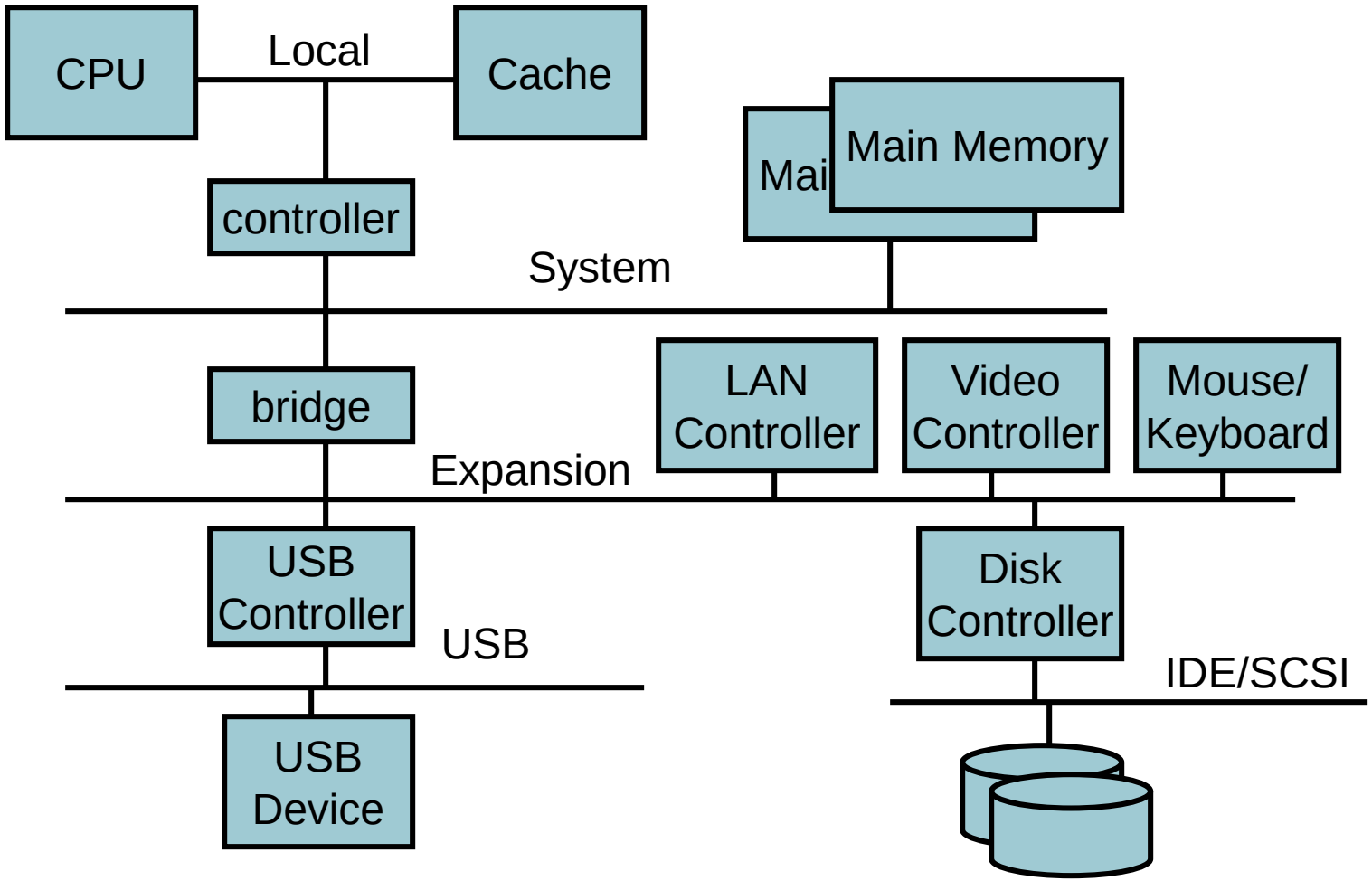
Chapter 6

Memory and Other I/O Topics

Interconnecting Components

- Need interconnections between
  - CPU, memory, I/O controllers
- Bus: shared communication channel
  - Parallel set of wires for data and synchronization of data transfer
  - Can become a bottleneck
- Performance limited by physical factors
  - Wire length, number of connections
- More recent alternative: high-speed serial connections with switches
  - Like networks

Hierarchical Bus Architecture



Bus Types

- Processor-Memory buses
  - Short, high speed
  - Design is matched to memory organization
- I/O buses
  - Longer, allowing multiple connections
  - Specified by standards for interoperability
  - Connect to processor-memory bus through a bridge

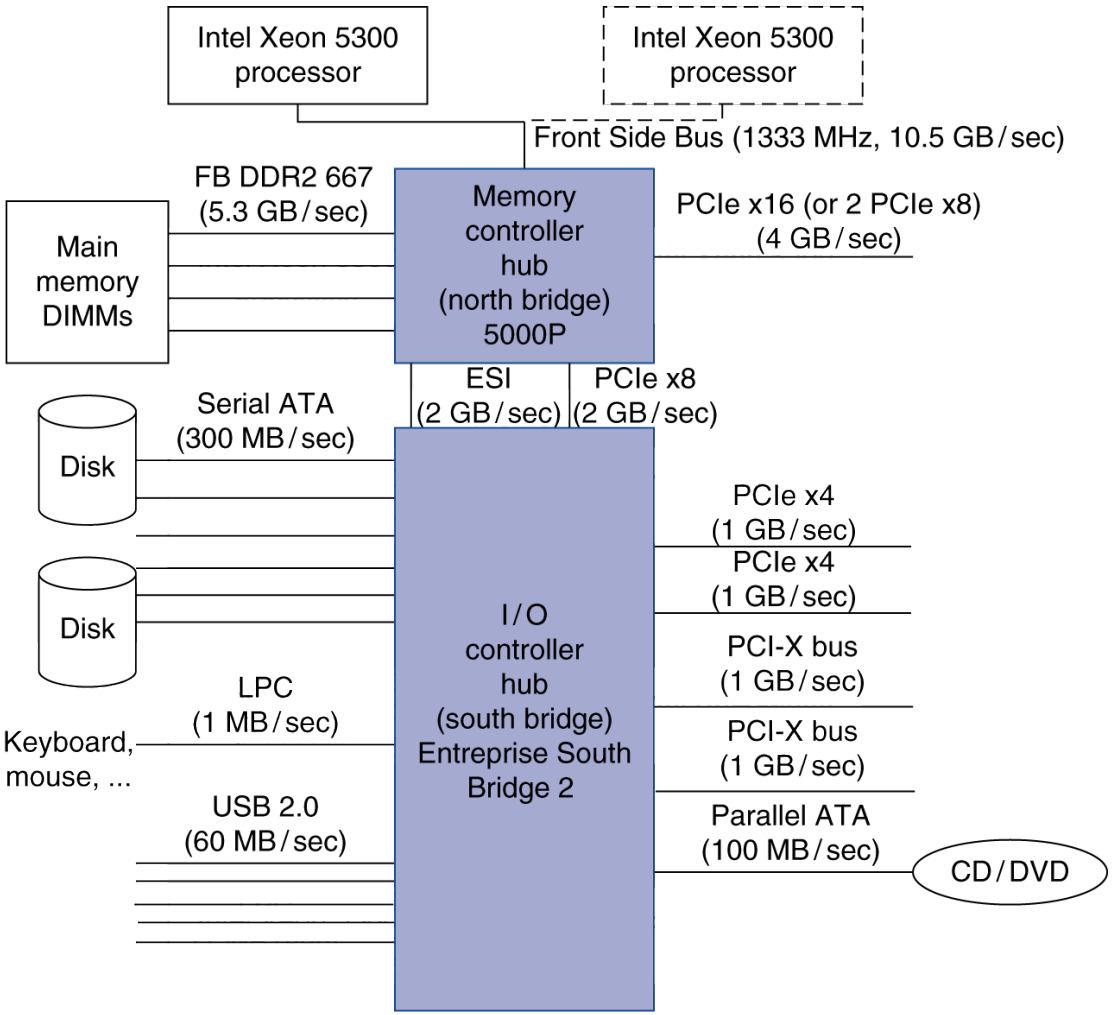
Bus Signals and Synchronization

- Data lines
  - Carry address and data
  - Multiplexed or separate
- Control lines
  - Indicate data type, synchronize transactions
- Synchronous
  - Uses a bus clock
- Asynchronous
  - Uses request/acknowledge control lines for handshaking

