

Optimizing Memory Bandwidth in Systems-on-Chip

Krishnan Srinivasan
Staff Architect – Sonics, Inc.

6th ANNUAL
multicore AND TECHNICAL
CONFERENCE
EXPO

Learn today. Design tomorrow.



Silicon Valley • May 2 - 5, 2011
McEnery Convention Center • San Jose

Evolution of Digital Entertainment SoCs

Driving SoC Complexity and Memory Bandwidth

- Relentless push for higher quality user experience – at minimum system cost!
- Feature convergence – Video, Voice, Data, Audio (in every consumer device!)



6th ANNUAL
multicore AND TECHNICAL CONFERENCE EXPO

Learn today. Design tomorrow.

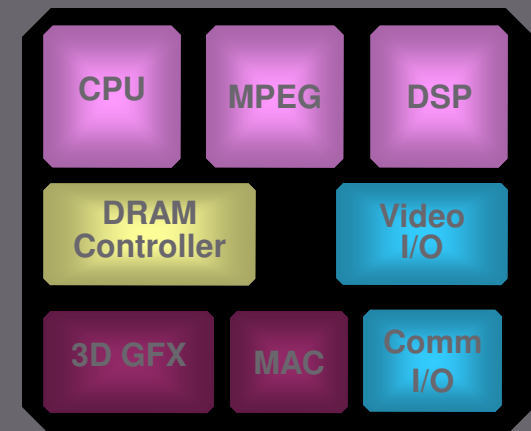


Silicon Valley • May 2 - 5, 2011
McEnery Convention Center • San Jose

Memory Subsystem Challenges

Massive feature integration: Driving SoC memory subsystem complexity to the extreme

- Multiple processors
 - CPU processor
 - DSP processor
 - Graphic processor
- Many high-performance engines
 - Video cores
 - DMA engines
- All traffic goes to DRAM



