



SONICS



High-Performance Memory Subsystems for Consumer Multicore SoCs

Drew Wingard, Sonics

- ***Multicore consumer SoC background***
- DRAM subsystem challenges
- Solution aspects
- Putting it all together

Consumer SoC Examples



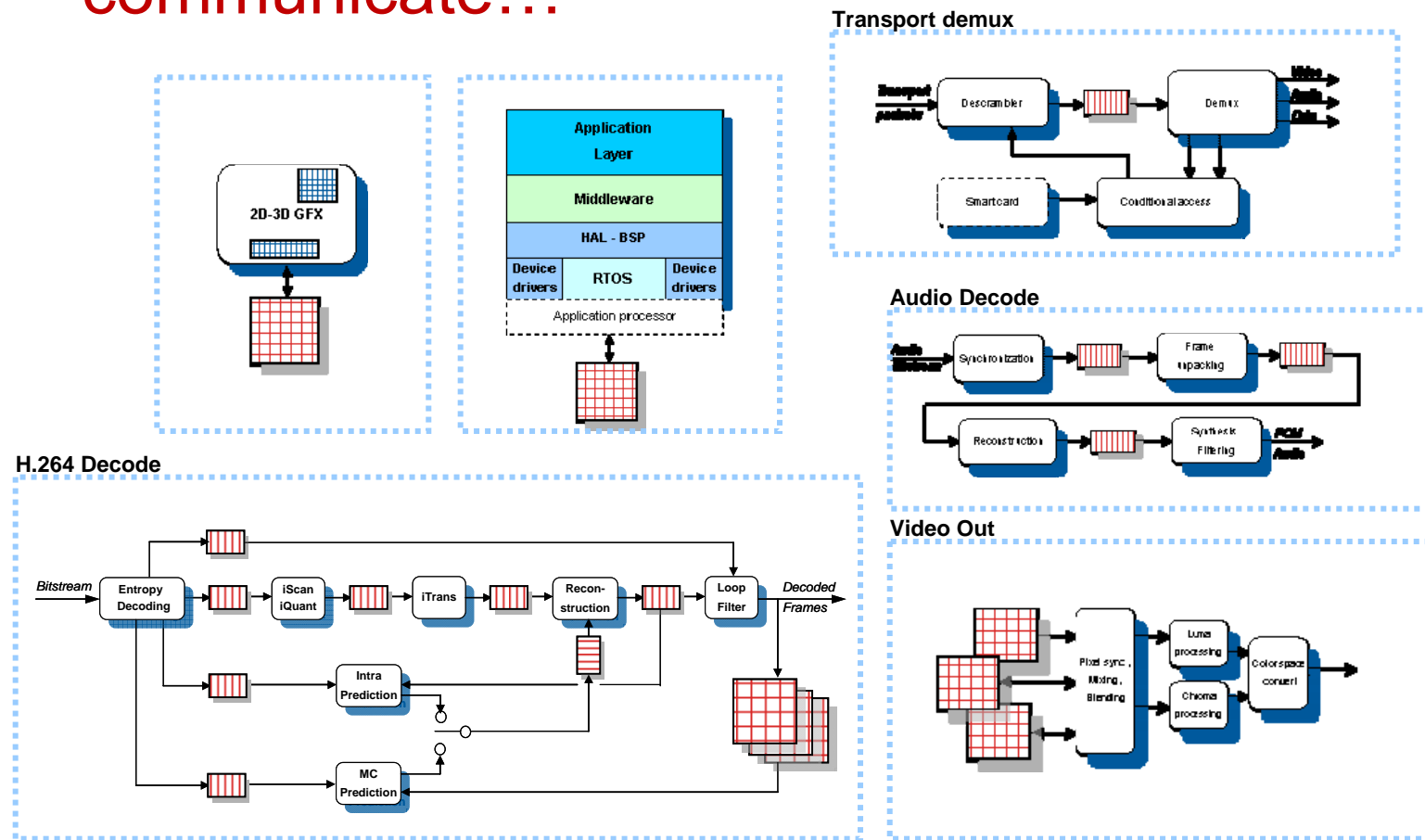
- What are some key applications for consumer SoCs?
- Key characteristic: relentless push for higher quality user experiences – at minimum system cost!



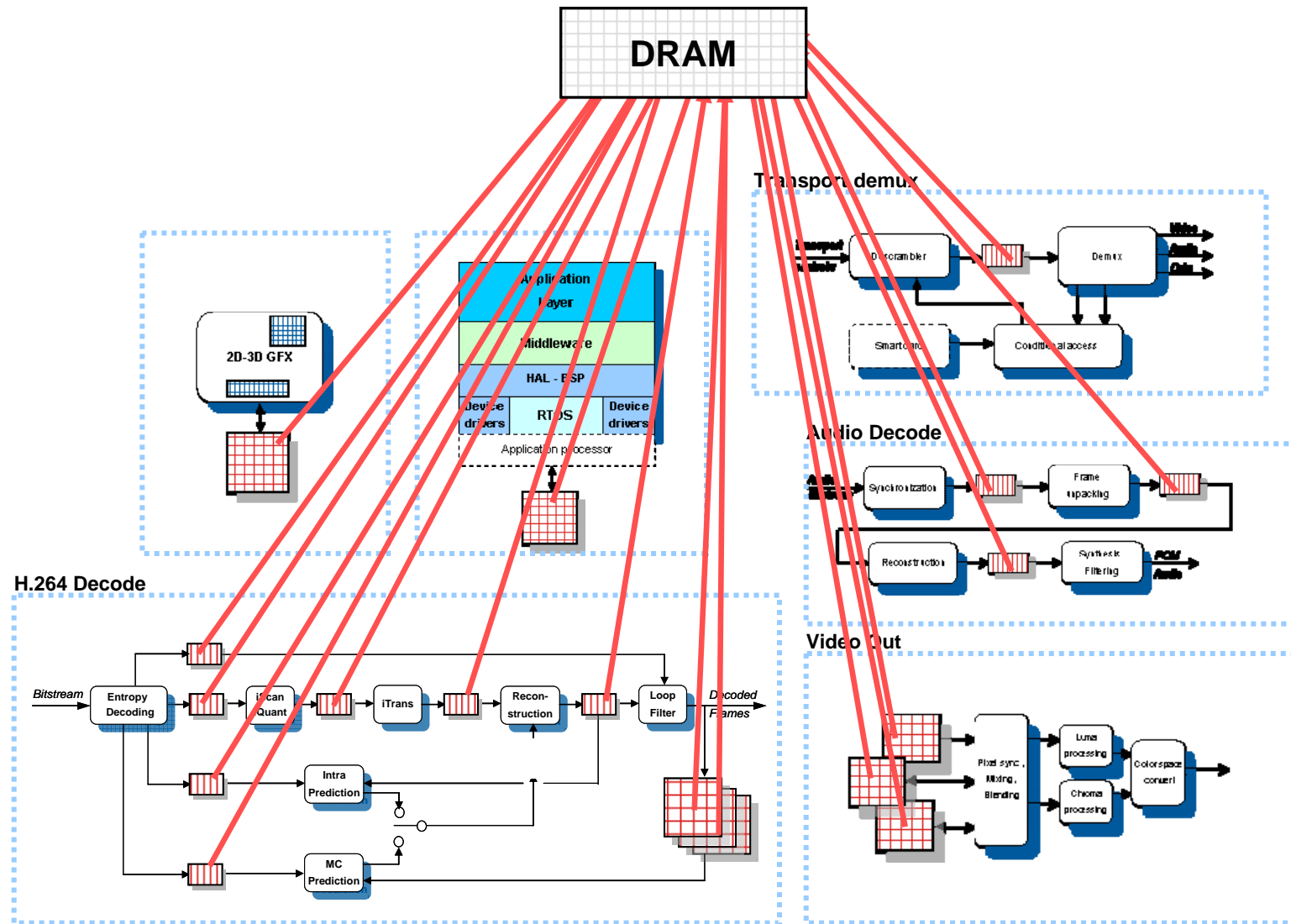
Concurrency in Consumer SoCs



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



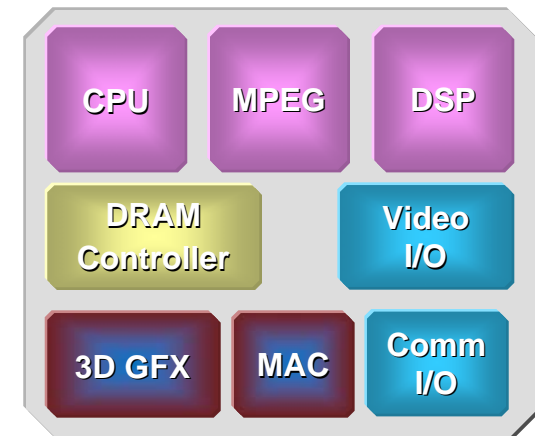
Concurrency in Consumer SoCs



- Assertion: video SoC applications have >> 50% of system traffic to/from external DRAM
- Consumer volumes and price points demand ***cheapest DRAM configurations*** that support required performance
- Implications:
 - SoC architecture is mostly a fan-in tree to external DRAM
 - Maximizing delivered DRAM ***throughput*** and ***utilization*** are key
 - Fewer DRAMs
 - Lower speed grades

