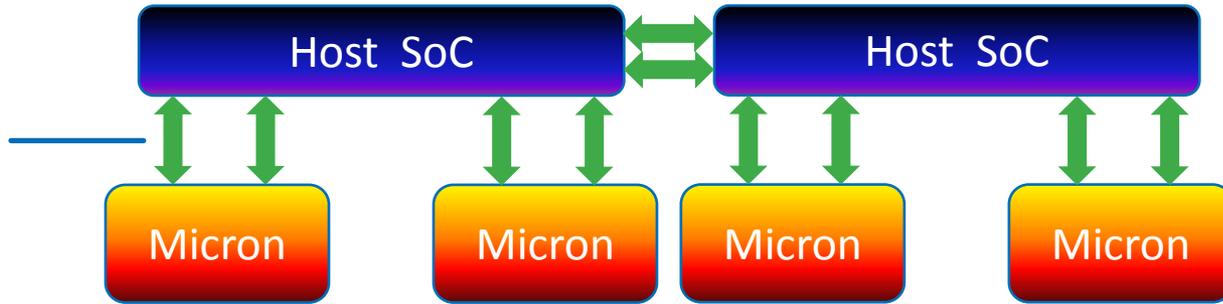
- Energy of command sequencing outstrips compute energy
- Energy of data movement outstrips compute energy
- Low-touch data inflates energy usage
- Big Graph, Big Data has poor locality = low-touch data

Maximize Throughput

- Enable high bandwidths, high concurrency, low latency, high outstanding transaction count

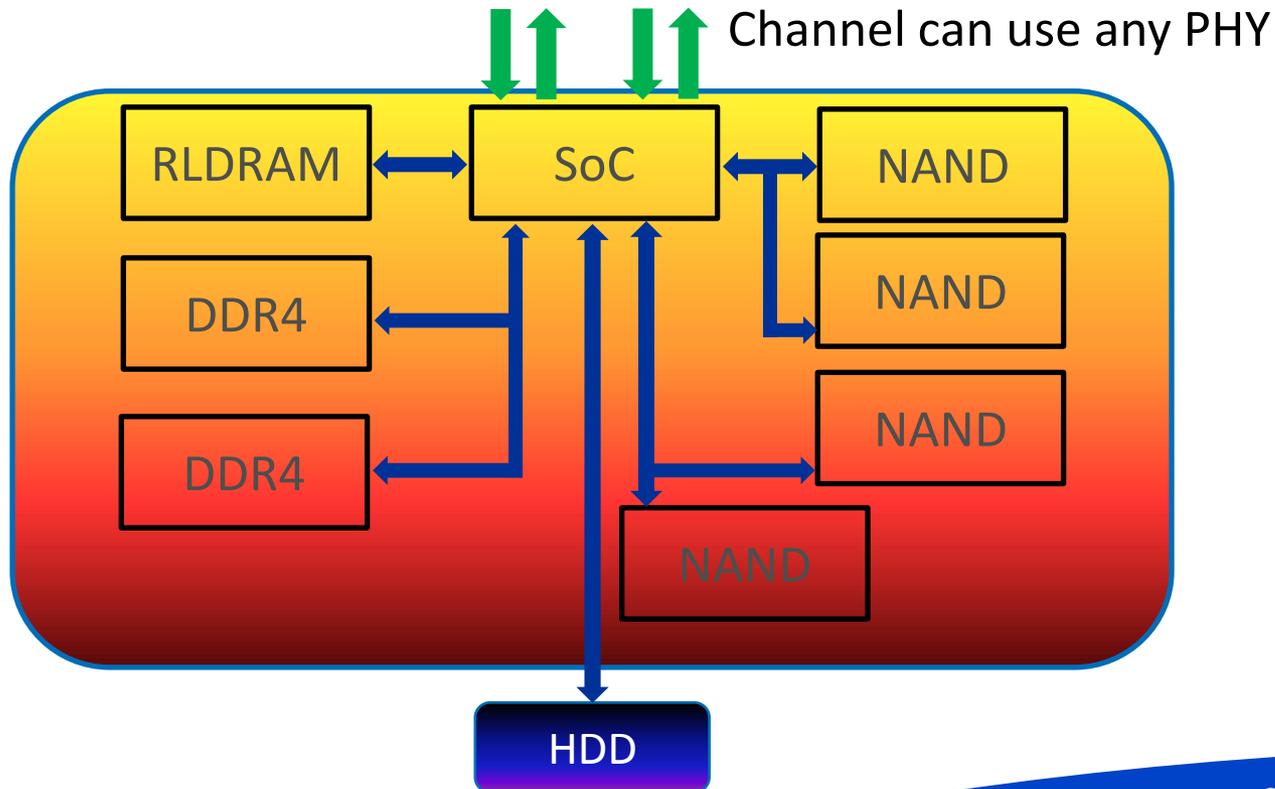
Where Protocol Fits in a System

Rigel protocol running on Channel



Example of many memory types in single module

Future memory types are especially interesting and seamlessly incorporated



Rigel v7 Packet Level View

Example of a Protocol Meeting the Goals

From a Packet Transport and Parser perspective, there are three fields

- Size
- Payload - size determined from SIZE field
- ECC – size determined from SIZE field



Simplest possible protocol supporting variable payload

PHY requirements would integrate into Rigel, together = BUS

- Data Bus Inversion, Link control codes such as framing must be ECC protected
- If added as separate layer, it would increase overhead too much

Important Principles 1-9

1. Packetized protocol
2. Minimize bit movement – treat like minimizing silicon area
3. Facilitate transactions which are sent to another device (e.g. atomic ops/RMW, moves/DMA, chip to chip operations)
4. Overhead optimized for successful transactions
5. Carefully trade off capability and simplicity
6. Avoid transmitting that which is already known
7. Minimize error control overhead
8. Minimize protocol field count
9. Allow field size configuration within reason

Important Principles 10-18

10. Abstract transaction semantics only (not memory-, not storage-semantics)
11. Make protocol fields optional
12. Harmonious coexistence of low- and high-latency devices/memory types
13. Enable complex operations while maintaining protocol simplicity
14. Permit unanticipated future functions without disrupting legacy
15. Provide “knobs” and visibility to the system designer
16. Provide means for controlling transaction ordering
17. Localized domains of ordering
18. Remain faithful to the concept of abstraction

Important Principles 19-28 (last)

19. Provide means for flow control
20. Ability to encapsulate other protocols
21. Support multiple data widths and rates
22. Resilience capabilities
23. Open standard
24. Support variable length packets
25. Support Full Duplex (not really a protocol issue, but needed)
26. Start-up dynamic configuration (negotiation)
27. Sensible error detection and handling (some at component, some at module, some at host)
28. Support interpretable address space, e.g. Global, Segmented