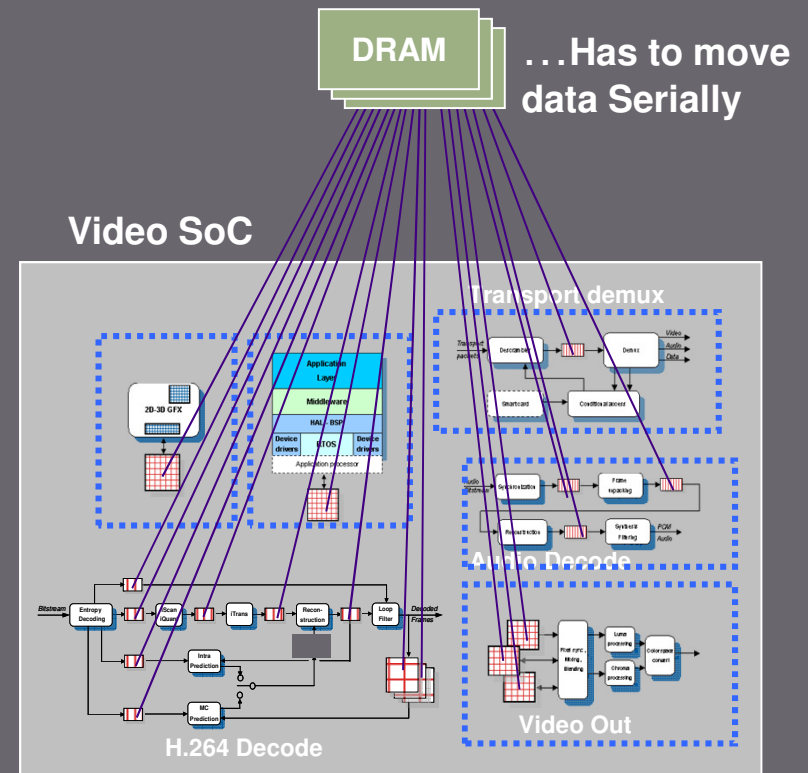
System On Chip

Distributed Heterogeneous Architectures

The Ubiquitous Memory Bottleneck

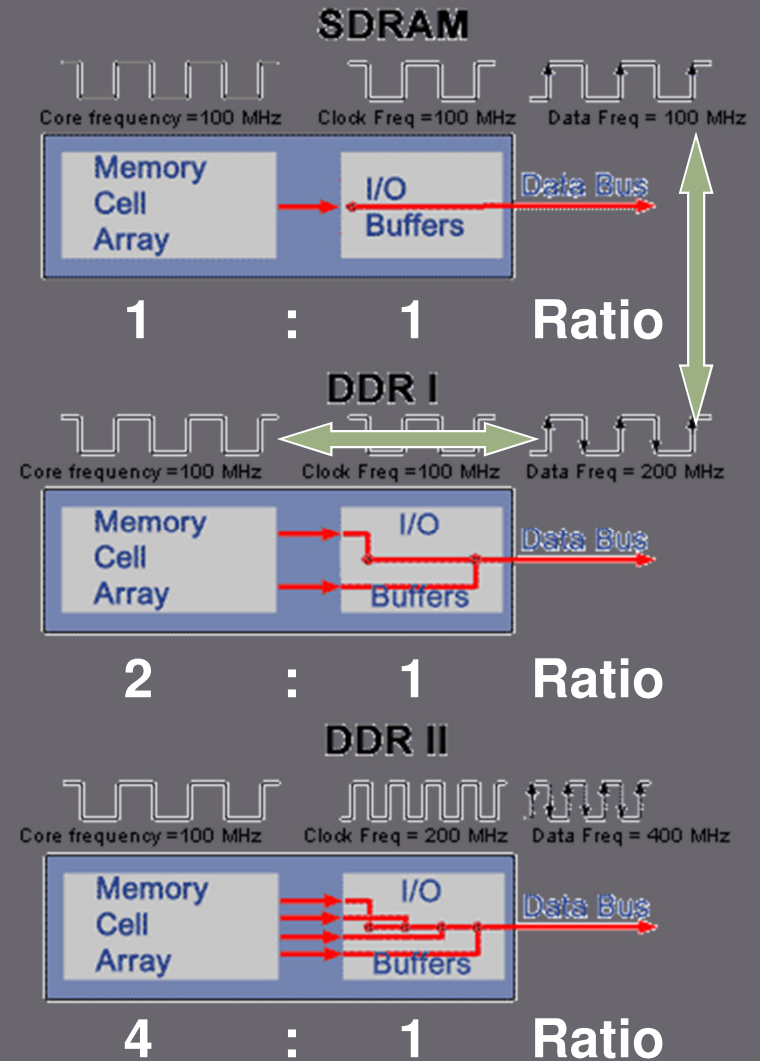
Consumer SoCs process data in parallel, but communicate...

- Limited bandwidth at the DRAM continues to be one of the unresolved challenges
- Competition among cores for DRAM degrades system performance, increasing cost and power consumption
- Advanced memory scheduling is one of today's most complex issues in advanced SoC design



Quick DRAM Refresher

- DRAM organized as independent *banks*
- Each bank composed of a number of *words*, each identified using a *row* and *column* addresses
- A row access loads an entire *page* of words into latches at the edge of the array; row accesses are slow
- A column access muxes out one word from the page latches; column accesses are fast!
- A *page miss* requires *pre-charging* the array to prepare for next row access
- DDR technology increases BW/pin by:
 - Sending data on both edges of the clock
 - Running interface at multiple of core freq.
 - DDR1: 2x; DDR2: 4x; DDR3: 8x
 - Core word is same multiple of interface word
 - Minimum DRAM burst length covers this
 - **DDR3 minimum burst length is 8**



Memory Optimization Advantages

- Satisfying a certain bandwidth at a lower frequency and lower data width has direct advantages
 - Lower operating frequency → Lower dynamic power consumption
 - Lower gate count → Lower static power consumption
 - Fewer DRAM parts → Lower cost
- Satisfying a certain bandwidth at a lower frequency and lower data width has indirect advantages
 - Lower worst case jitter → Smaller buffers at the processing engines
- Satisfying a certain bandwidth may be a system requirement