- Multicore consumer SoC background
- ***DRAM subsystem challenges***
 - *Massive connectivity to DRAM*
 - *Achieving high DRAM utilization*
 - *QoS*
 - *Widely varying DRAM access styles*
 - *Increasing access granularity*
- Solution aspects
- Putting it all together

- **Massive feature integration**
 - Driven largely by Moore's Law (supply) and convergence (demand)
- **Continued movement of complexity to software**
- **Distributed architectures**
 - Higher scalability (and independence?)
- **Multiple processors**
 - (Multicore) CPU
 - DSP
 - Special purpose (MPEG, GFX, ...)
- **Distributed DMA**
 - Removes centralized DMA bottleneck
 - Simplifies driver software integration



System On Chip

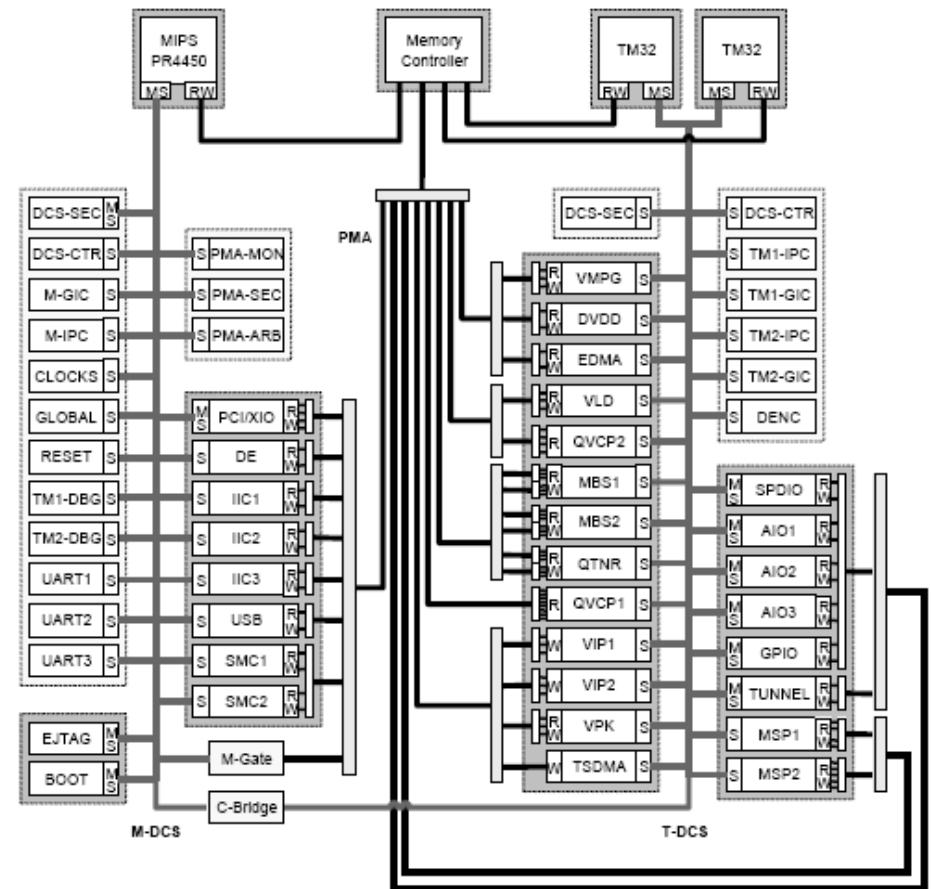
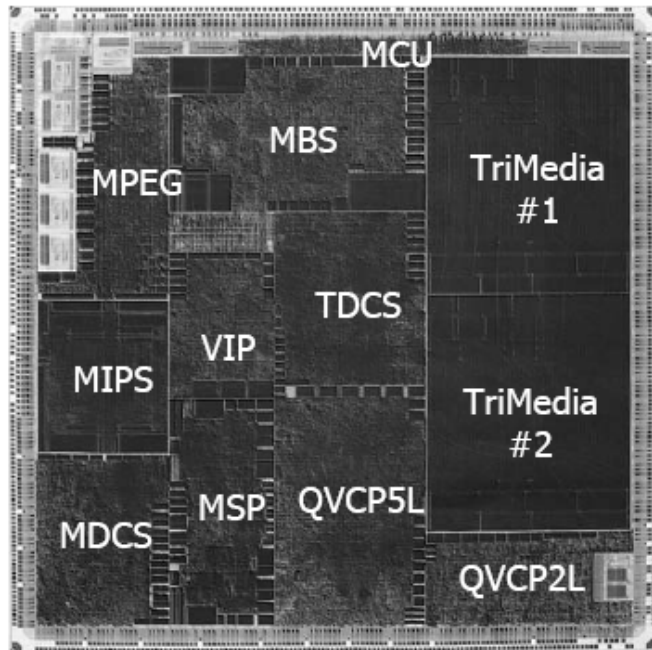
Massive Connectivity to DRAM

putting it all together



Viper2 (PNX8550)

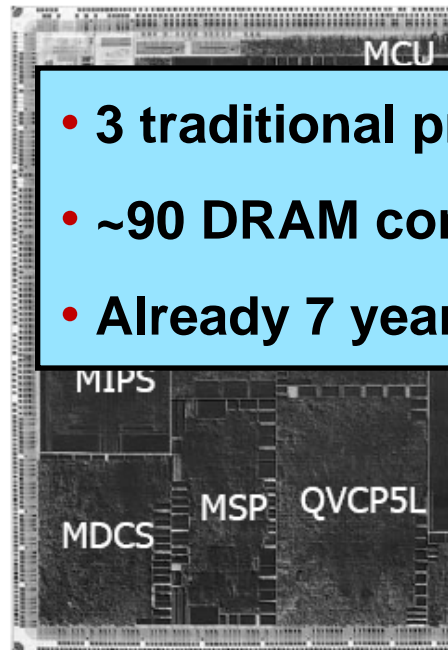
- 0.13 μm
- ~50 M transistors
- ~100 clock domains
- more than 70 IP blocks



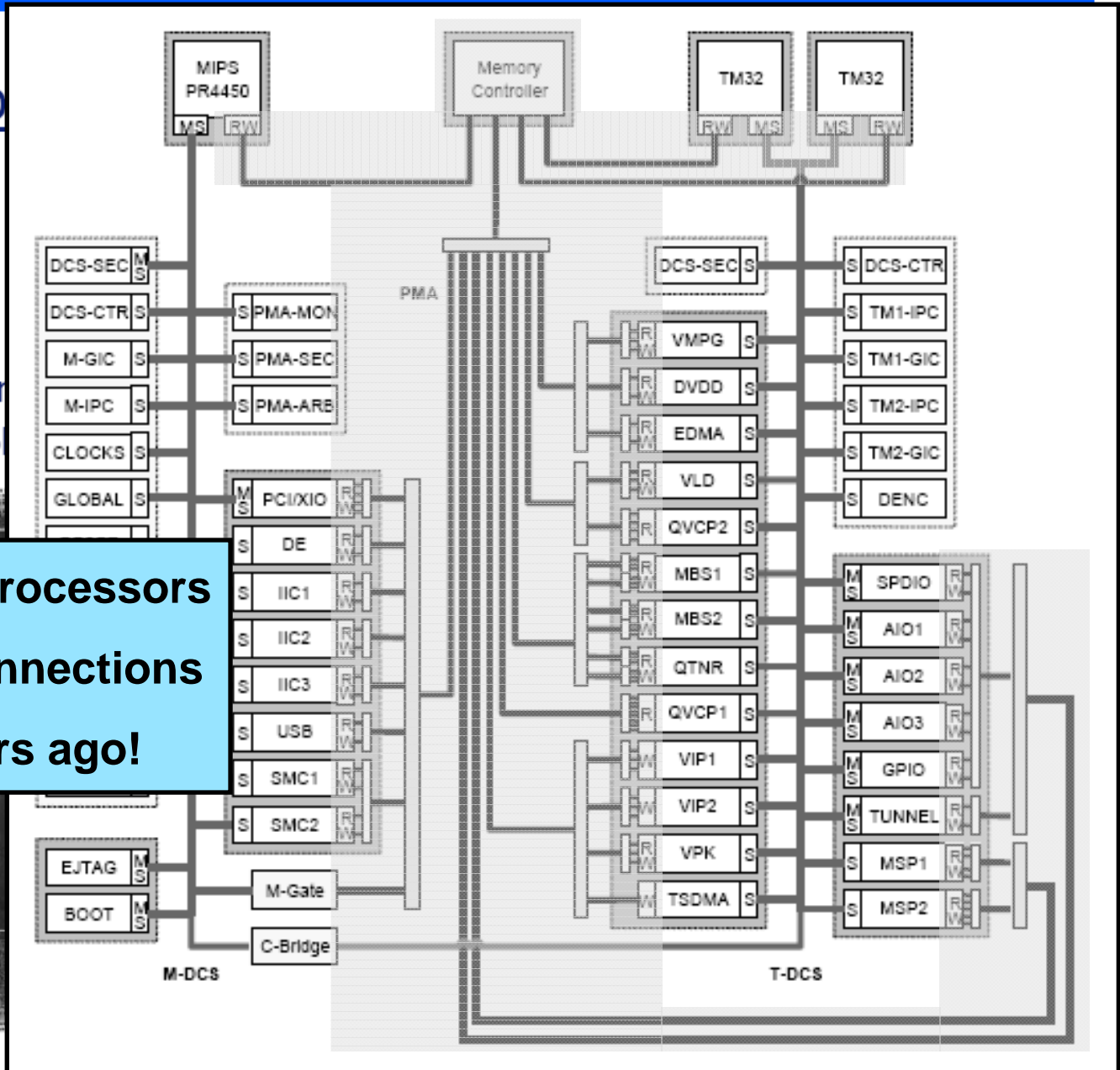
putting it all to

Viper2 (PNX8550)

- 0.13 μm
- ~50 M transistors
- ~100 clock domains
- more than 70 IP blocks



- 3 traditional processors
- ~90 DRAM connections
- Already 7 years ago!



