



A Memory System Design Framework: Creating Smart Memories

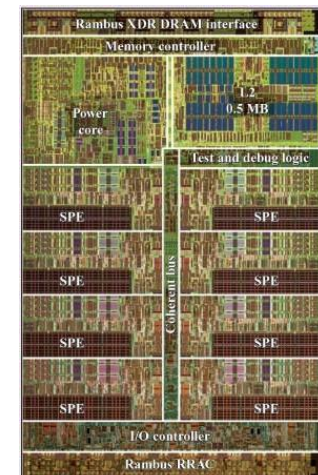
Amin Firoozshahian, Alex Solomatnikov

Hicamp Systems Inc.

Ofer Shacham, Zain Asgar,
Stephen Richardson, Christos Kozyrakis, Mark Horowitz

Stanford University

- ## Intel Nehalem





Looking a Little More Closely

Sun Rock





Reality...

- **Replicated cores**
- **Incredibly complicated memory system**
 - Large amounts of logic
- **Innovation is in the memory system**
 - Transactions, streaming, fast synchronization, security, etc.
- **Never exactly the same**
- **Where all the bugs are!**



ISA for Memory Systems

- *Can we regularize the memory system hardware?*
- “Program” it rather than “Design” it?
- **Benefits:**
 - Reduce design time
 - Patch errors
 - Run-time tuning
- **How can we do this?**



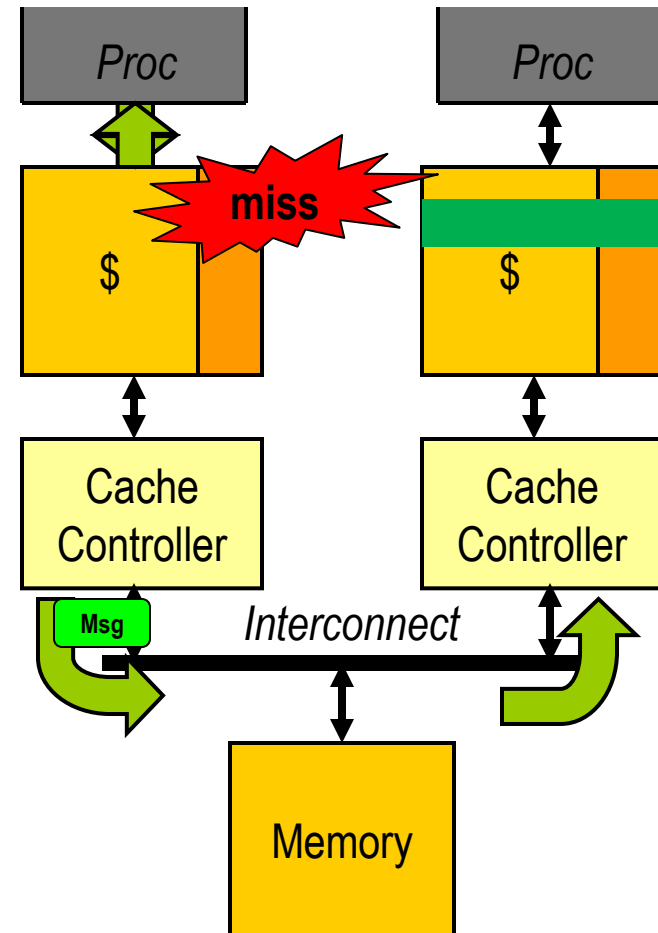
Shared Memory System

■ Resources:

- Local memory
 - Data, state bits
- Interconnect
- Controllers

■ Operations:

- Probing state bits
- Track requests
- Communication
- Data movements (spill / refill)