I/O Bus Examples

	Firewire	USB 2.0	PCI Express	Serial ATA	Serial Attached SCSI
Intended use	External	External	Internal	Internal	External
Devices per channel	63	127	1	1	4
Data width	4	2	2/lane	4	4
Peak bandwidth	50MB/s or 100MB/s	0.2MB/s, 1.5MB/s, or 60MB/s	250MB/s/lane 1×, 2×, 4×, 8×, 16×, 32×	300MB/s	300MB/s
Hot pluggable	Yes	Yes	Depends	Yes	Yes
Max length	4.5m	5m	0.5m	1m	8m
Standard	IEEE 1394	USB Implementers Forum	PCI-SIG	SATA-IO	INCITS TC T10

Typical x86 PC I/O System



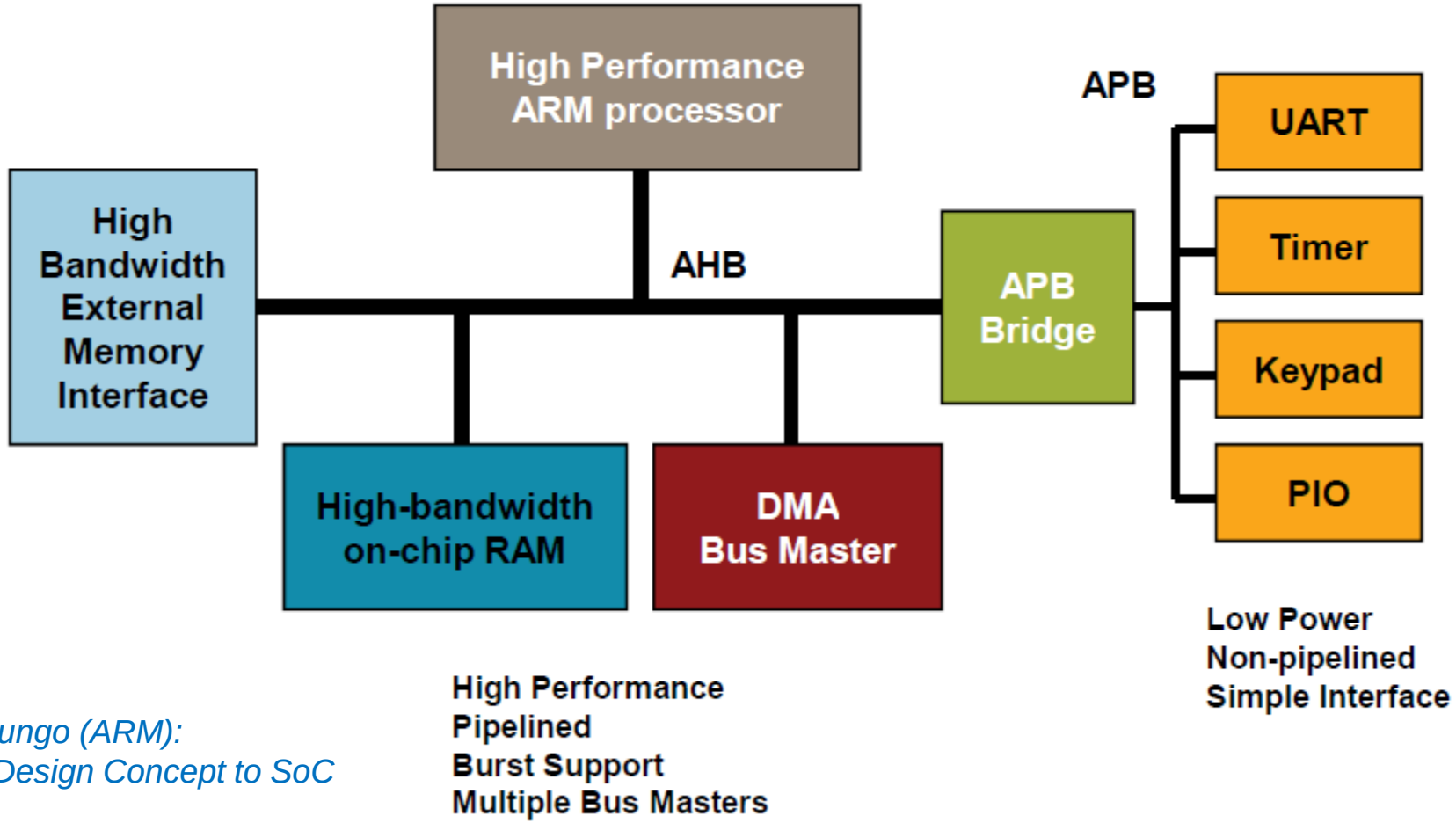
Intel & AMD I/O chip sets

	Intel 5000P chip set	Intel 975X chip set	AMD 580X CrossFiret
Target segment	Server	Performance PC	Server/Performance PC
Front Side Bus (64 bit)	1066/1333 MHz	800/1066 MHz	—
Memory controller hub ("north bridge")			
Product name	Blackbird 5000P MCH	975X MCH	
Pins	1432	1202	
Memory type, speed	DDR2 FBDIMM 667/533	DDR2 800/667/533	
Memory buses, widths	4 × 72	1 × 72	
Number of DIMMs, DRAM/DIMM	16, 1 GB/2 GB/4 GB	4, 1 GB/2 GB	
Maximum memory capacity	64 GB	8 GB	
Memory error correction available?	Yes	No	
PCIe/External Graphics Interface	1 PCIe x16 or 2 PCIe x	1 PCIe x16 or 2 PCIe x8	
South bridge interface	PCIe x8, ESI	PCIe x8	
I/O controller hub ("south bridge")			
Product name	6321 ESB	ICH7	580X CrossFire
Package size, pins	1284	652	549
PCI-bus: width, speed	Two 64-bit, 133 MHz	32-bit, 33 MHz, 6 masters	—
PCI Express ports	Three PCIe x4		Two PCIe x16, Four PCI x1
Ethernet MAC controller, interface	—	1000/100/10 Mbit	—
USB 2.0 ports, controllers	6	8	10
ATA ports, speed	One 100	Two 100	One 133
Serial ATA ports	6	2	4
AC-97 audio controller, interface	—	Yes	Yes
I/O management	SMbus 2.0, GPIO	SMbus 2.0, GPIO	ASF 2.0, GPIO

ARM Advanced Microcontroller Bus Architecture (AMBA)

- On-chip interconnect specification for SoC
- Promotes re-use by defining a common backbone for SoC modules using standard bus architectures
  - AHB – Advanced High-performance Bus (system backbone)
    - High-performance, high clock freq. modules
    - Processors to on-chip memory, off-chip memory interfaces
  - APB – Advanced Peripheral Bus
    - Low-power peripherals
    - Reduced interface complexity
  - ASB – Advanced System Bus
    - High performance alternate to AHB
  - AXI – Advanced eXtensible Interface
  - ACE – AXI Coherency Extension
  - ATB – Advanced Trace Bus