- **Utilization Rate:** the percentage of DRAM data cycles that transfer data that is useful to the system
- **Example – at 85% utilization, 2 DDR3-1600 parts in a x32 configuration (e.g. 2 x16 DDR3 DRAMs) deliver:**
 - $(85\%) \times (1600 \text{ Mbits/sec/pin}) \times (32 \text{ pins}) / (8 \text{ bits/Byte}) = 5.44 \text{ GBytes/sec}$
- **Things which reduce DRAM utilization:**
 - Refresh cycles
 - RD/WR data bus turnaround
 - Page misses
 - Partial bursts (lengths < BL)
 - Unaligned bursts
 - Command bus conflicts (across banks)
 - QoS optimizations!
- **DRAM schedulers arbitrate among a set of system transactions to optimize DRAM utilization rates**
 - Perhaps QoS, too!

- System transactions targeting DRAM limit the peak utilization per initiator, but exploiting the parallelism between initiators allows an intelligent scheduler to optimize utilization
- Utilization-related traffic characteristics:
 - Burst lengths, address sequences and address alignment
 - Address relationships across transactions (e.g. 2D)
 - Number of outstanding transactions
 - Read/write mix
 - Ordering constraints
 - Time-domain behavior (i.e. isochronous vs. bursty/asynchronous)

How Traffic Impacts DRAM Utilization



Traffic Characteristic	Utilization Impact	Refresh cycles	RD/WR turnaround	Page misses	Partial bursts	Unaligned bursts	Command conflicts	QoS optimizations
Burst lengths/sequences/alignment			X	X	X	X	X	
Addressing across transactions			X	X			X	X
# of outstanding transactions			X	X			X	X
Read/write mix			X					
Ordering constraints			X	X			X	X
Time-domain behavior			X	X			X	X

Assumes refresh is independent from traffic

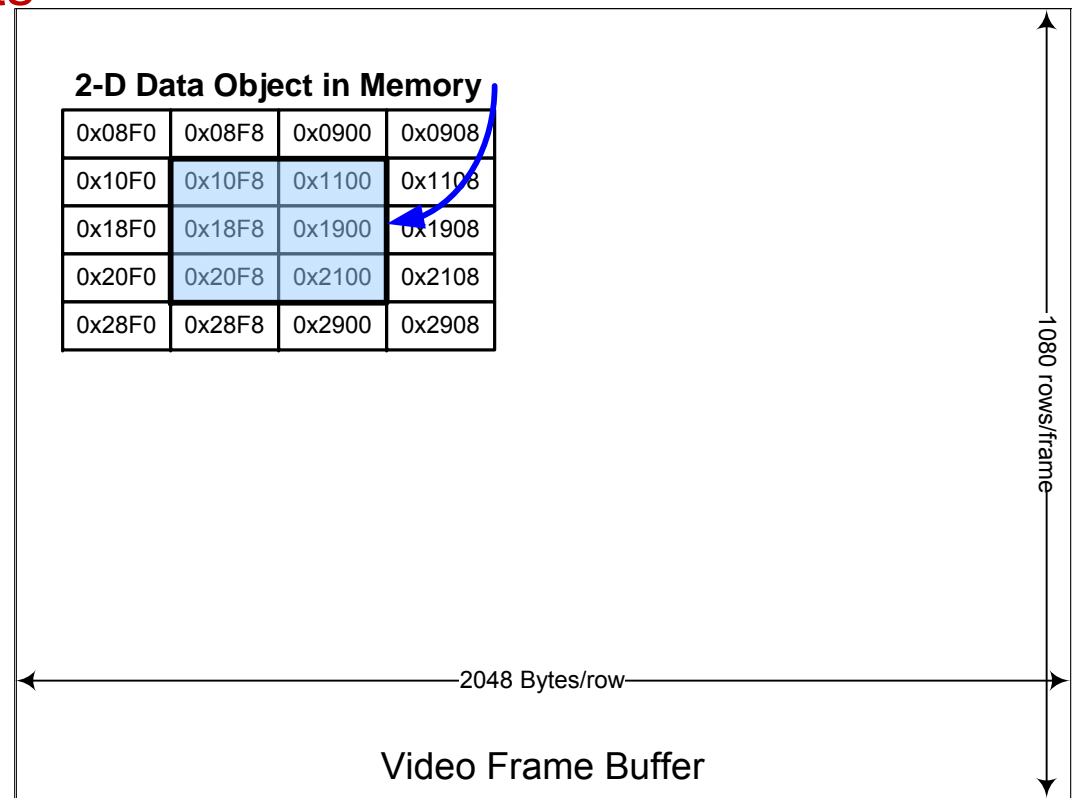
- High DRAM utilization (throughput) and Quality of Service are in conflict
 - Utilization prefers long DRAM bursts
 - DRAM operates most efficiently
 - QoS demands short DRAM bursts
 - Provide low latency service for CPUs
 - Control buffering requirements for real-time users
- Consumer SoCs use the DRAM scheduler to tune the trade-off between utilization and QoS

■ Exploiting spatial locality is key for high utilization

- CPUs tend to stay with an O/S page (multiple DRAM pages)
- Much processing and I/O DMA uses long incrementing bursts
- Image processing is tougher

■ Two-dimensional bursts

- 2D transaction using a single read or write command
- Popular for HD video and graphics



Locality Challenges: DRAM Access Styles



■ Exploiting spatial locality is key for high utilization

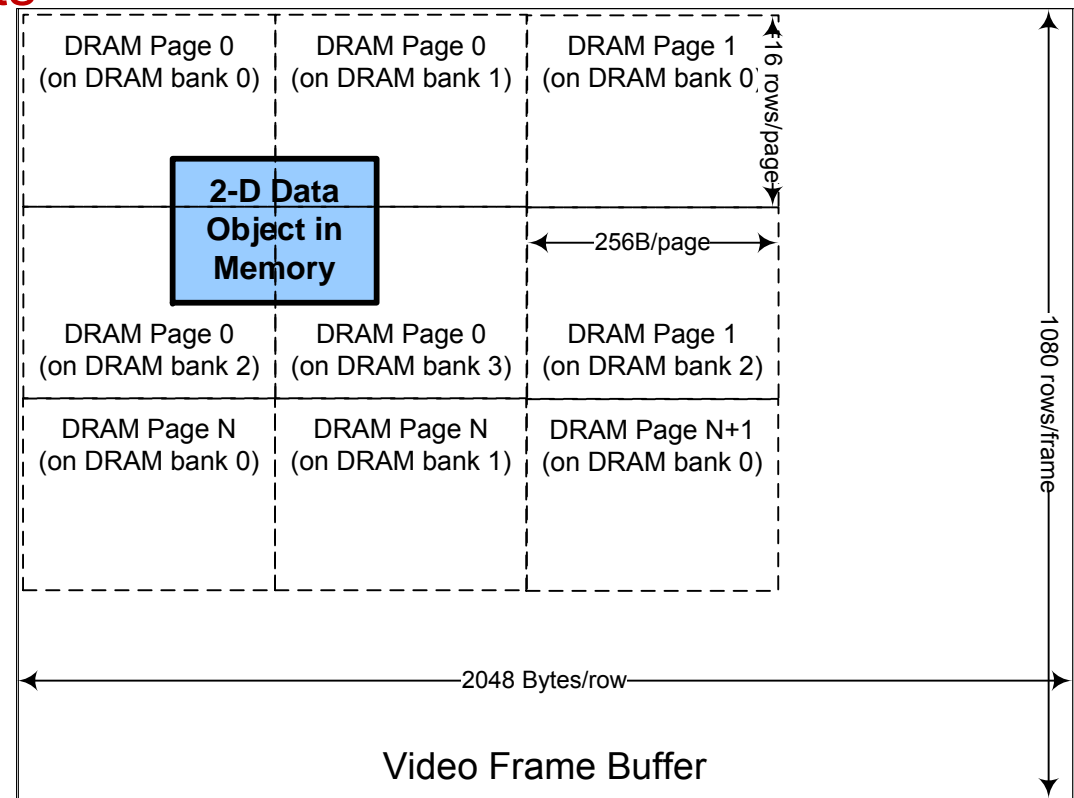
- CPUs tend to stay with an O/S page (multiple DRAM pages)
- Much processing and I/O DMA uses long incrementing bursts
- Image processing is tougher