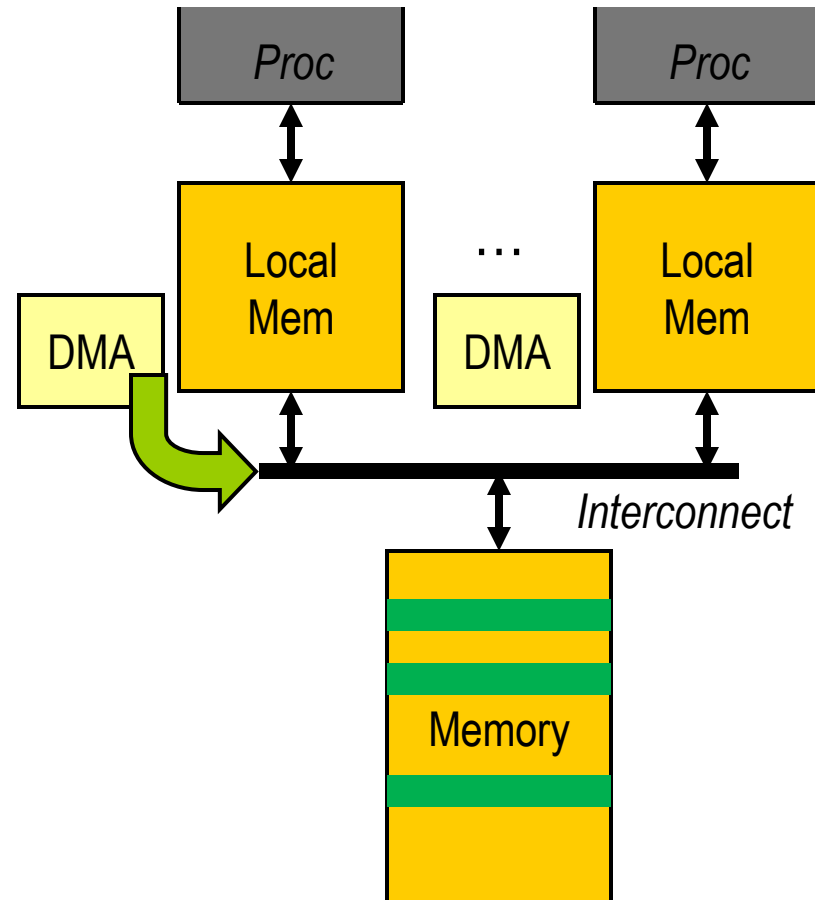
Streaming Memory System

■ Resources:

- Local memory
- Interconnect
- Controllers

■ Operations:

- Communication
- Data movements
- Track outstanding transfers





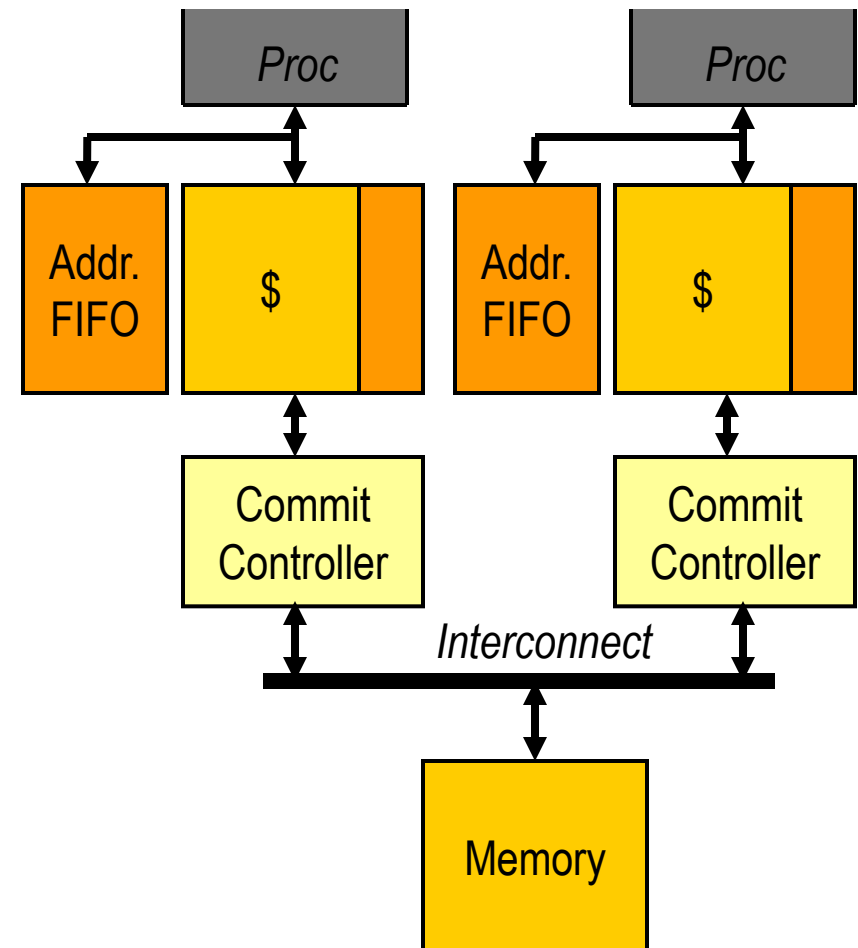
Transactional Memory System

■ Resources

- Local memory
 - More state bits
- Interconnect
- Controllers

■ Operations

- Data movements
- State checks / updates
- Communication





Commonalities

- **Same resources and operations**

- **Different in:**
 - How the operations are sequenced
 - Interpretation of state bits

- **We need:**
 - Flexible local storage and interconnect
 - Programmable controllers