Top Down Memory Efficiency Approach

System-level and Unit-level decisions affect performance

- System-level decisions
 - DDR type (DDR-2, DDR-3, LPDDR2, Wide-I/O DRAM)
 - DDR Width
 - Number of memory channels

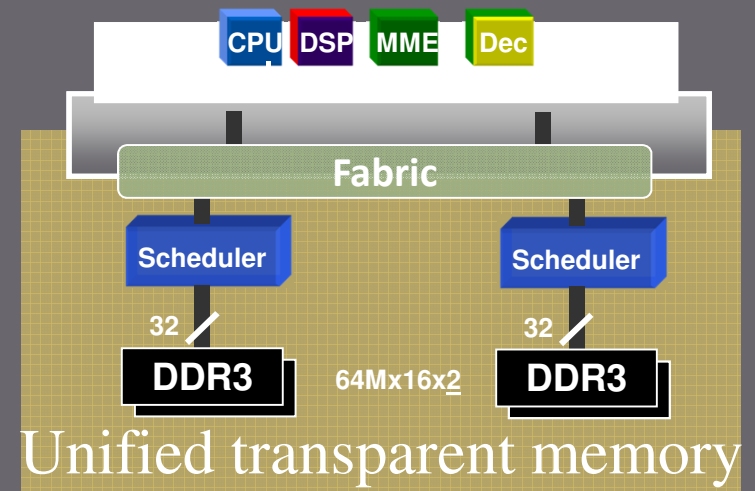
- Unit-level Decisions
 - Interconnect arbitration
 - Memory scheduler/controller arbitration
 - Interaction between interconnect and memory subsystem

System-Level Memory Considerations

- How many concurrent DRAMs are needed?
 - How should traffic to multiple DRAMs be partitioned?
- Remember, DDR-3 minimum burst length is 8
 - Assume 32 byte accesses
 - Examples are cache miss, a row of narrow macroblock
 - Assume DDR-3 interface = 8 bytes
 - Driven by bandwidth requirements ($BW = \text{interface width} \times \text{frequency}$)
 - Minimum access granularity is 64 bytes
 - DDR memory bandwidth for such accesses are reduced by 50%!!!
 - **Wide memory interfaces (more memory) do not result in better efficiency**
- Multi-Channel memories can solve the problem!
 - Two memory channels each with a 4-byte interface
 - Better concurrency, no idle cycles during accesses
 - However...proper load balancing is critical

Multi-Channel Memory Considerations

- Ordering requirements
 - Protocol/Consistency models have ordering requirements
 - Where should the traffic be split?
 - Is using re-order buffers practical?
- Load Balancing
 - Improves memory efficiency
 - Over a time window, 100% traffic on one memory limits total BW to 50%
 - However, 50% traffic on each memory sets total BW to 100%
 - Important to choose the time window carefully
 - Too fine - will cause ordering overhead to dominate
 - Too coarse - will affect efficiency
- How to balance the load?
 - Software driven
 - Hardware driven (subsystem does the load balancing)
 - Support memory interleaving between the DRAM channels
 - Memory subsystem is transparent to the initiators



TSV-based DRAM Systems

- Problems in current systems
 - In consumer & mobile SoCs, DRAM *bandwidth* needs grow faster than *capacity*
 - Scaling DRAM bandwidth requires *extra DRAMs*
 - And power-hungry PHYs
 - Wider DRAM interfaces & deeper pipelining increases access *granularity*, driving need for *multi-channel* approaches
 - Causes extra pin costs (after 2 channels)
 - Current DRAM interfaces are a *bottleneck* between:
 - Lots of parallel initiators (data clients)
 - Lots of parallel DRAM banks (data servers)
- Removing the interface bottleneck
 - Be limited instead by DRAM bank bandwidths
 - Without needing fancy PHYs

Introducing Wide I/O

- A new JEDEC DRAM standard (in process)
- Based on Through Silicon Via (TSV) technology
 - Multi-die stacking through via
- Each DRAM looks like 4 channels of LPDDR2-800 x32
 - 4 banks/channel
 - 128b @ 200MHz
 - ~1200 connections!
 - Single Data Rate
- Why TSV?
 - Much lower power than traditional solutions
 - Extremely high peak bandwidth
 - Suitable for power-sensitive mobile systems