■ Two-dimensional bursts

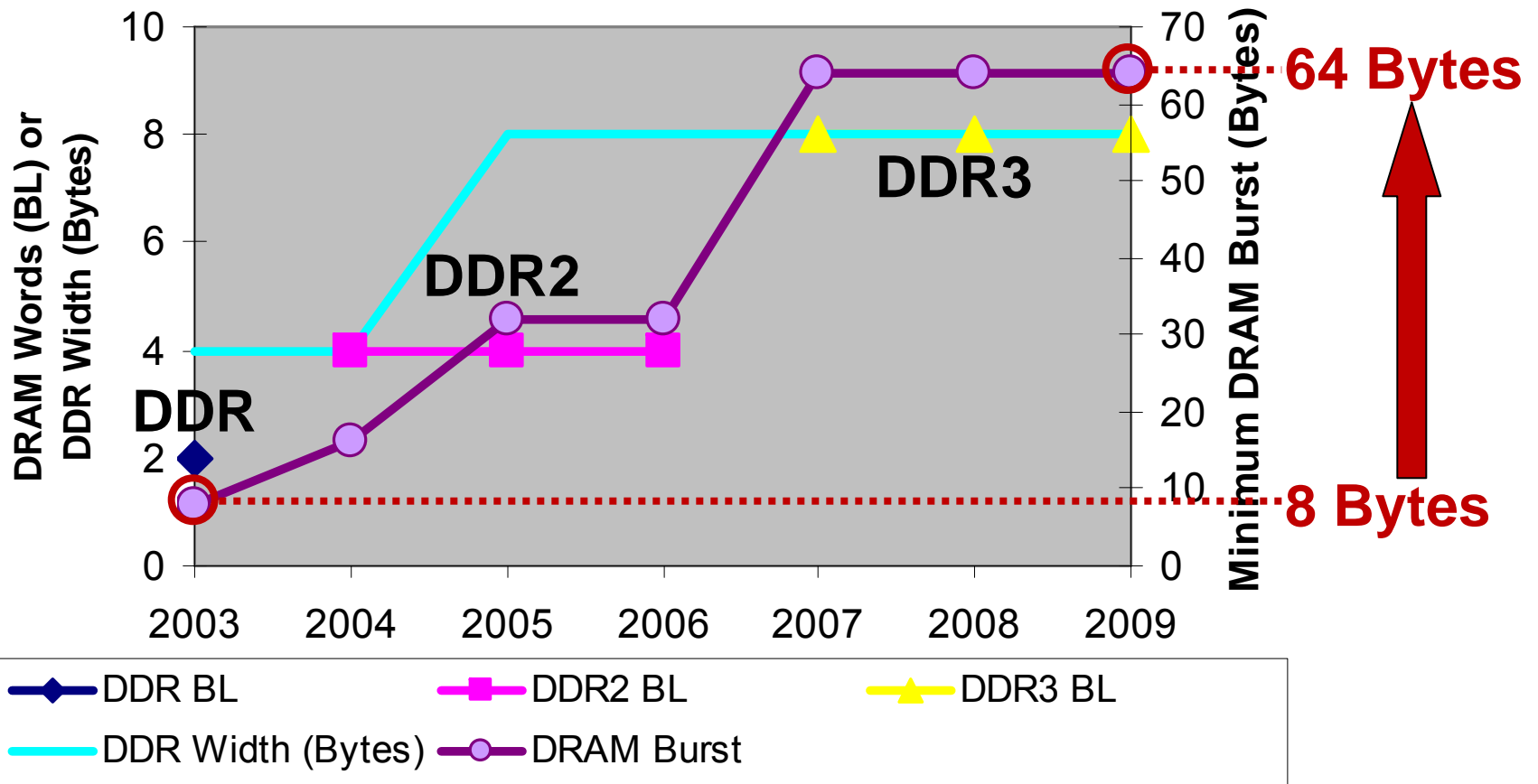
- 2D transaction using a single read or write command
- Popular for HD video and graphics

■ Address tiling

- Rearrange DRAM address organization to exploit locality
- Avoids page misses
- One size doesn't fit all!



Access Granularity: DRAM Burst Sizes Growing Too Large



Data Transfers Shorter Than Burst Size Lose Efficiency
Many SoC Data Objects \leq 32 Bytes!

Example: Analytic Traffic Characterization



Traffic Flow	Burst Length (dword)	Height	Aligned?	Best-case Transfer Efficiency			Tiled?	Page misses per DDR Burst					
				16	32	64		N	Y	N	Y	N	Y
				16	32	64	Burst (Bytes)	16	16	32	32	64	64
				8	8	8	BL	8	8	8	8	8	8
				2	4	8	Data Width (B)	2	2	4	4	8	8
Vid decode Wr	2	8	N	53%	36%	22%		53%	7%	73%	9%	89%	11%
Vid decode Rd	10	3	N	85%	74%	59%		17%	6%	30%	10%	47%	16%
Vid back end	32	1	Y	100%	100%	100%		6%	6%	13%	13%	25%	25%

Source: Customer (HDTV) System Dataflow

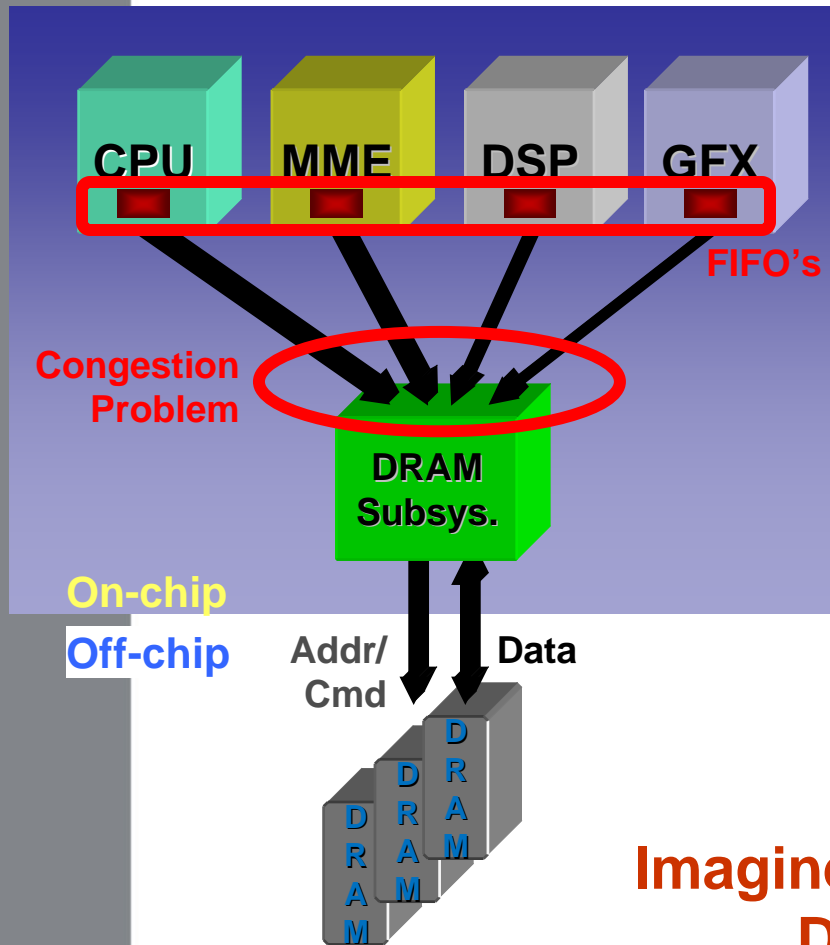
- **As DRAM burst size increases, efficiency drops substantially for short and/or unaligned traffic**
 - MPEG macro-block fetch is an easy example
 - Long burst traffic stays efficient
 - CPU traffic loses efficiency if DDR burst size > cache line size
- **2D traffic generates many page misses for row/bank/column DRAM address organization**
 - Address tiling reduces 2D page misses substantially
 - Long burst traffic is (again) tolerant
 - But, traditional CPU traffic prefers row/bank/column!