Local Memories

- **Programmable memory mat**

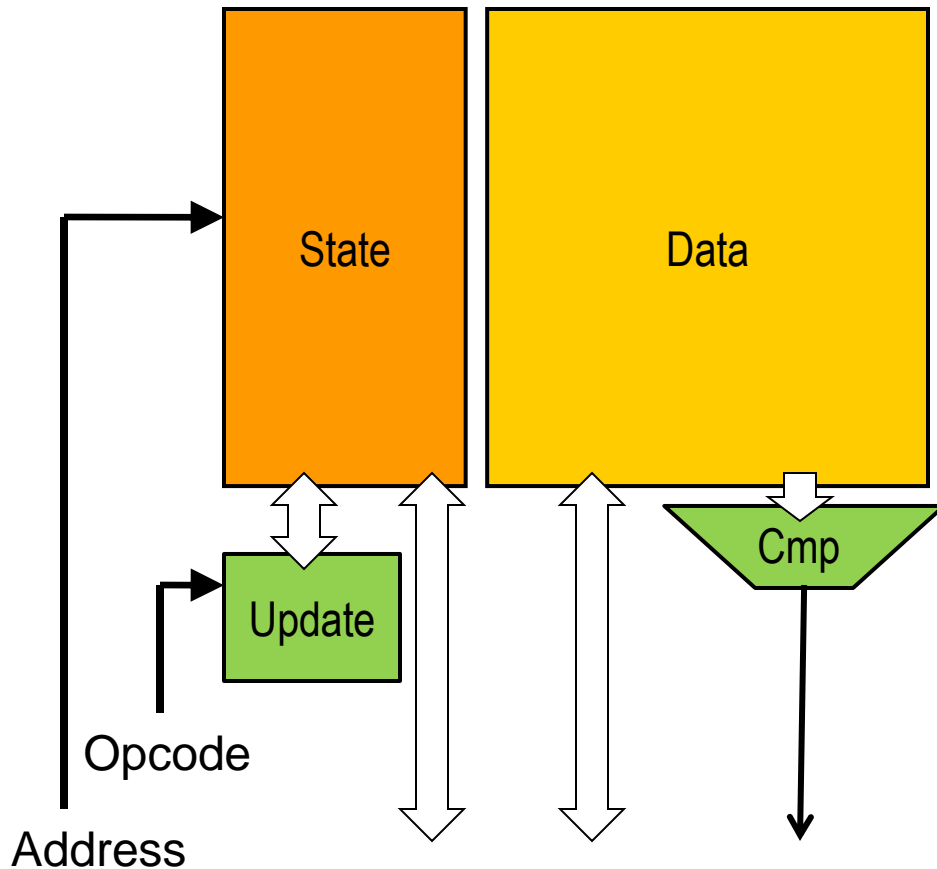
- Data array
- State bits
- PLA logic
- Comparator

- **Accessed by**

- Address, Opcode

- **Returns**

- data, state, compare result





Programmable Controllers

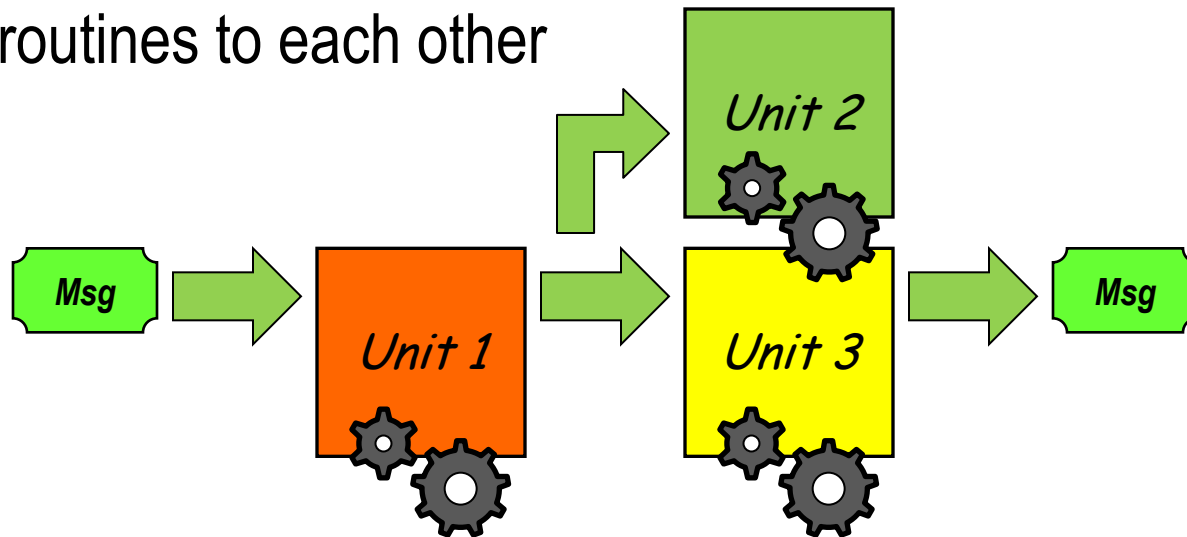
- **Use an off-the-shelf processor?**
 - FLASH, Typhoon, etc.