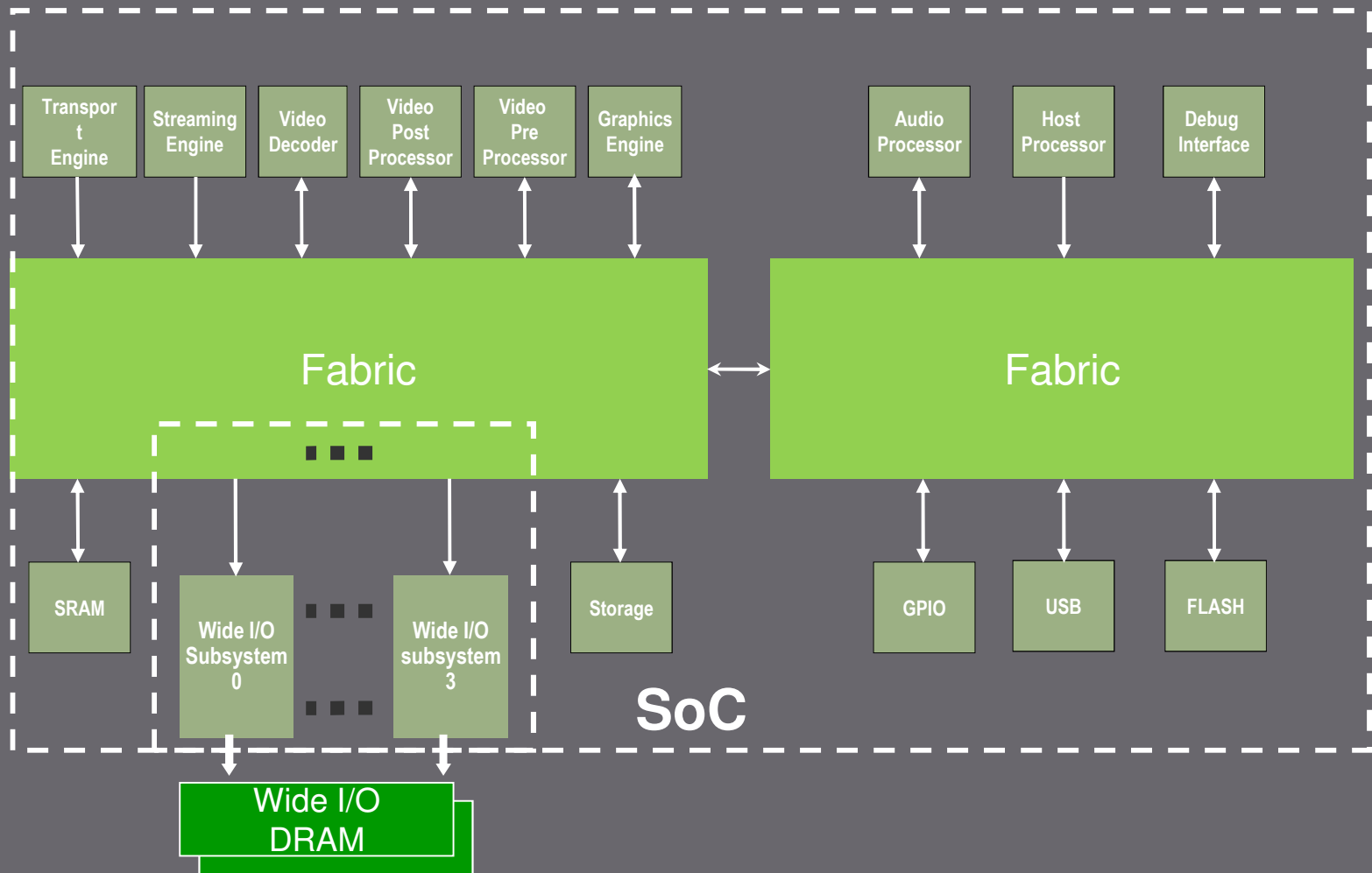
6th ANNUAL
multicore AND TECHNICAL CONFERENCE
EXPO

Learn today. Design tomorrow.