- Multicore consumer SoC background
- DRAM subsystem challenges
- ***Solution aspects***
 - *High concurrency interconnect networks*
 - *Single ported DRAM controller protocols*
 - *High utilization scheduling with QoS*
 - *Scalable multichannel*
 - *Flexible address tiling*
- Putting it all together

Star Topology Memory Subsystems



Traditional Approach



- Initiators present requests in parallel to multi-port scheduler
- FIFO's at initiators provide
 - Rate decoupling
 - Service jitter tolerance
- DRAM subsystem needs no FIFO, only pipelining
- System performance limited only by traffic & scheduler

But:

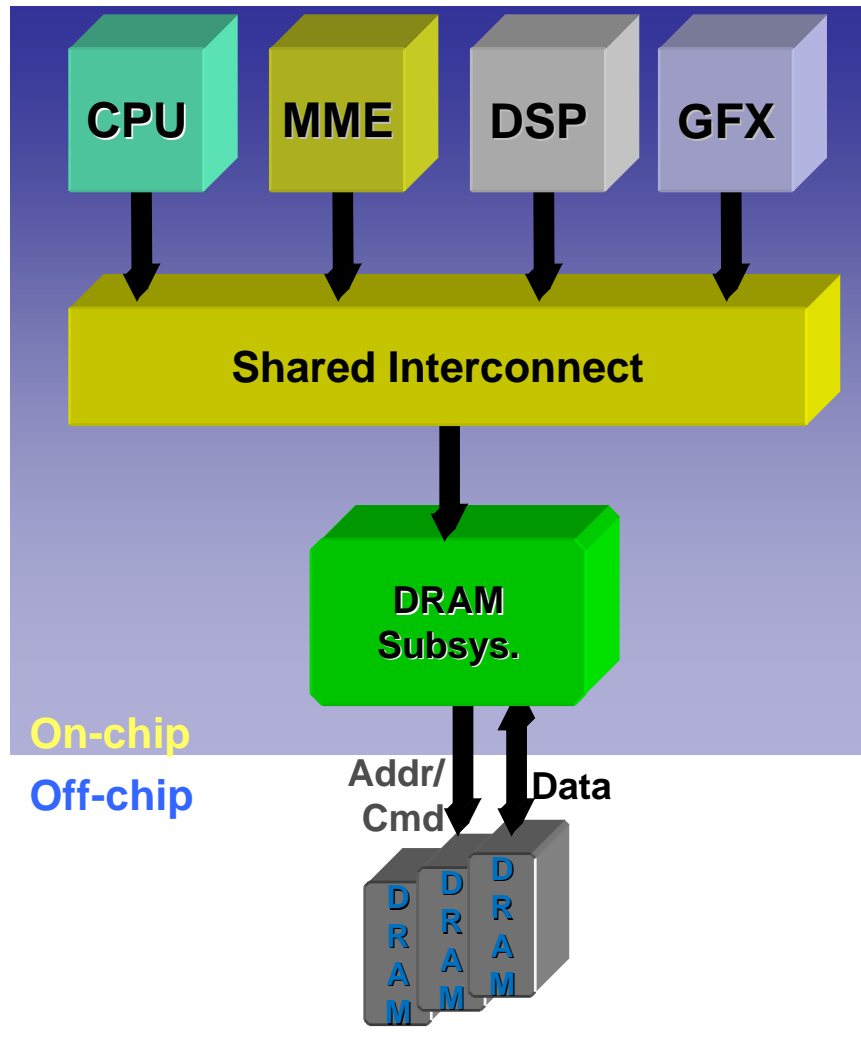
- LOTS of wires/congestion
- Lots of small/inefficient FIFO's
- Large part of system must be BW matched to DRAM

**Imagine Dedicated Links From > 100
DRAM-connected Cores!**

Single-ported Memory Subsystems



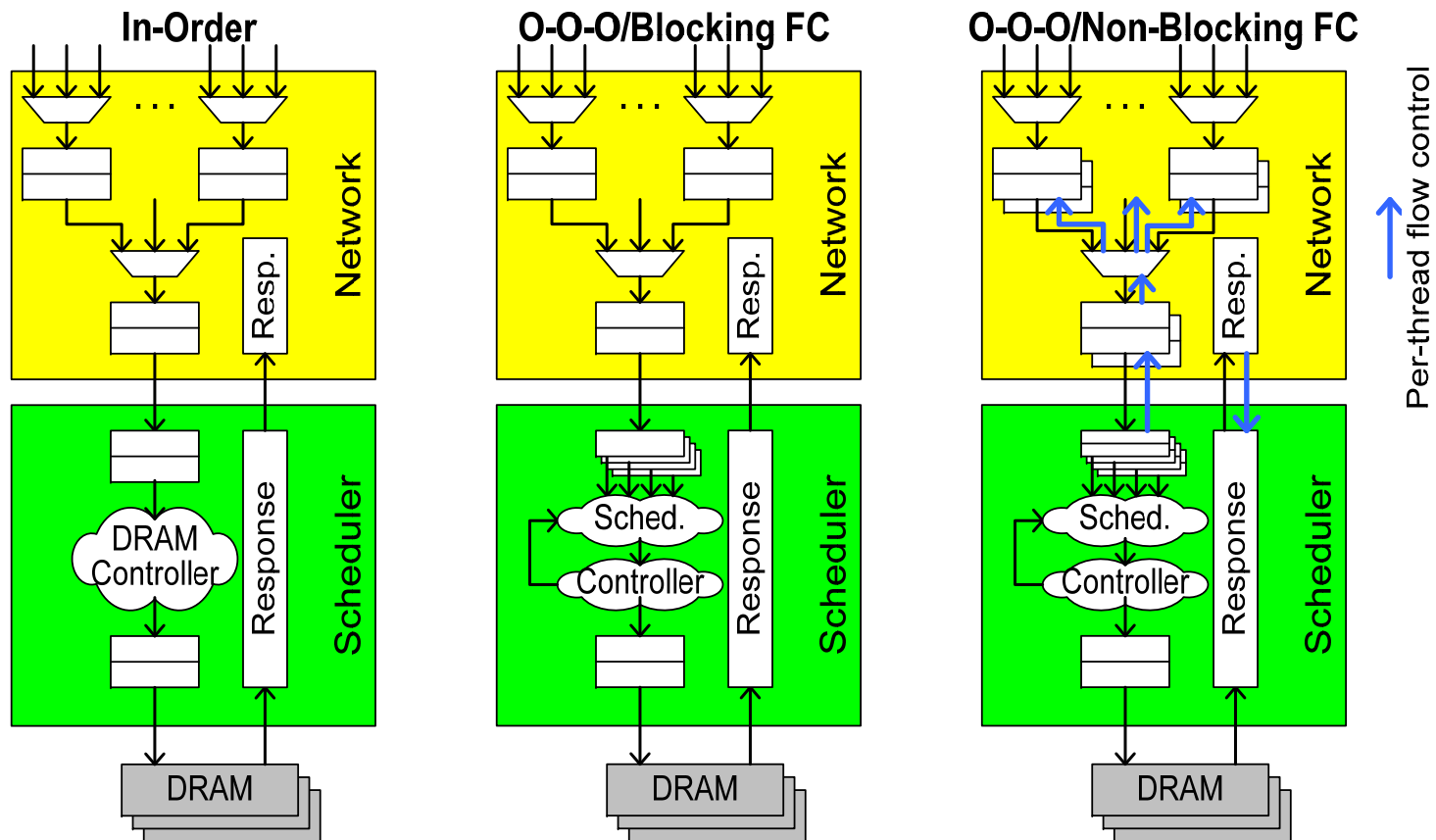
Shared Interconnect Approach



- Interconnect presents requests in series to single-port scheduler
- Saves wires/congestion
- But:
 - Interconnect arbitration impacts scheduling
 - Risks lower utilization
 - May not meet deadlines
 - Where do FIFO's live?
 - How much of system is BW-matched to DRAM?
 - System performance also limited by communication system

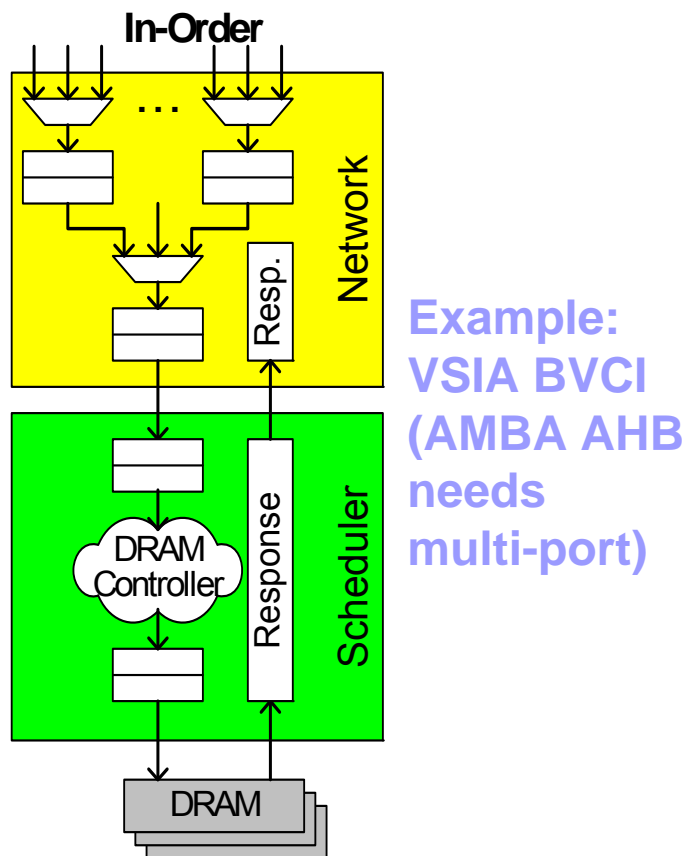
Single-port DRAM Protocols

- Interconnect and subsystem must support multiple outstanding requests (cover DRAM pipeline depth)