- **Too slow**
 - All the way to the L1 cache interface

- **Our approach:**
 - Micro-coded engines (functional units)
 - Each class of operations in a separate engine

Programming

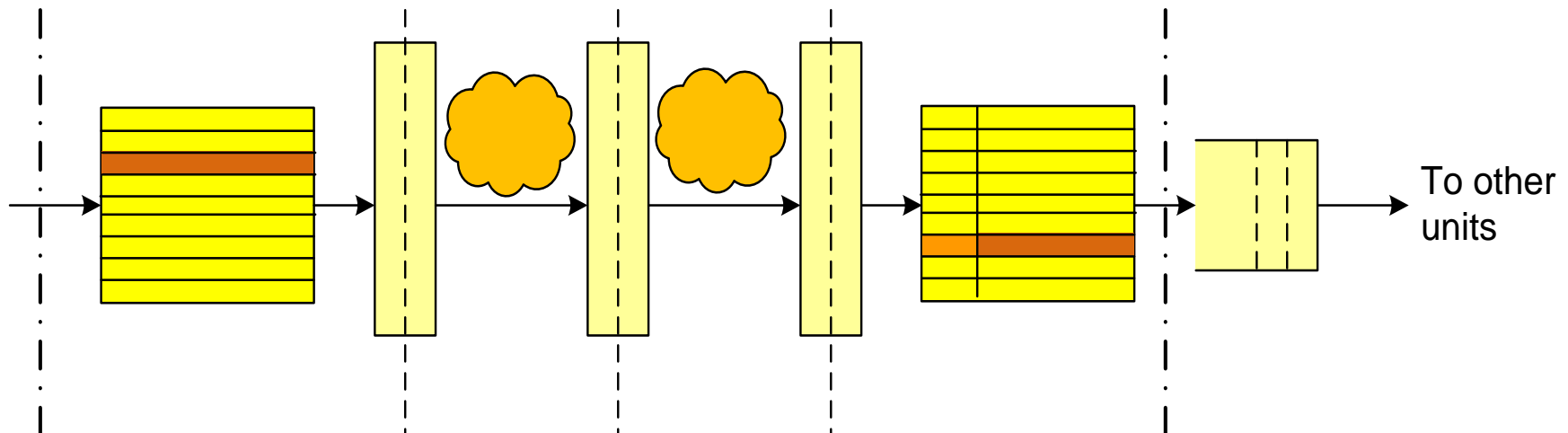
- **A set of subroutines**
 - A set of basic operations
 - Executed in a functional unit
- **Each one calls next**
 - Link subroutines to each other