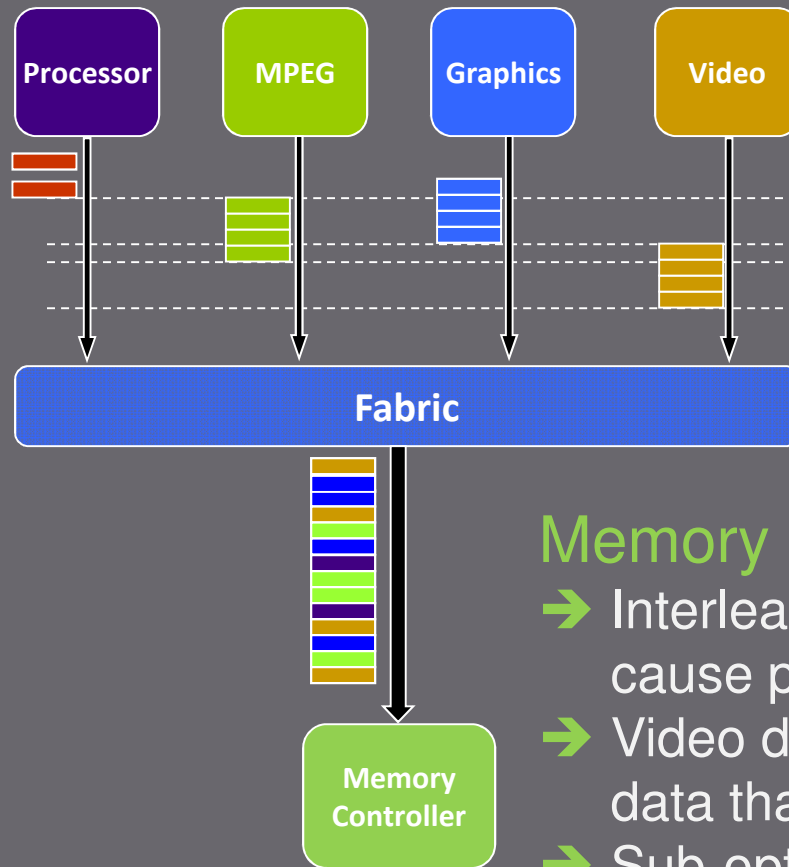


Silicon Valley • May 2 - 5, 2011
McEnergy Convention Center • San Jose

Concept: Complete Wide I/O Solution



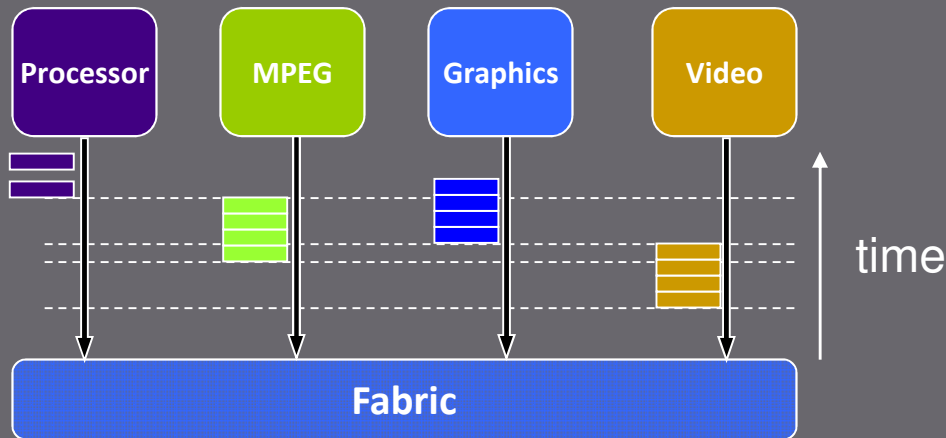
Unit Level Memory Optimization



Memory agnostic arbitration in fabric:

- Interleaving of traffic at the fabric may cause page thrashing in the DRAM
- Video decoder requires macro-blocks of data that is inherently memory unfriendly
- Sub-optimal QoS-memory efficiency tradeoff

Importance of Efficient Scheduling



Scheduler **re-orders** high priority CPU request to minimize latency

Schedules transactions to maintain highest DRAM efficiency

Breaks long system bursts into smaller chunks for optimal QoS/efficiency trade-off

Why is QoS Important?