In-order Protocol

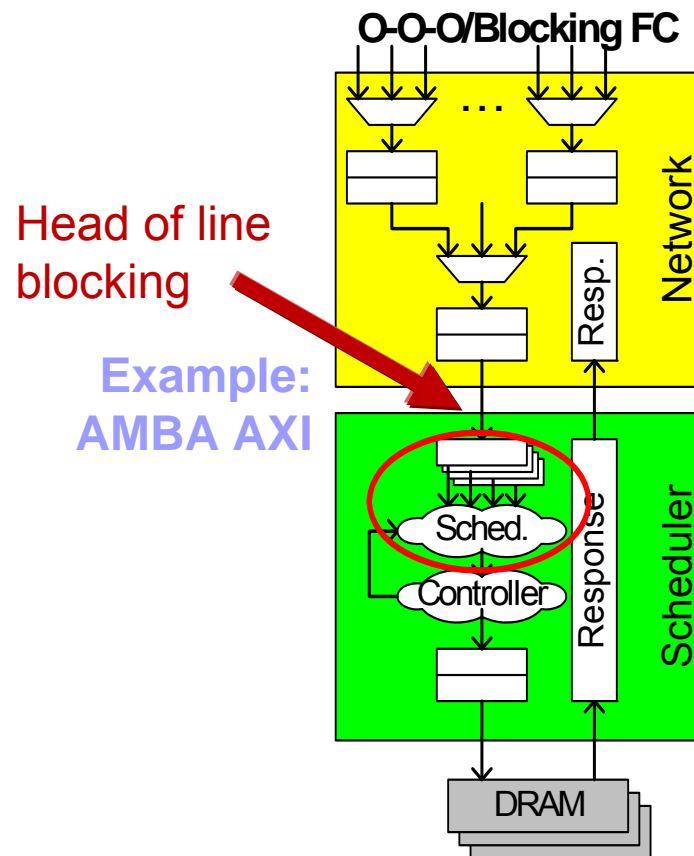
- Interface protocol supports multiple outstanding burst requests, but all service matches request order



- Simplest scheme (lowest hardware cost)
- Service order determined by interconnect arbitration
- Scheduler can only optimize pipeline (looking ahead for page misses to other banks)
- High efficiency requires long bursts, leads to high latency (poor QoS)

Out-of-order Protocol with Blocking FC

- Interface protocol provides ordering tags to allow scheduler to reorder some requests, but flow control is shared across all tags



- Interconnect presents requests in order
 - Scheduler queues requests & chooses order to optimize throughput and QoS
- But
- Bursty flows can fill queues, hurting latency & BW for others
 - Full queues block into network
 - Frequency scales poorly with queue depth

Single-port DRAM Subsystem Protocols



Ordering/ flow control	In-order/ blocking	Out-of-order/ blocking	Out-of-order/ non-blocking
Peak BW limited by	DRAM	DRAM	DRAM
Ordering flexibility	None	High	High
Queuing	None	Shared	Per-thread
Compiled RAM-friendly	No	No	Yes
Init. BW==DRAM BW	Yes	Yes	No
DRAM efficiency	Medium	High	High
Max. CPU latency	High	Medium	Low
Data interleaving	None	Minor	High

Optimizing for High Utilization



Goal	Approach
Minimize RD/WR turnarounds	Group reads and writes
Hide page misses	Bank scheduling
Avoid page misses to banks with conflicting transfers in flight	Bank state tracking
Maximize page/bank scheduling opportunities at minimum area	Expose intrinsic traffic concurrency to scheduler
Ensure write data bursts at DDR rate	Buffer write data in DDR clock domain
Ensure read data absorbed at DDR rate	Buffer read data in DDR clock domain
Isolate SoC architecture from DDR clock	Asynchronous FIFO
Prevent low-bandwidth initiators from stalling DDR	Decoupling FIFO
Minimize CPU latency	Make CPU highest priority, interleave bursts & ensure paths cannot block
Eliminate page/bank thrashing	Protect groups of DDR bursts against higher priority traffic
Protect against best-effort traffic starvation	Demote QoS traffic overusing bandwidth & fair best-effort arbitration

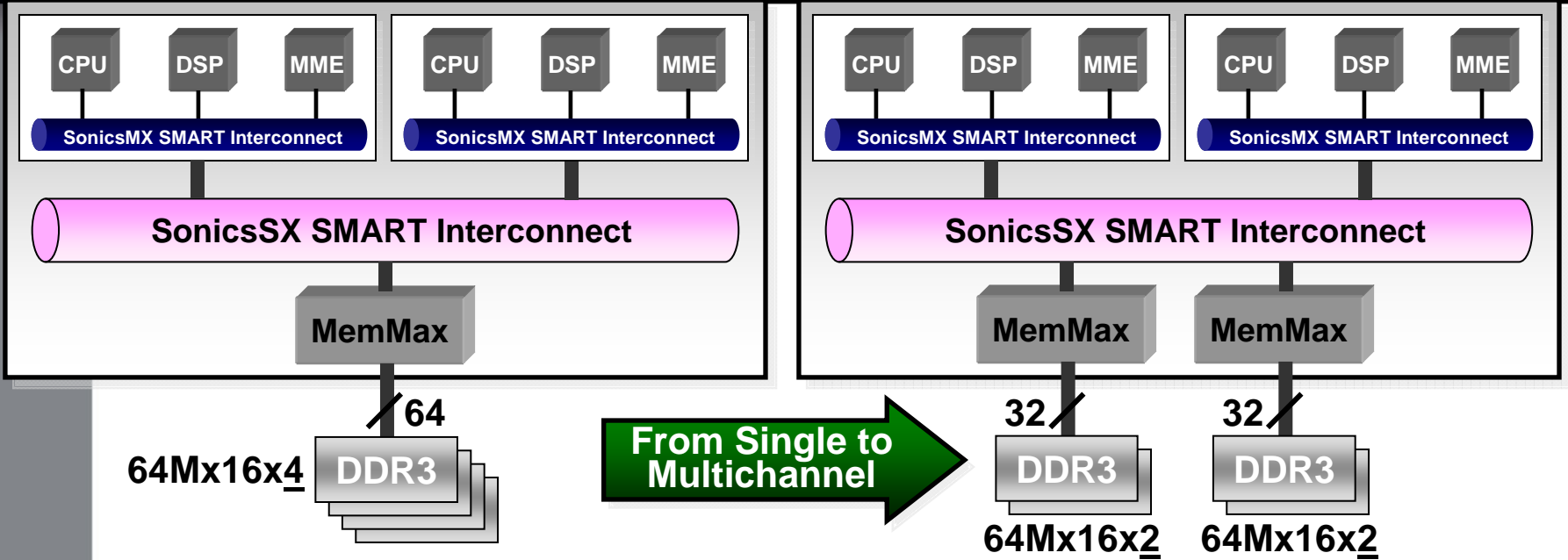
QoS-based Arbitration



- Initiator data flow threads mapped to DRAM threads by interconnect
 - e.g. 40 data flows sharing 8 DRAM threads in a digital video system
- Independent threads assigned to QoS level
- Non-blocking, multi-threaded fabric and DRAM interfaces allow:
 - Higher priority requests to interleave with & respond before others
 - Guaranteed BW threads to minimize buffering & receive latency guarantees
 - High DRAM utilization