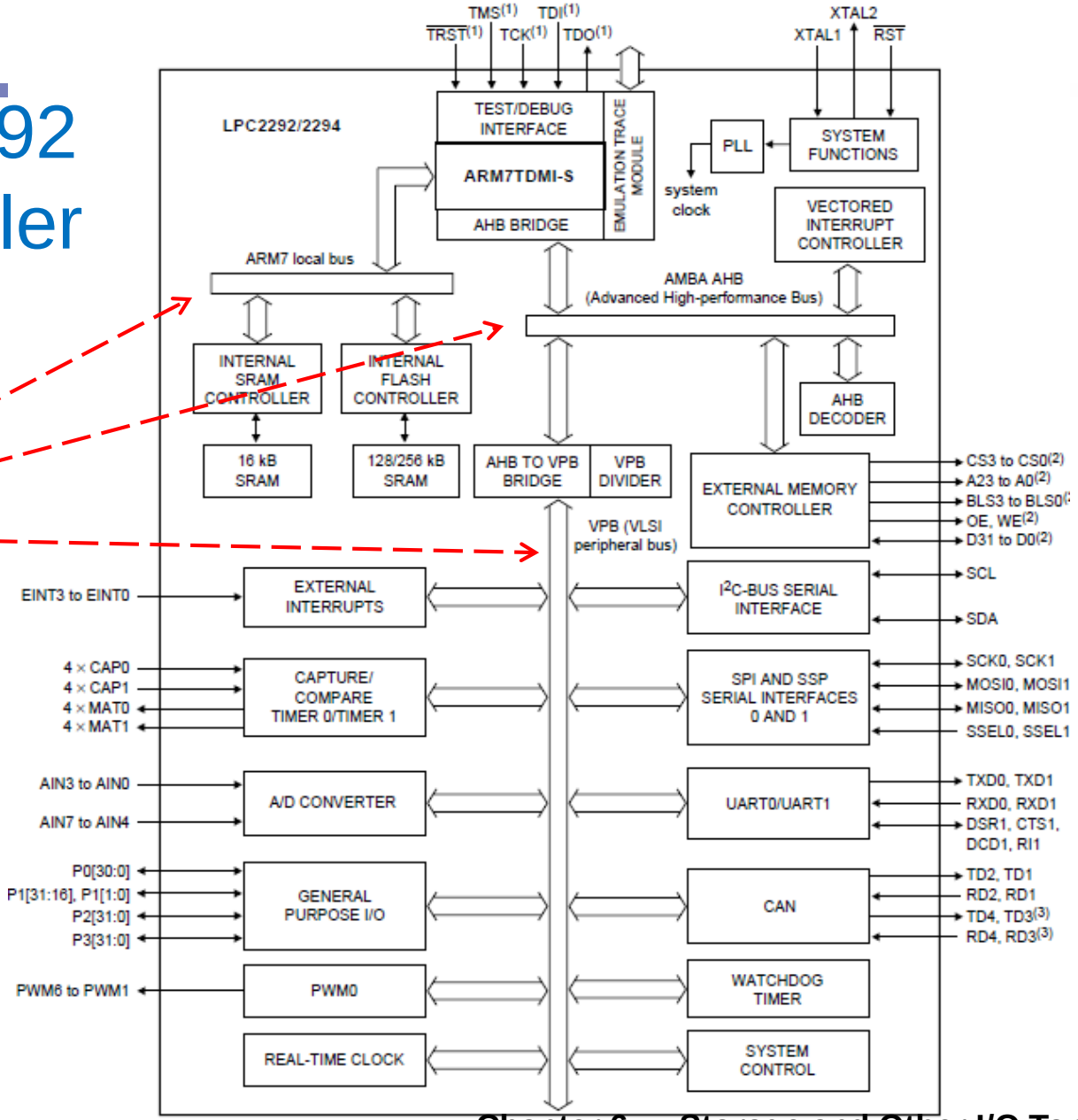
Example AMBA System



Joe Bungo (ARM):
CPU Design Concept to SoC

NXP LPC2292 Microcontroller

ARM buses



ARM Cortex-A9 System IP

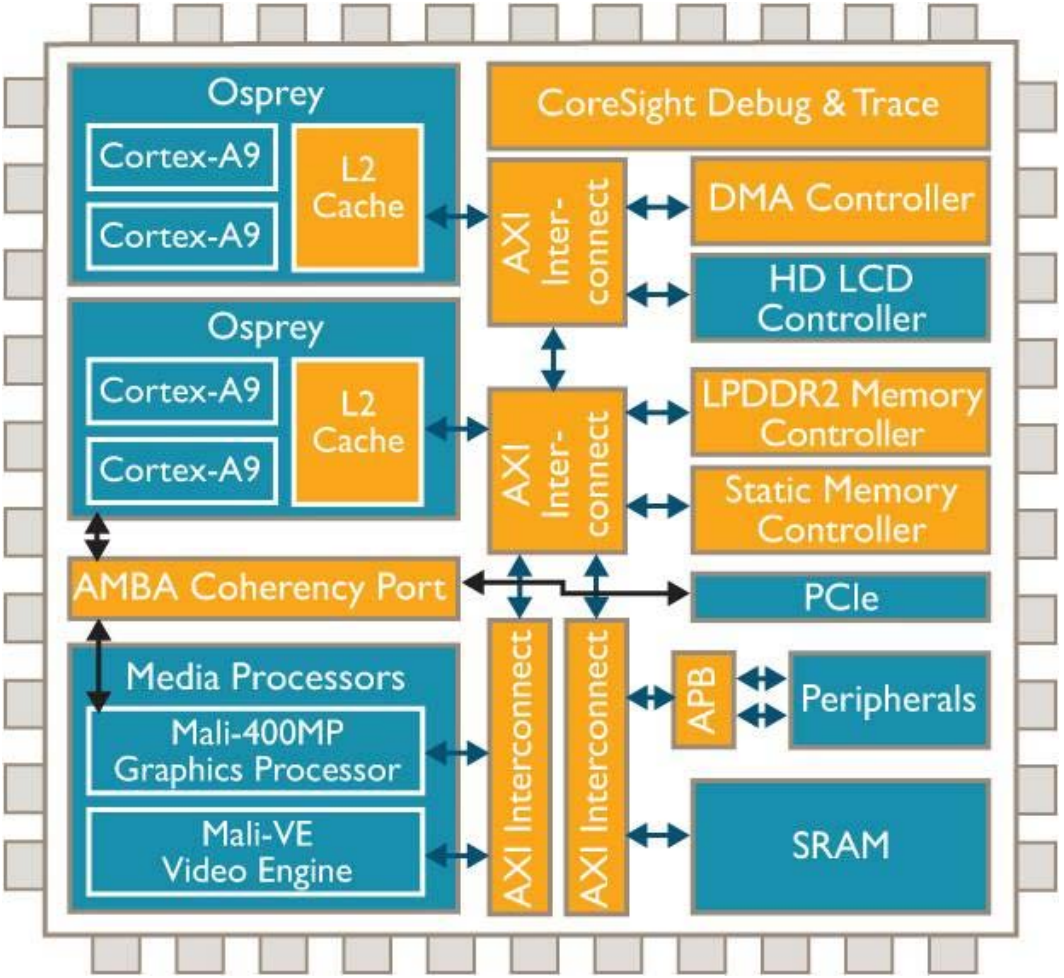
Interconnect SoC components

Description	AMBA Bus	System IP Components
Advanced AMBA 3 Interconnect IP	AXI	NIC-301, PL301
DMA Controller	AXI	DMA-330 , PL330
Level 2 Cache Controller	AXI	L2C-310 , PL310
Dynamic Memory Controller	AXI	DMC-340 , PL340
DDR2 Dynamic Memory Controller	AXI	DMC-342
Static Memory Controller	AXI	SMC-35x , PL35x
TrustZone Address Space Controller	AXI	PL380
CoreSight™ Design Kit	ATB	CDK-11