- Poor QoS leads to over-dimensioning of the system
 - Buffers are larger than necessary
 - Data paths may be wider than necessary
 - Result is wasted gates
- Poor CPU latency can severely compromise performance

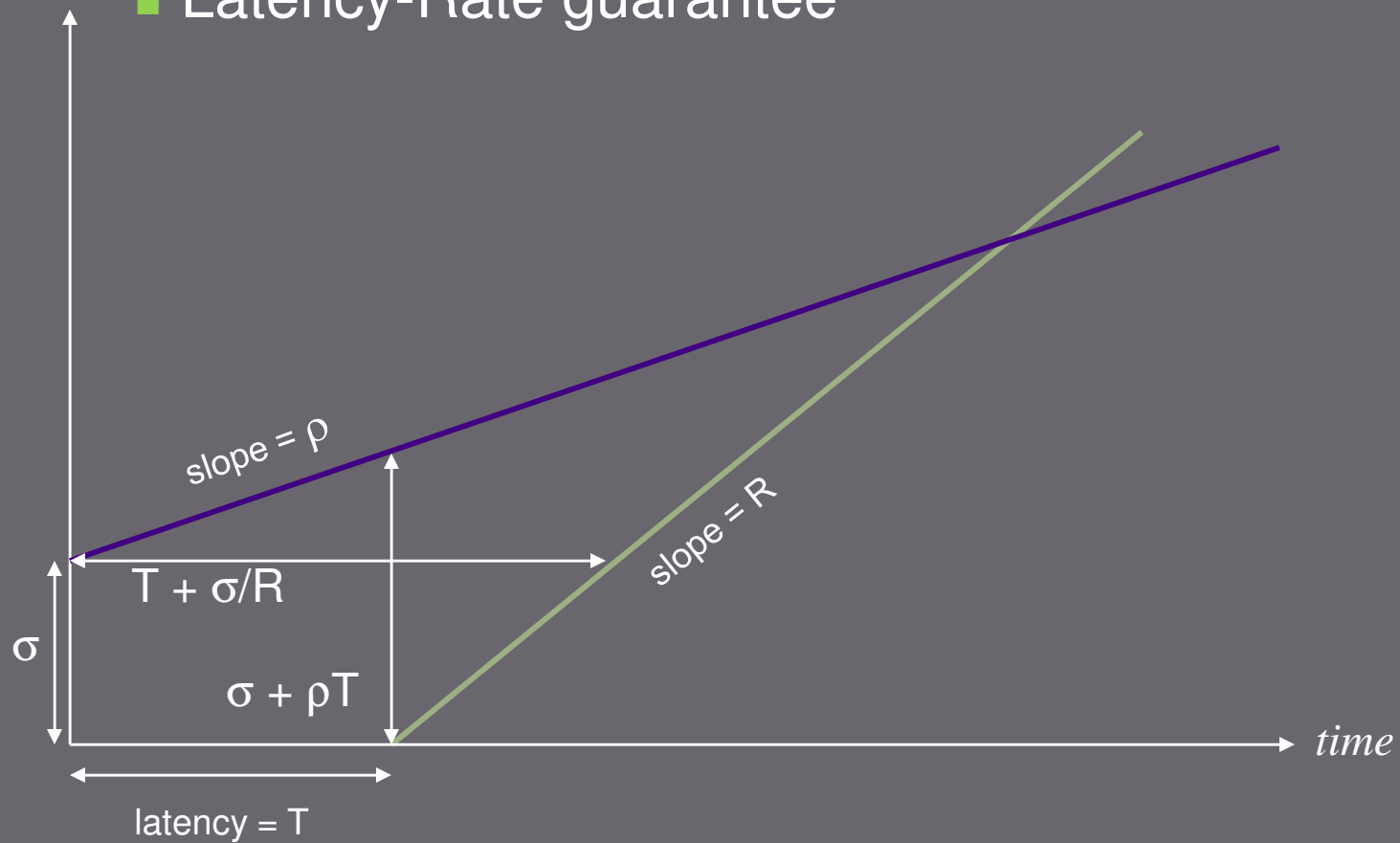
DRAM Latency	CPU Average Access Time	Effective Performance
σ^{\dagger}	1.0 cycles	100 %
20	1.4 cycles	71 %
30	1.6 cycles	63 %
40	1.8 cycles	55 %
50	2.0 cycles	50 %
60	2.2 cycles	45 %