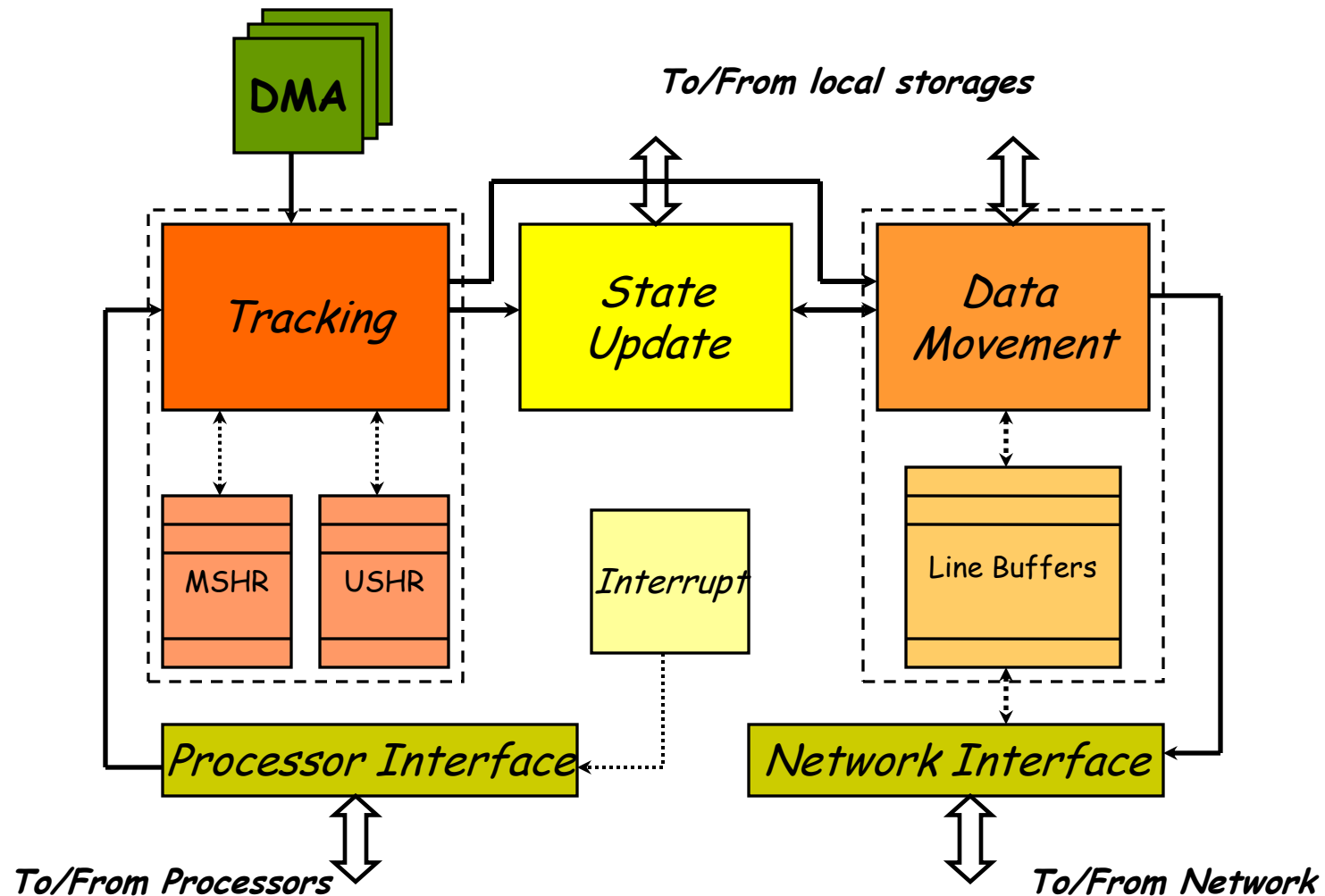


Microarchitecture

- A small pipeline
- Configuration (“program”) memories
 - Horizontal micro-code
 - Decide what to do
 - Decide how to proceed

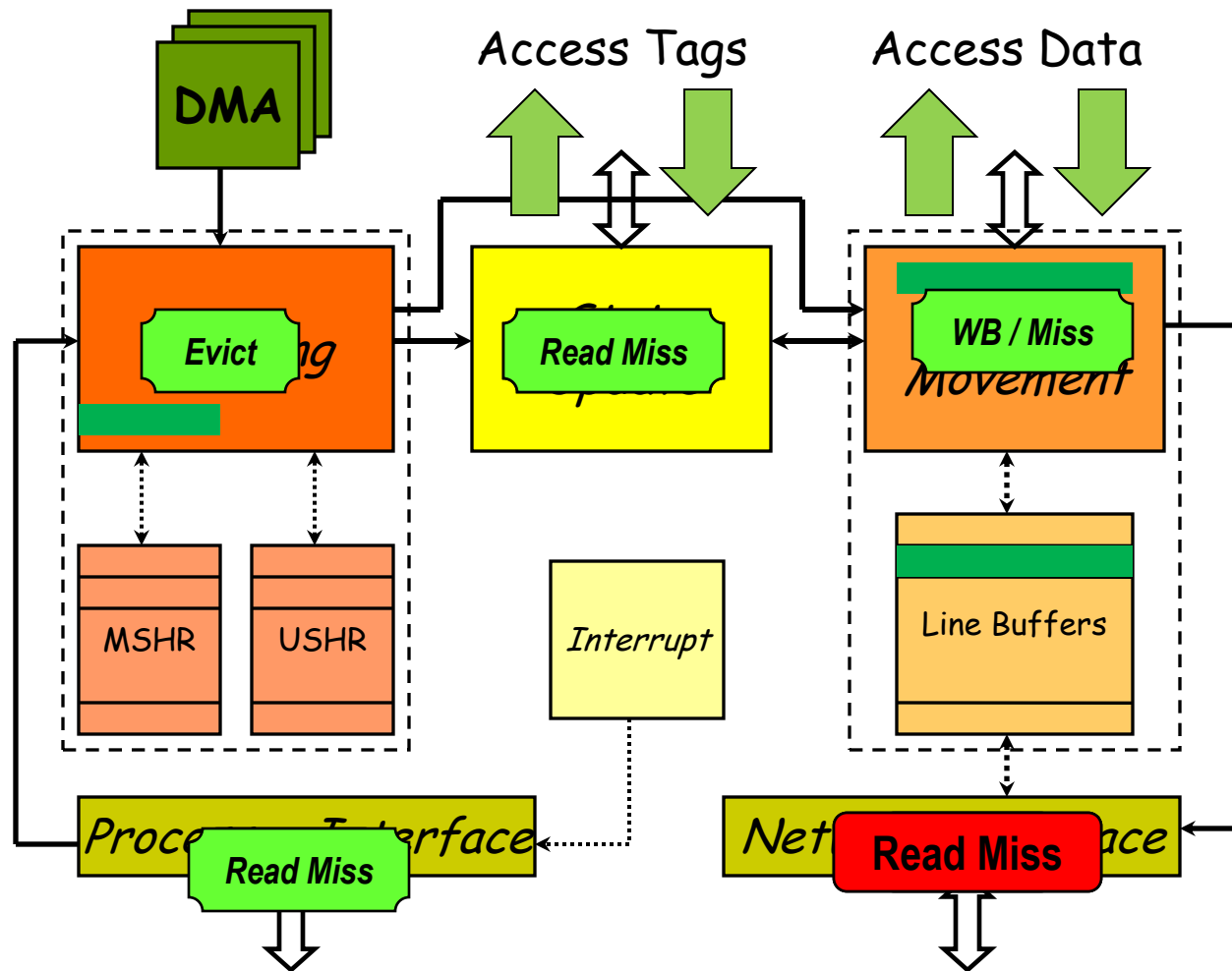


Organization





Read Miss Example





Programming Complexity

■ Cache Coherence

- Message types received by controller: 6
 - From processor: Cache miss, Upgrade miss, Prefetch
 - From network: Coherence request, Refill, Upgrade
- Subroutine types in Tracking unit: 11