CoreLink peripherals for AMBA

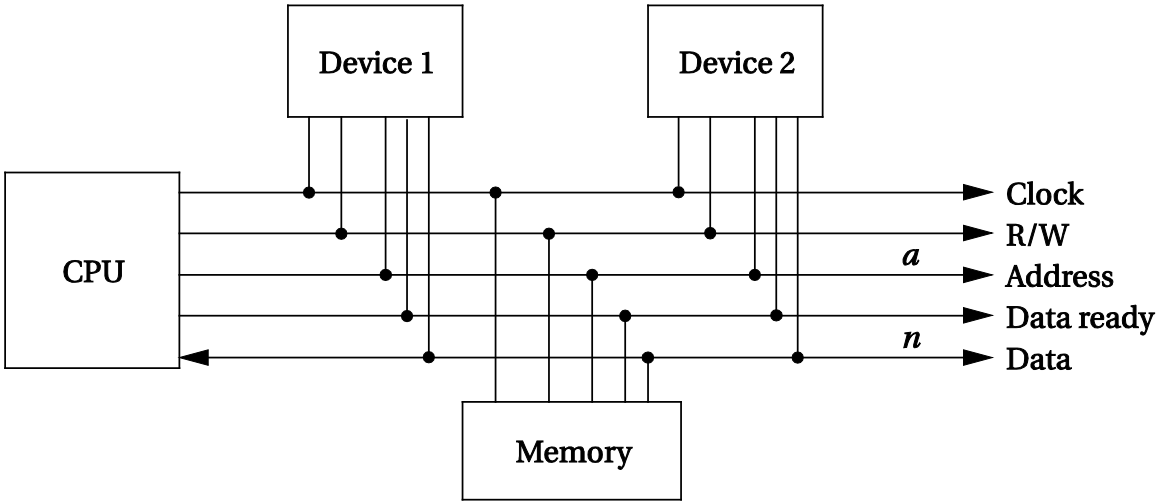
Coretex-A9 SoC

“CoreLink” = interconnect + memory controllers for Cortex/Mali

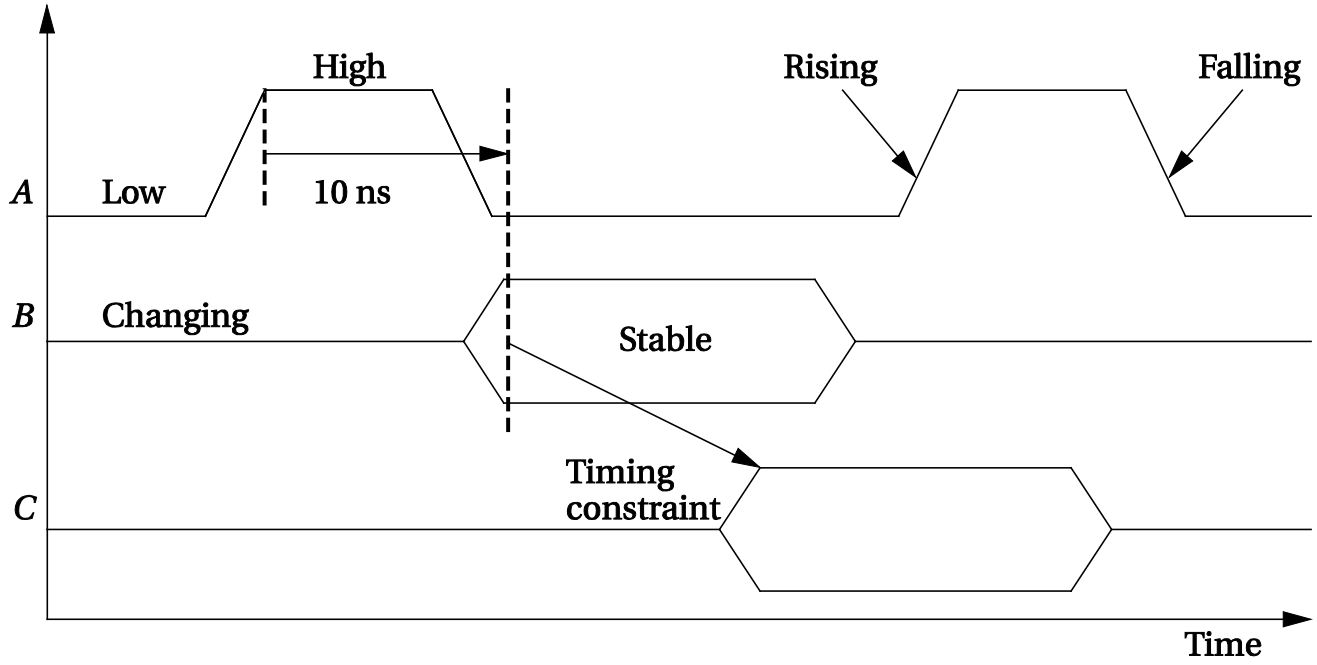


Microprocessor buses

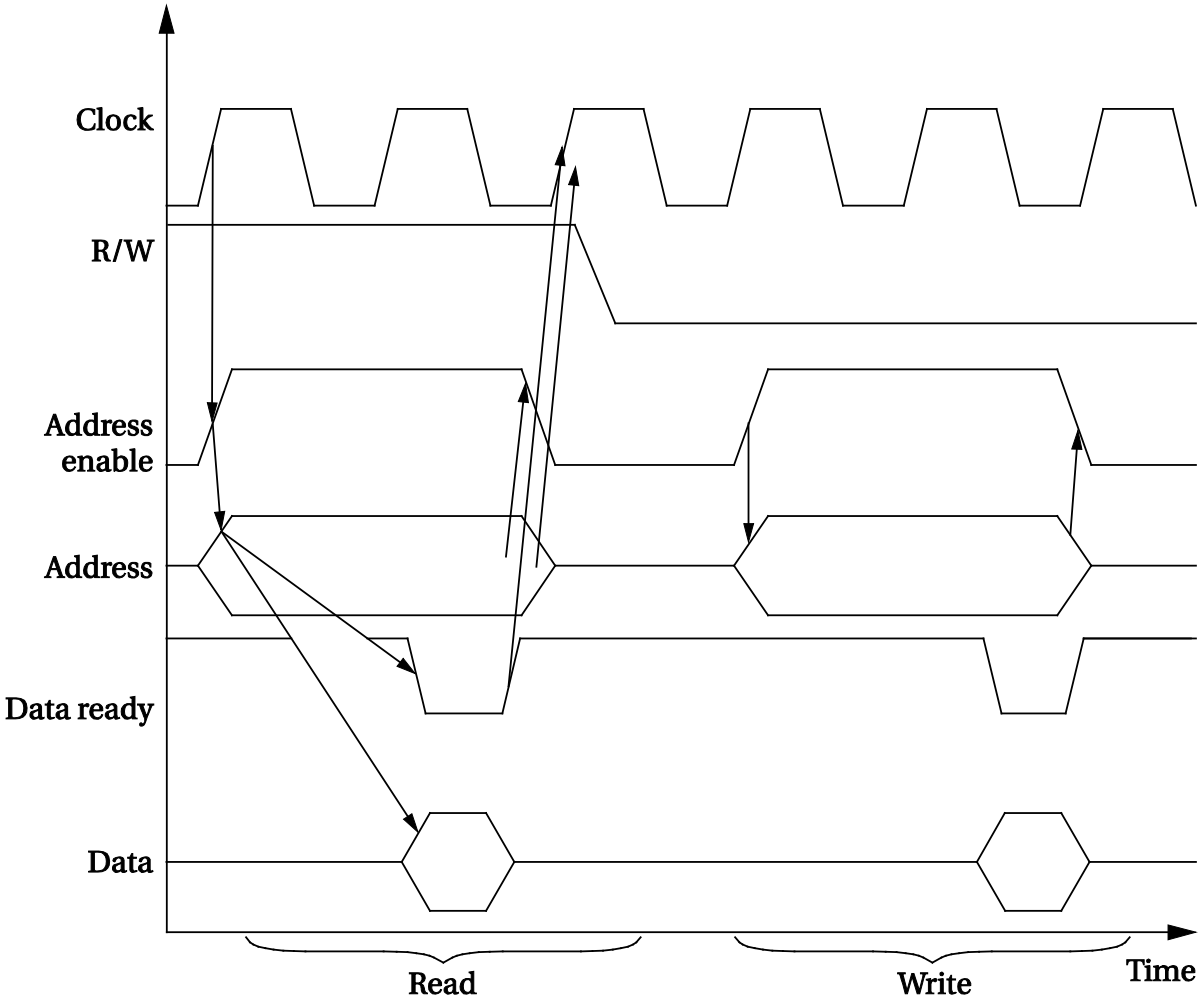
- Clock provides synchronization.
- R/W is true when reading (R/W' is false when reading).
- Address is a-bit bundle of address lines.
- Data is n-bit bundle of data lines.
- Data ready signals when n-bit data is ready.