QoS Requirements

1. Ensure no hard real time (HRT) traffic ever misses a deadline
 - Completing earlier is OK but not necessary
 - Completing later than deadline is unacceptable
2. Prioritize low-latency (LL) traffic
 - To minimize latency for CPU cache line fills
 - But do not violate HRT deadlines
3. Avoid starving any resource indefinitely
 - Even best effort (BE) traffic must be serviced eventually
4. When all of the above goals are met, make choices that maximize throughput

Servicing HRT Traffic

- Latency-Rate guarantee

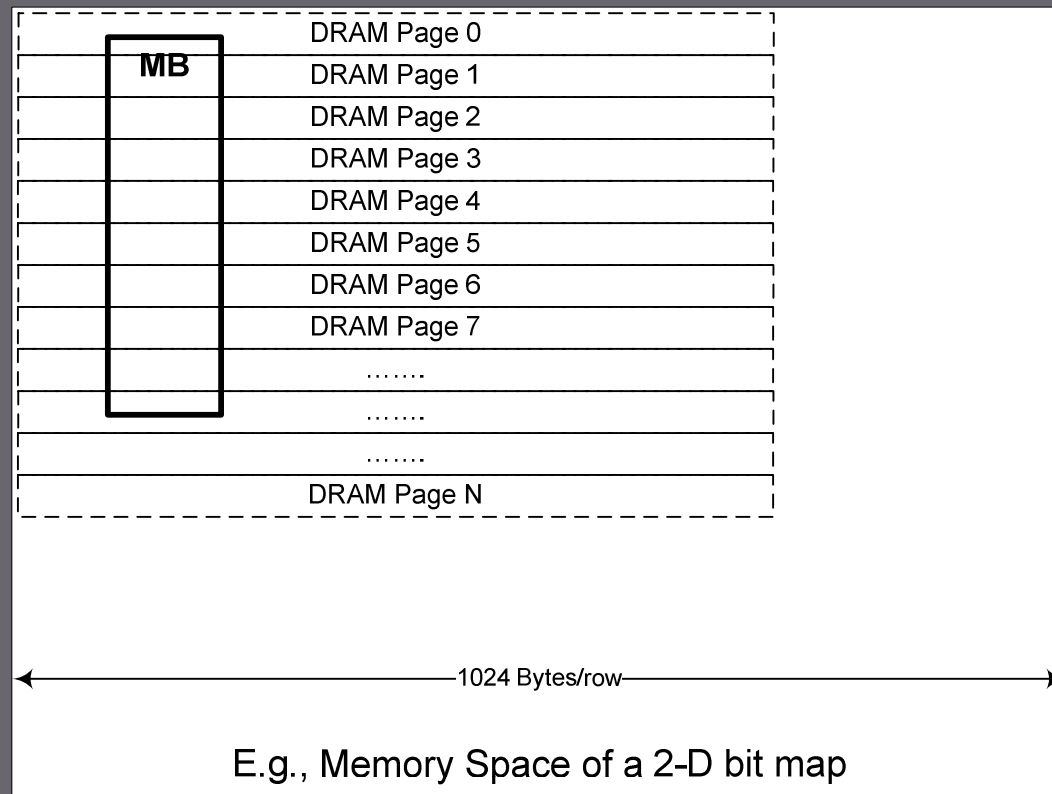


Improving Memory Bandwidth

- Minimize direction change for consecutive DRAM accesses
 - Under the same QoS level prefer accesses that minimize turnarounds
 - RD RD RD RD WR WR WR WR **versus**
 - RD WR RD WR RD WR RD WR
- Minimize Page Misses
 - Close pages predictively hide the latency
 - Prefer requests to pages that are already open
 - Interleave requests to hide the page miss penalty