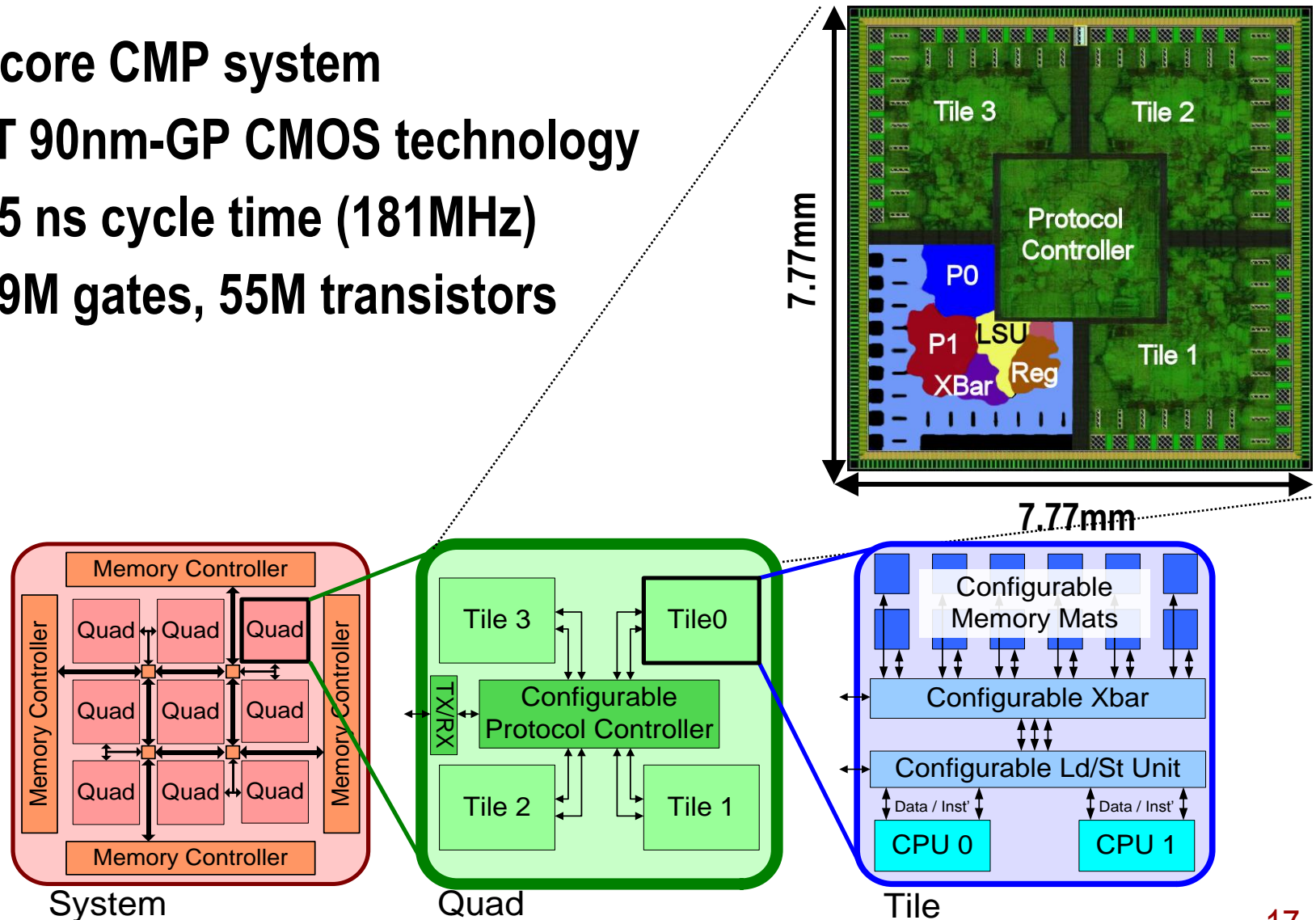
■ Streaming

- Message types: 5
 - Direct access, Gather, Scatter, Gather reply, Scatter ack.
- Subroutine types in Tracking unit: 9