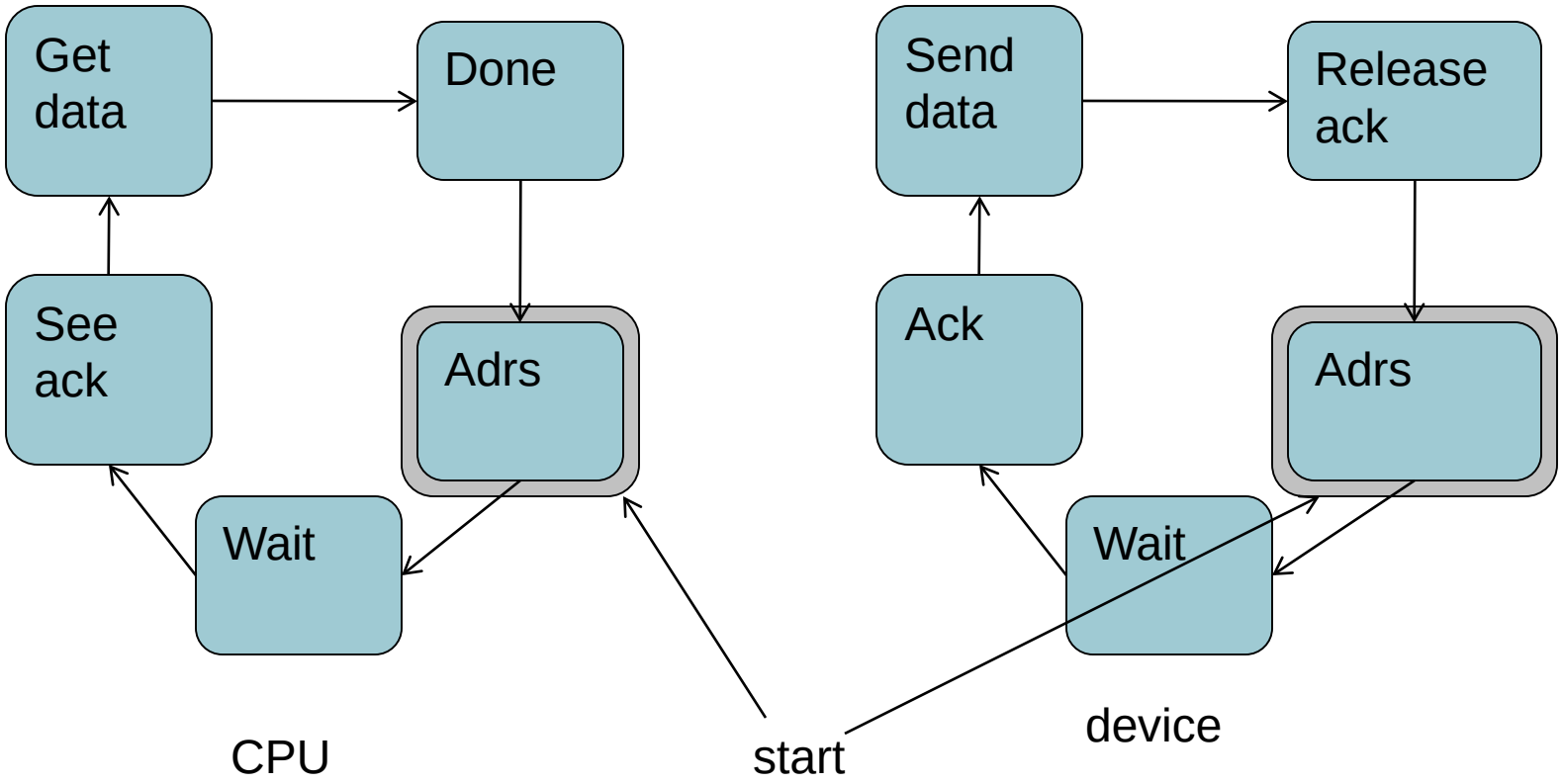
Timing diagrams



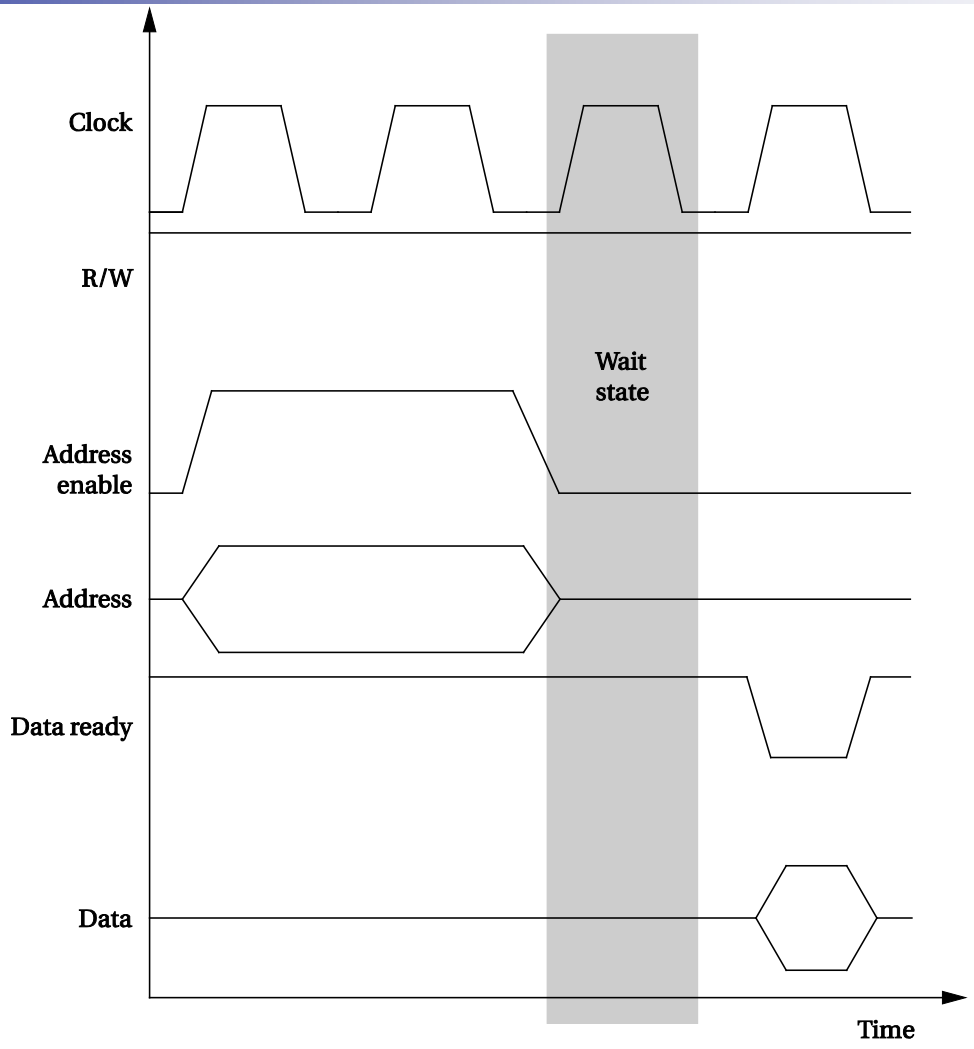
Bus read and write



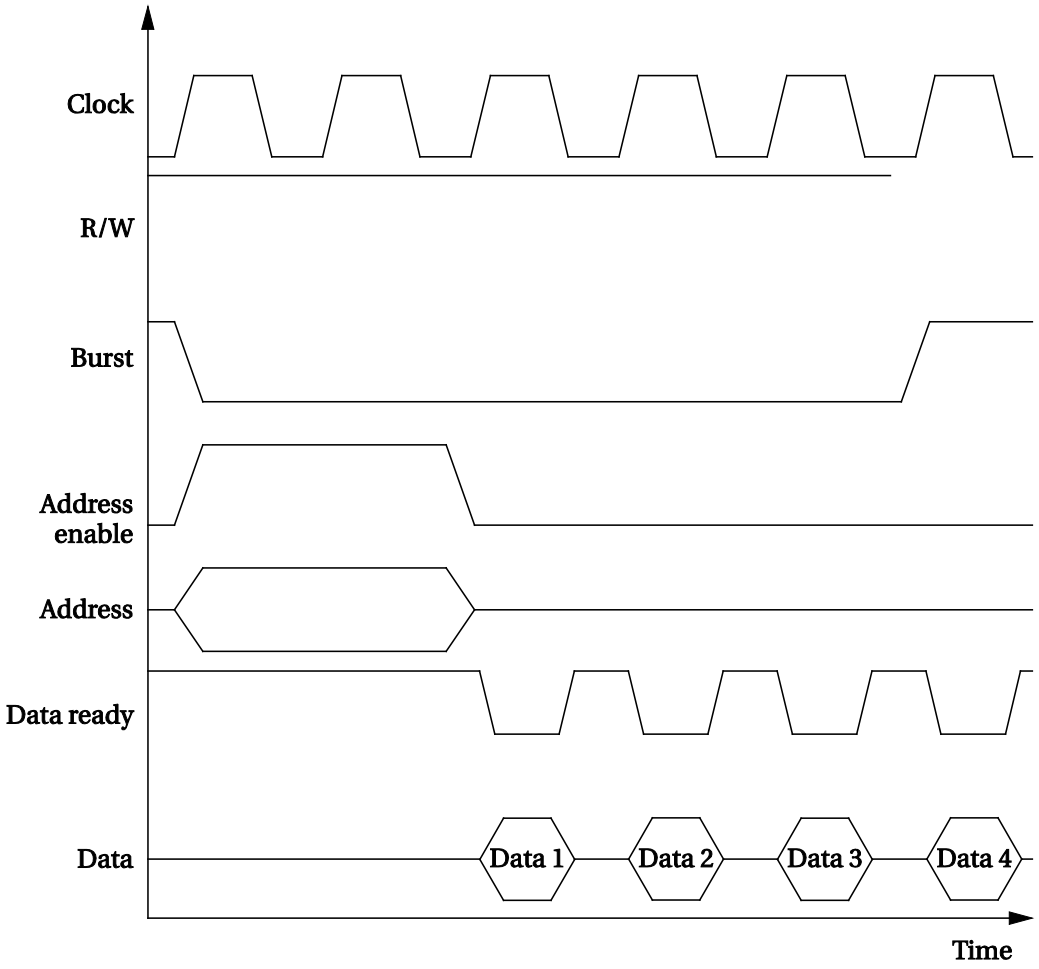
State diagrams for bus read



Bus wait state

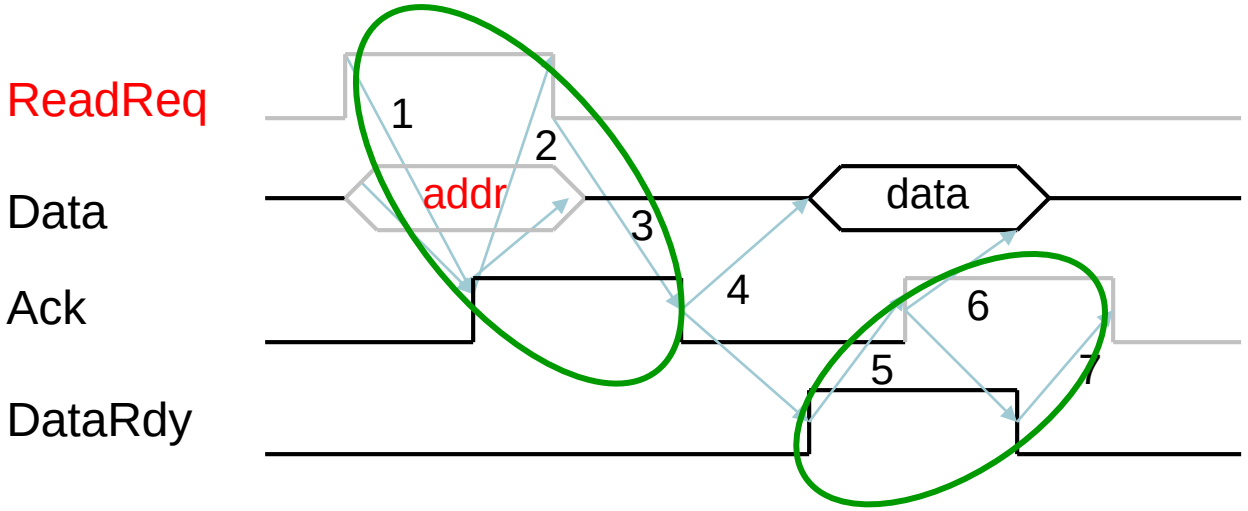


Bus burst read



Asynchronous Bus Handshaking Protocol

- Output (read) data from memory to an I/O device

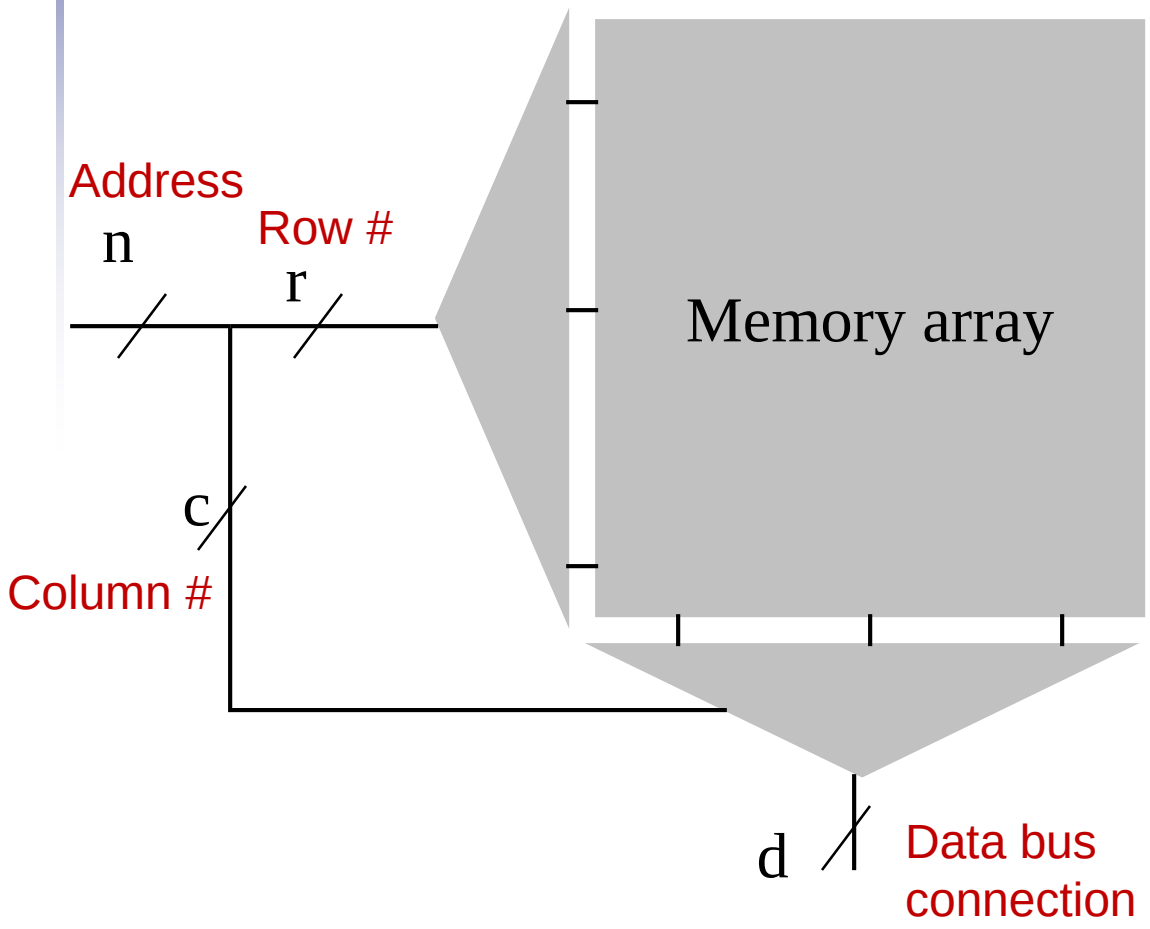


I/O device signals a request by raising ReadReq and putting the addr on the data lines

1. Memory sees ReadReq, reads addr from data lines, and raises Ack