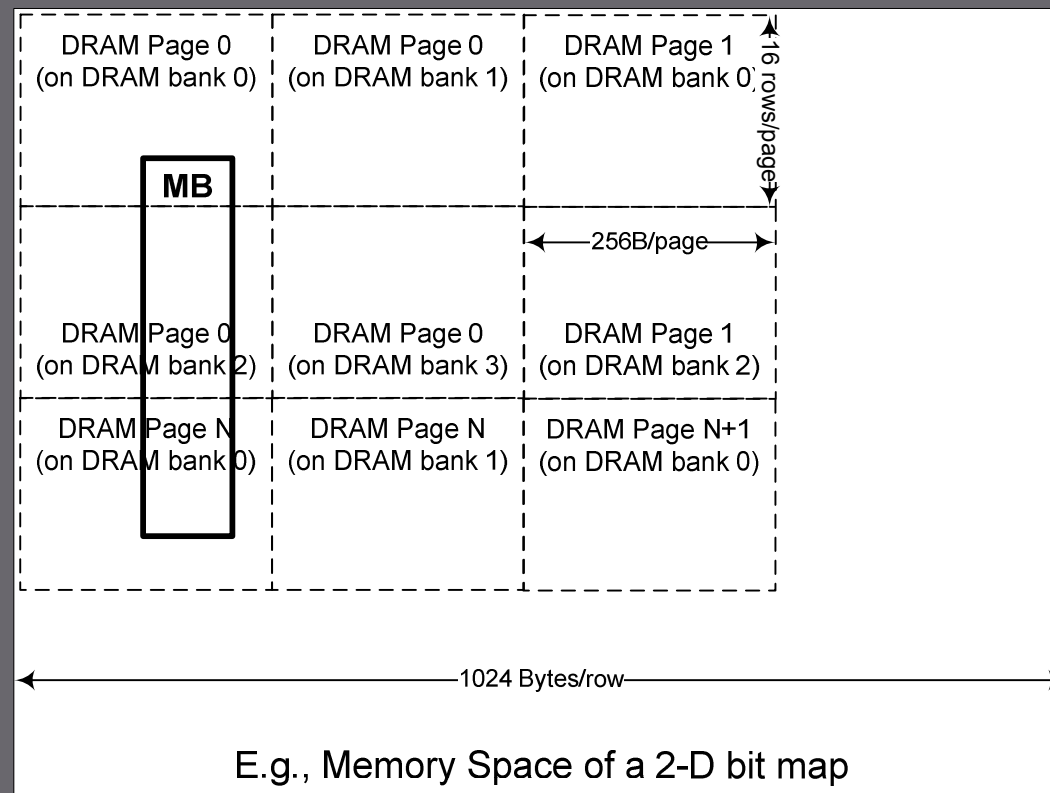
Video Decoder Example

- 4-bank configuration, 4KB pages (1024x32b), 1024B/row
- DRAM address organization in un-tiled space: 1 row per page, page size = 4KB
- $\langle \text{row}[r:0] \rangle \langle \text{bank}[1:0] \rangle \langle \text{col}[9:0] \rangle$
- Numerous page misses



Video Decoder (cont'd)

- 4-bank configuration, 4KB pages (1024x32b), 1024B/row
- DRAM address organization in tiled space:
<row[r:2]><bank[1]><col[9:6]><row[1:0]><bank[0]><col[5:0]>
- Much fewer page misses



Summary

- High-Performance SoCs are primarily limited by memory bandwidth
- Designers are often forced to add memory to achieve bandwidth
 - Extra cost/wasted memory
- System-level approach is essential when solving memory bandwidth challenges
 - Memory controllers need to be system-aware
 - QoS, Address Tiling
 - Systems need to be memory-aware
 - Single versus multiple channels
 - Load balancing, and traffic partitioning have become more important
- TSV - Key emerging technology for mobile SoCs
 - Allows high-bandwidth with low-power characteristics
 - Similar architecture to multi-channel DRAM
 - Requires load balancing and other system-level techniques