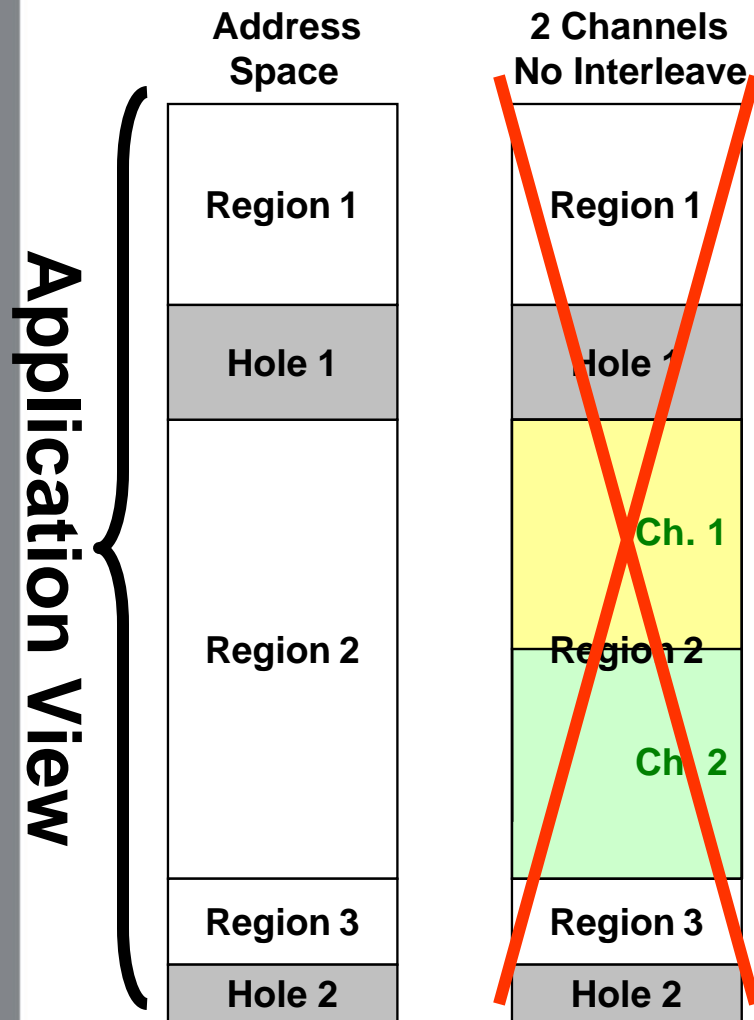
Thread QoS Level	Bandwidth Allocation ?	QoS Model
Priority	Yes	Low latency while within BW allocation, best-effort otherwise
Bandwidth	Yes	Guaranteed BW while within BW allocation, best-effort otherwise
Best-effort	No	N/A

Multichannel Solves Access Granularity



	DDR2	DDR3	DDR3
Channels	1	1	2
Data Width (B)	4	4	2
Effective BW	100%	84%	100%

Source: Customer (HDTV) System Dataflow
Constant Frequency/Ideal Load Balancing



Key Problems:

■ Load balancing

- Must balance memory traffic evenly among channels

■ Maintaining throughput

- Multiple channels cause throughput & ordering issues for pipelined memories

Software and IP cores must manage multiple channels explicitly

Interleaved Multichannel Technology (IMT*): Seamless Transition to Multichannel

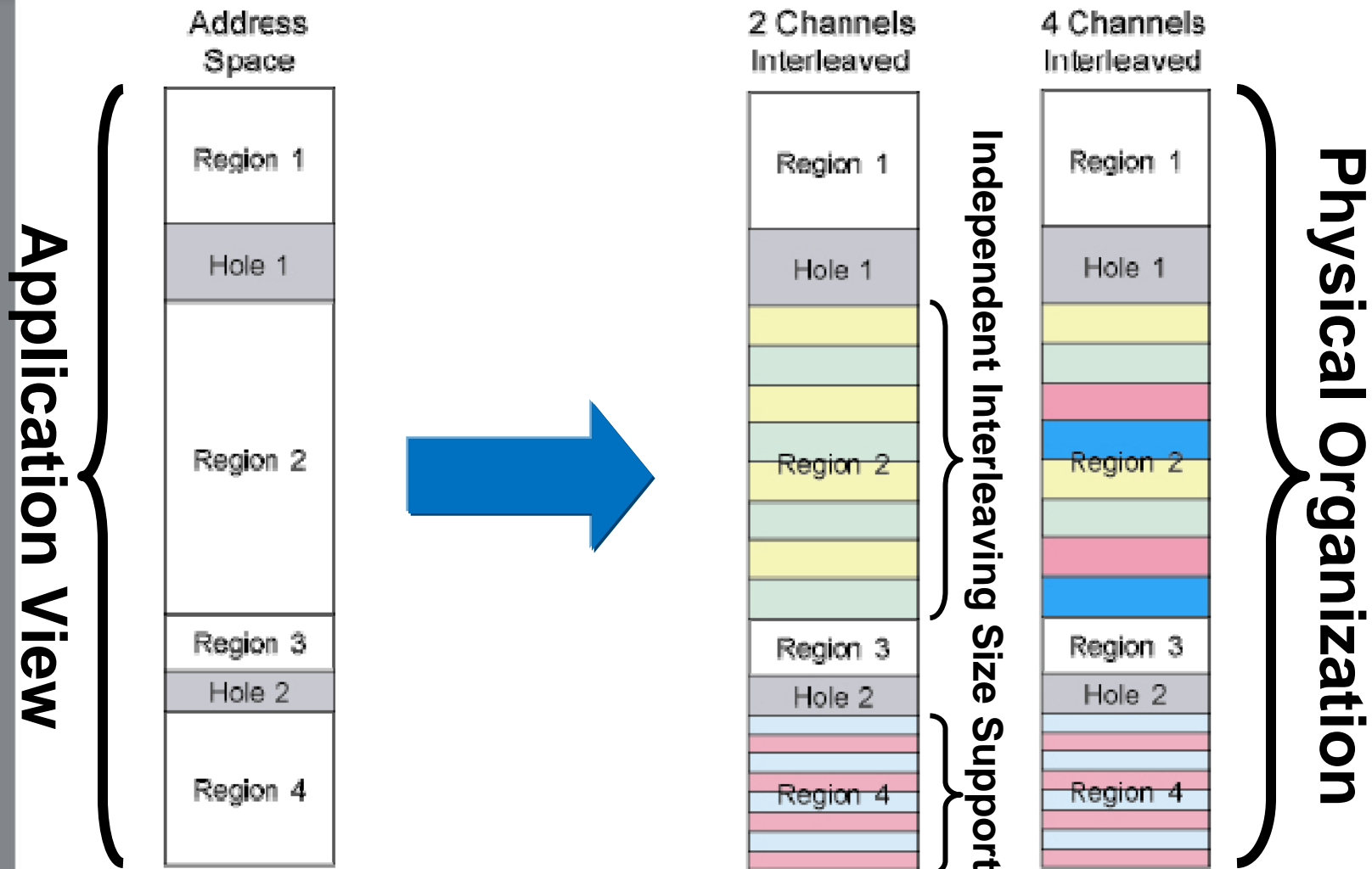


*patents pending

- **Interleaving support requires splitting traffic for delivery to proper channel**
 - Splitting in memory controller creates performance and routing congestion bottleneck
- **Predictably high performance**
 - Automatically spreads traffic across channels to ensure load balancing
 - Keeps DRAMs operating at full throughput, without costly reorder buffers
- **Scalable architecture**
 - Up to 8 interleaved channels within the same address region
 - Fully distributed to avoid bottlenecks & placement restrictions
- **Application flexibility**
 - Transparent to software and initiator hardware
 - Supports full or partial memory configurations – at run time

Multichannel Interleaving in the Interconnect
Higher Performance, Lower Area, More Scalable

Transparent Multichannel Interleaving with Access-Optimized Boundaries



2D Bursts, Address Tiling & Multichannel



■ Two-dimensional block bursts

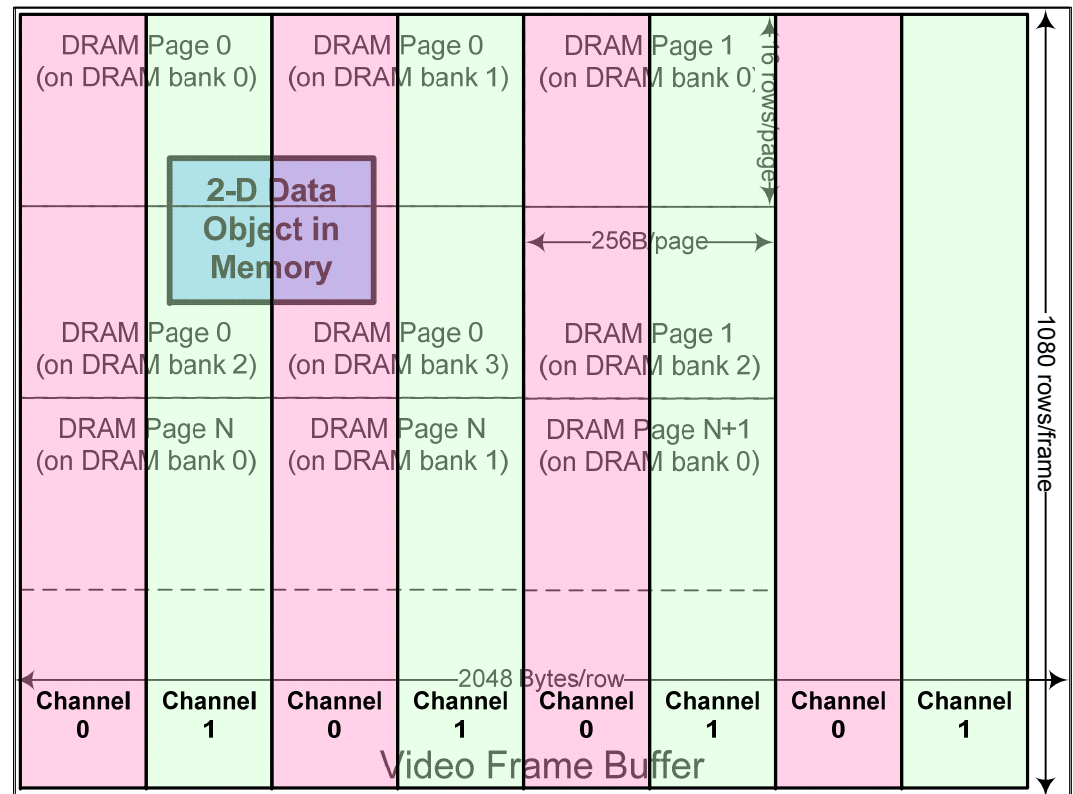
- 2D transaction using a single read/write command
- Popular for HD video and graphics