2. I/O device sees Ack and releases the ReadReq and data lines
3. Memory sees ReadReq go low and drops Ack
4. When memory has data ready, it places it on data lines and raises DataRdy
5. I/O device sees DataRdy, reads the data from data lines, and raises Ack
6. Memory sees Ack, releases the data lines, and drops DataRdy
7. I/O device sees DataRdy go low and drops Ack

Memory device organization

Memory "organization" = 2^n x d
(from system designer's perspective)

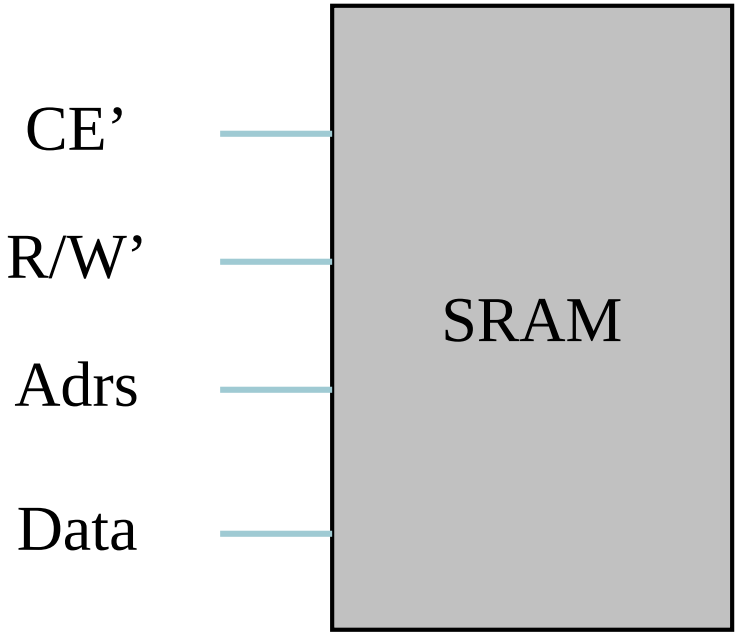


- Size.
  - Address width.