Smart Memories

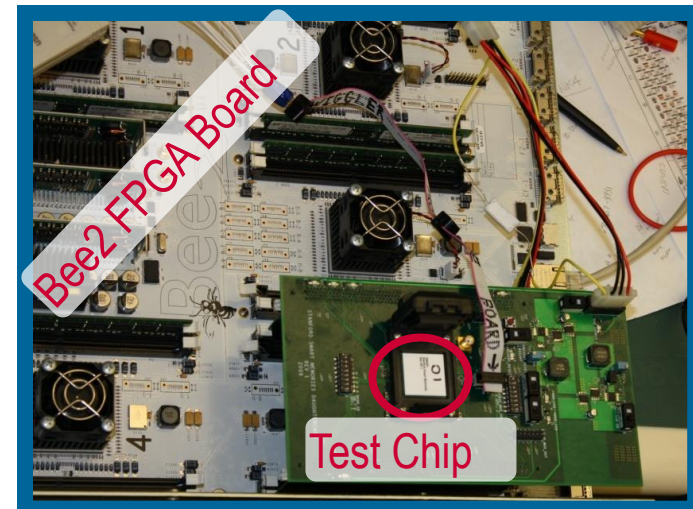
- 8-core CMP system
- ST 90nm-GP CMOS technology
- 5.5 ns cycle time (181MHz)
- 2.9M gates, 55M transistors



Status

- System bring-up.....☒
- System configuration.....☒
- JTAG tests.....☒
- Coherent shared memory tests..☒
- Transactional tests (TCC).....☒
- Streaming tests.....☒

- More testing in progress
- Planning for a 32-processor system





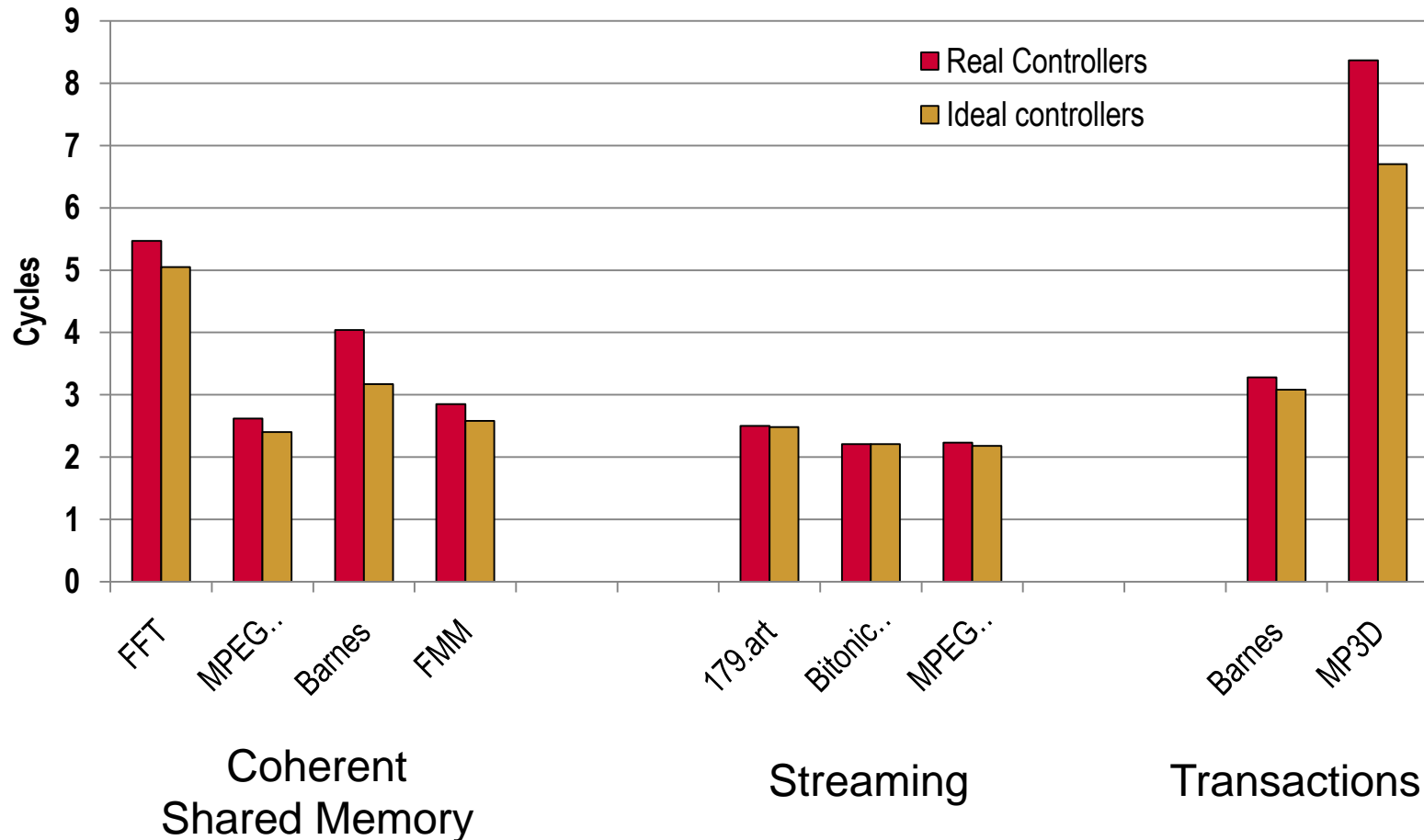
Evaluation

- **Comparison with a hardwired controller**
 - But which one? You would claim I am cheating!
- **Compare with an “ideal” controller**
 - Assume controller actions occur in zero time
 - Account for external actions
 - Data read/write
 - Message send/receive
- **Gives an upper bound**