■ Address tiling

- Rearrange DRAM address organization to exploit 2D locality
- Avoids page misses
- >1 tiling schemes active at once

■ Channels divide buffer into columns

- Network splits 2D bursts that cross channel edges



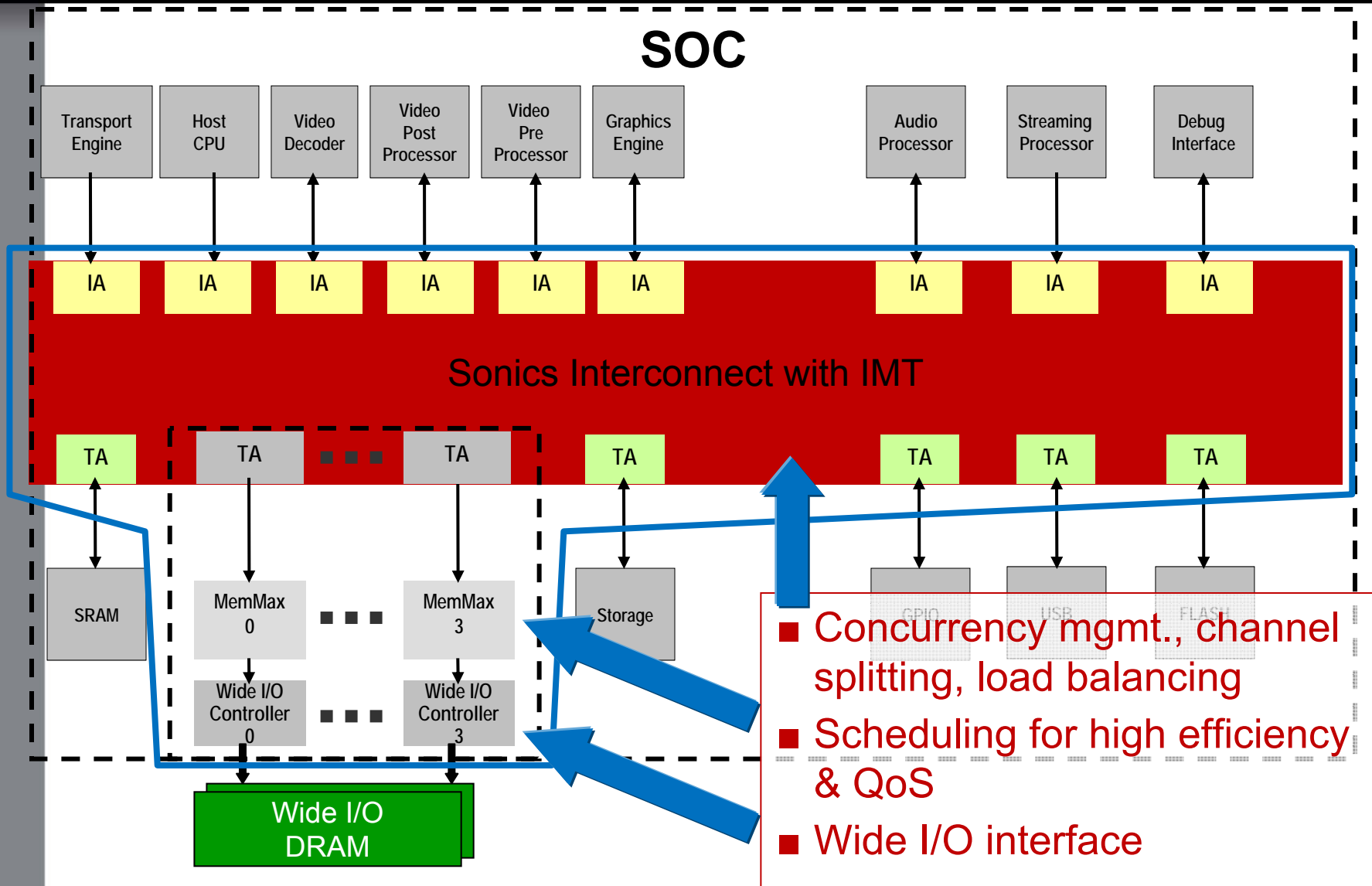
- Multicore consumer SoC background
- DRAM subsystem challenges
- Solution aspects
- ***Putting it all together***

DRAM-limited Consumer SoCs: Solution Requirements



- DRAM subsystems optimized for high utilization *and* good Quality of Service (QoS) characteristics
- On chip interconnection networks that manage the large and increasing numbers of DRAM consumers
 - And protect the IP cores from DRAM evolution
- Solutions to inefficiencies due to different access patterns and access granularities
- Analysis tooling to enable SoC architecture exploration and performance validation

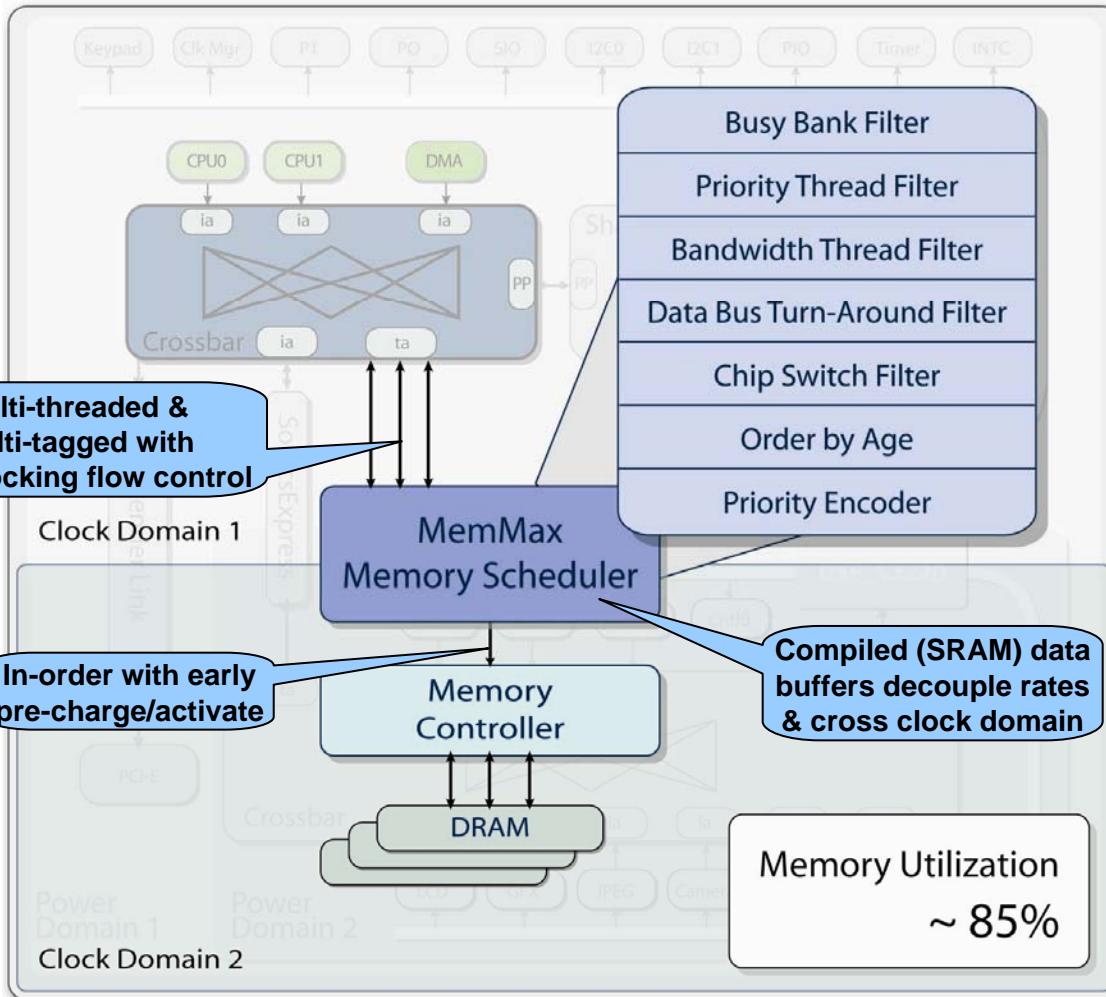
Example Wide I/O System Solution



MemMax™ Memory Scheduler



MemMax



- Address tiling
- Request grouping
- 2D burst support
- Guaranteed bandwidth QoS with demotion
- Per-thread buffer sizing
- Run time re-programmable



SONICS



THANK YOU!