    n = r + c
  - DRAM:

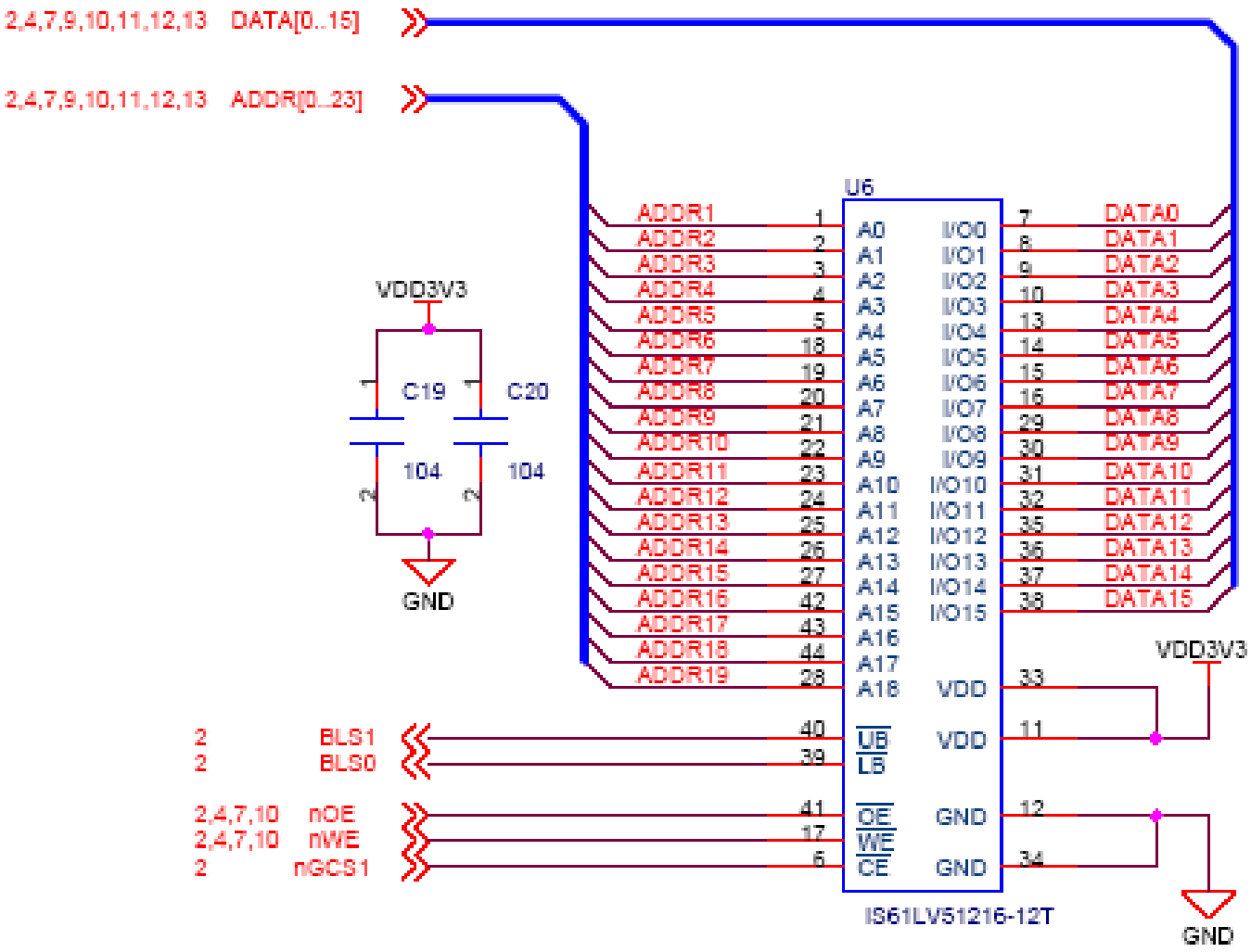
    mux r & c bits
- Aspect ratio.
  - Data width d.

Typical generic SRAM

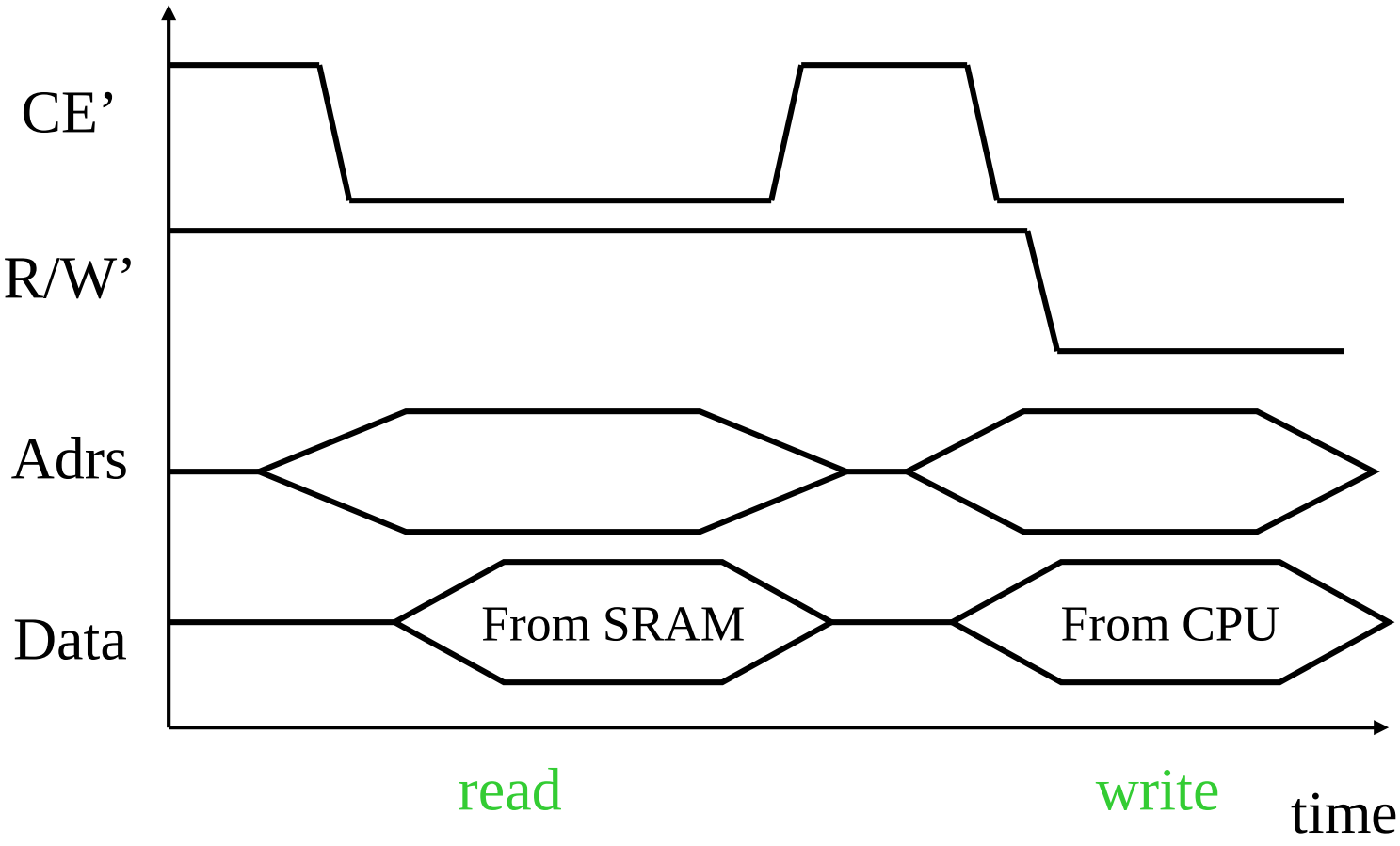


- Often have separate OE’ and WE’ instead of one R/W’ signal.
- Multi-byte Data bus devices usually have byte-select signals.