Average Read Latency

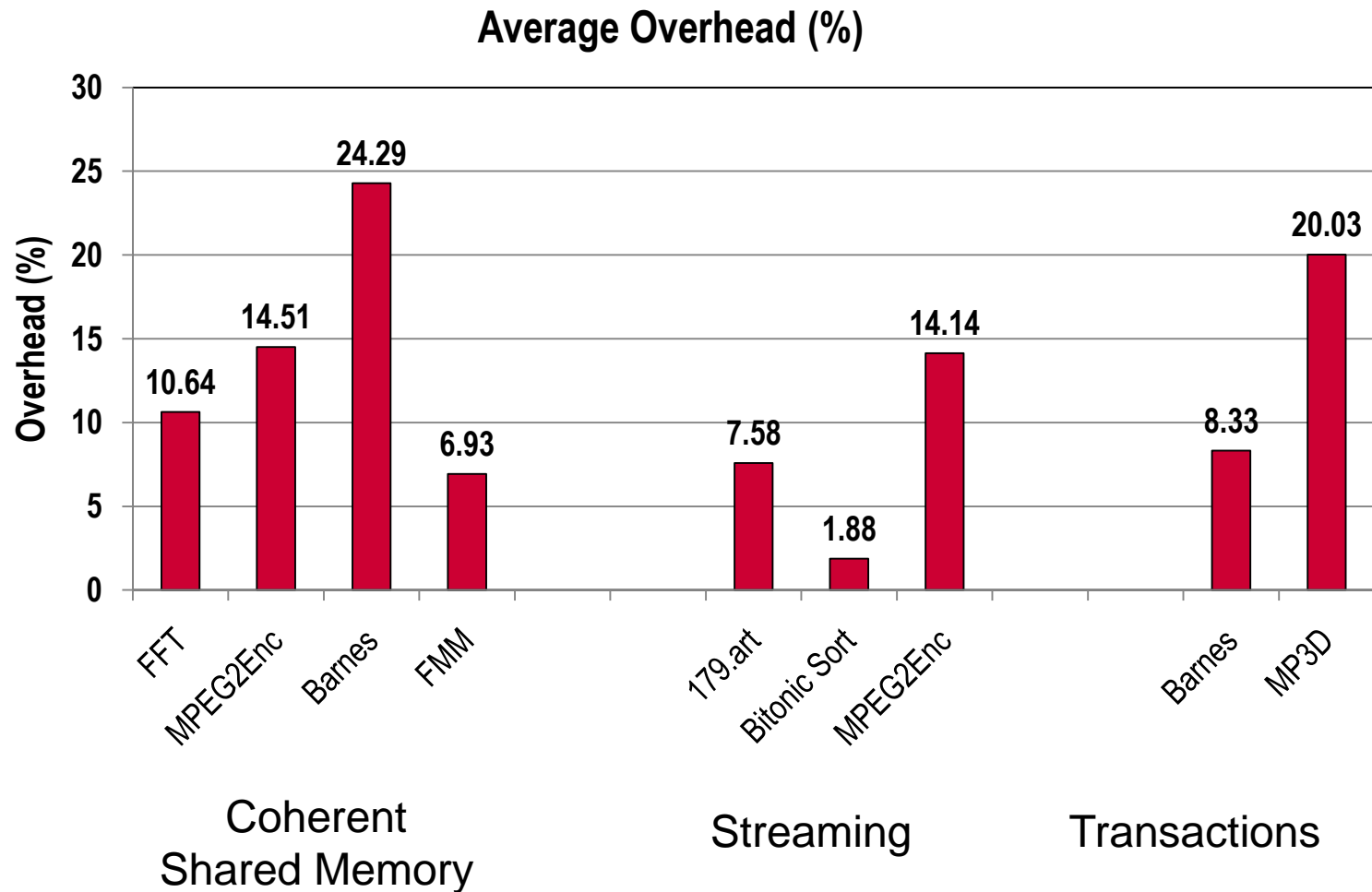
Average Read Latency - 32 processor system





Execution Time

■ Total average overhead: 15%





Conclusion

- **Strong similarity between memory systems**
 - Common resources and operations
- **A framework for memory systems design**
 - Generate specific “instances”
- **Modest performance overhead**
 - Compared to ideal systems