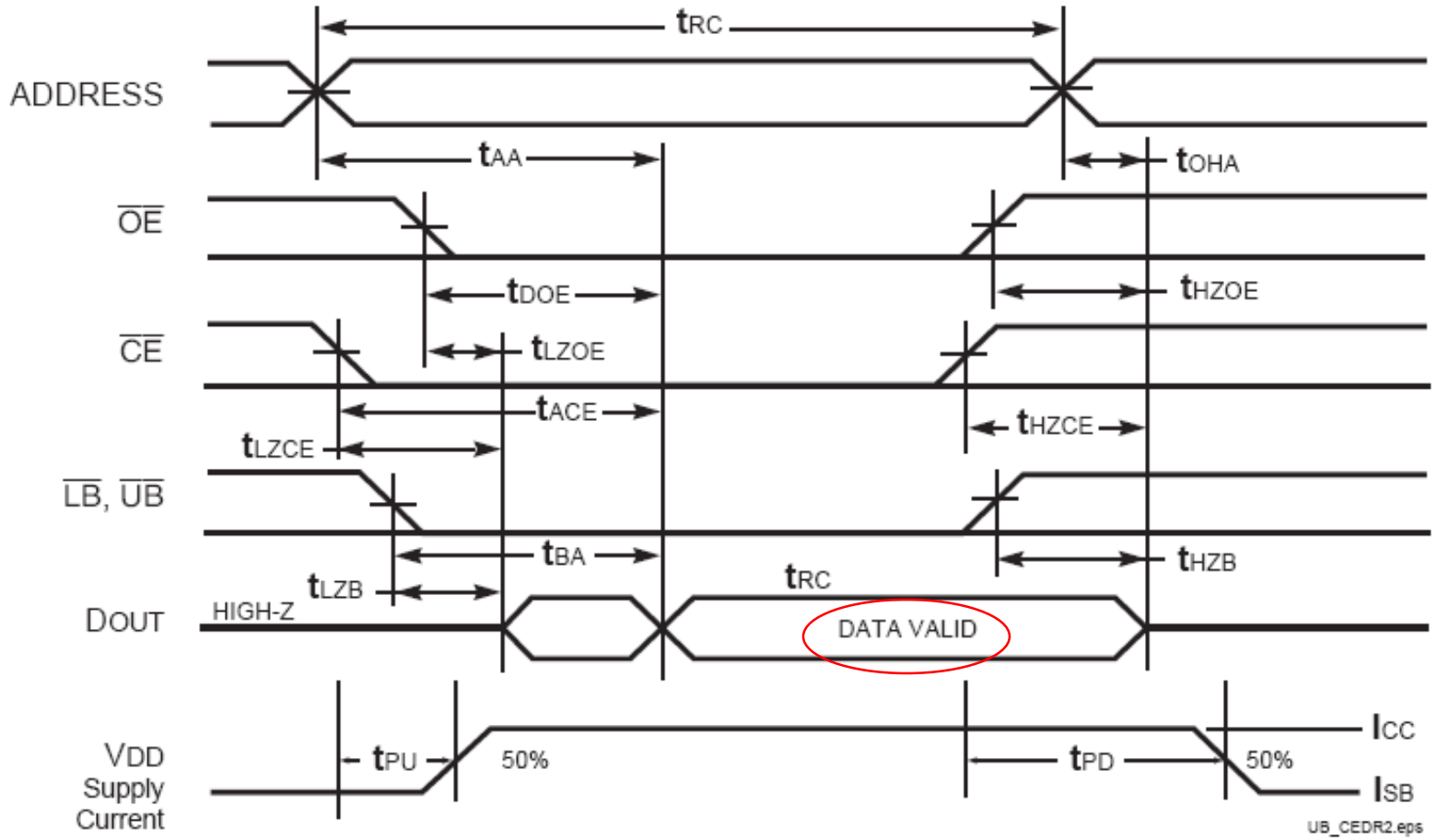
512K x 16 SRAM (on uCdragon board)



Generic SRAM timing



ISSI *IS61LV51216* SRAM read cycle



ISSI *IS61LV51216* SRAM timing

READ CYCLE SWITCHING CHARACTERISTICS[1] (Over Operating Range)

Symbol	Parameter	-8		-10		-12		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
t_{RC}	Read Cycle Time	8	—	10	—	12	—	ns
t_{AA}	Address Access Time	—	8	—	10	—	12	ns
t_{OHA}	Output Hold Time	3	—	3	—	3	—	ns
t_{ACE}	$\overline{CE}$ Access Time	—	8	—	10	—	12	ns
t_{DOE}	$\overline{OE}$ Access Time	—	3.5	—	4	—	5	ns
t_{HZOE}[2]	$\overline{OE}$ to High-Z Output	—	3	—	4	0	5	ns
t_{LZOE}[2]	$\overline{OE}$ to Low-Z Output	0	—	0	—	0	—	ns
t_{HZCE}[2	$\overline{CE}$ to High-Z Output	0	3	0	4	0	6	ns
t_{LZCE}[2]	$\overline{CE}$ to Low-Z Output	3	—	3	—	3	—	ns
t_{BA}	$\overline{LB}$, $\overline{UB}$ Access Time	—	3.5	—	4	—	5	ns
t_{HZB}[2]	$\overline{LB}$, $\overline{UB}$ to High-Z Output	0	3	0	3	0	4	ns
t_{LZB}[2]	$\overline{LB}$, $\overline{UB}$ to Low-Z Output	0	—	0	—	0	—	ns
t_{PU}	Power Up Time	0	—	0	—	0	—	ns
t_{PD}	Power Down Time	—	8	—	10	—	12	ns

Design example #1

- Find the bandwidth of a synchronous bus
  - Clock period T = 50ns
  - 1 cycle required to xmit address/data
  - Bus width = 32 bits
  - Memory access time = 200ns

  Address + Memory-Read + Send_Data =
  50ns + 200ns + 50ns = 300ns
  BW = 4 Bytes/300ns = 13.3 Mbytes/sec

- Burst transfer 4 words from synchronous DRAM, at one clock each for words 2-3-4

  50ns + 200ns + 50ns + (3 x 50ns) = 450ns
  BW = 16 Bytes/450ns = 35.6 Mbytes/sec

Design example #2

- Asynchronous bus
  - 40ns to complete each “handshake” (HS)
  - Memory access time = 200ns
  - 32-bit data bus

(addr) (mem-read) (data xfer)
HS-1 + HS-2-3-4* + HS-5-6-7
40ns + 200ns + 3x40ns = 360ns
BW = 4 Bytes/360ns = 11.1 Mbytes/sec

- Memory read (200ns) concurrent with 3 handshakes
- (3 x 40ns), so memory read time dominates

I/O Management

- I/O is mediated by the OS
  - Multiple programs share I/O resources
    - Need protection and scheduling
  - I/O causes asynchronous interrupts
    - Same mechanism as exceptions
  - I/O programming is fiddly
    - OS provides abstractions to programs

I/O Commands

- I/O devices are managed by I/O controller hardware
  - Transfers data to/from device
  - Synchronizes operations with software
- Command registers
  - Cause device to do something
- Status registers
  - Indicate what the device is doing and occurrence of errors
- Data registers
  - Write: transfer data to a device
  - Read: transfer data from a device

I/O Register Mapping

- Memory mapped I/O
  - Registers are addressed in same space as memory
  - Address decoder distinguishes between them
  - OS uses address translation mechanism to make them only accessible to kernel
- I/O instructions
  - Separate instructions to access I/O registers
  - Can only be executed in kernel mode
  - Example: x86

Polling

- Periodically check I/O status register
  - If device ready, do operation
  - If error, take action
- Common in small or low-performance real-time embedded systems
  - Predictable timing
  - Low hardware cost
- In other systems, wastes CPU time

Interrupts

- When a device is ready or error occurs
  - Controller interrupts CPU
- Interrupt is like an exception
  - But not synchronized to instruction execution
  - Can invoke handler between instructions
  - Cause information often identifies the interrupting device
- Priority interrupts
  - Devices needing more urgent attention get higher priority
  - Can interrupt handler for a lower priority interrupt

I/O Data Transfer

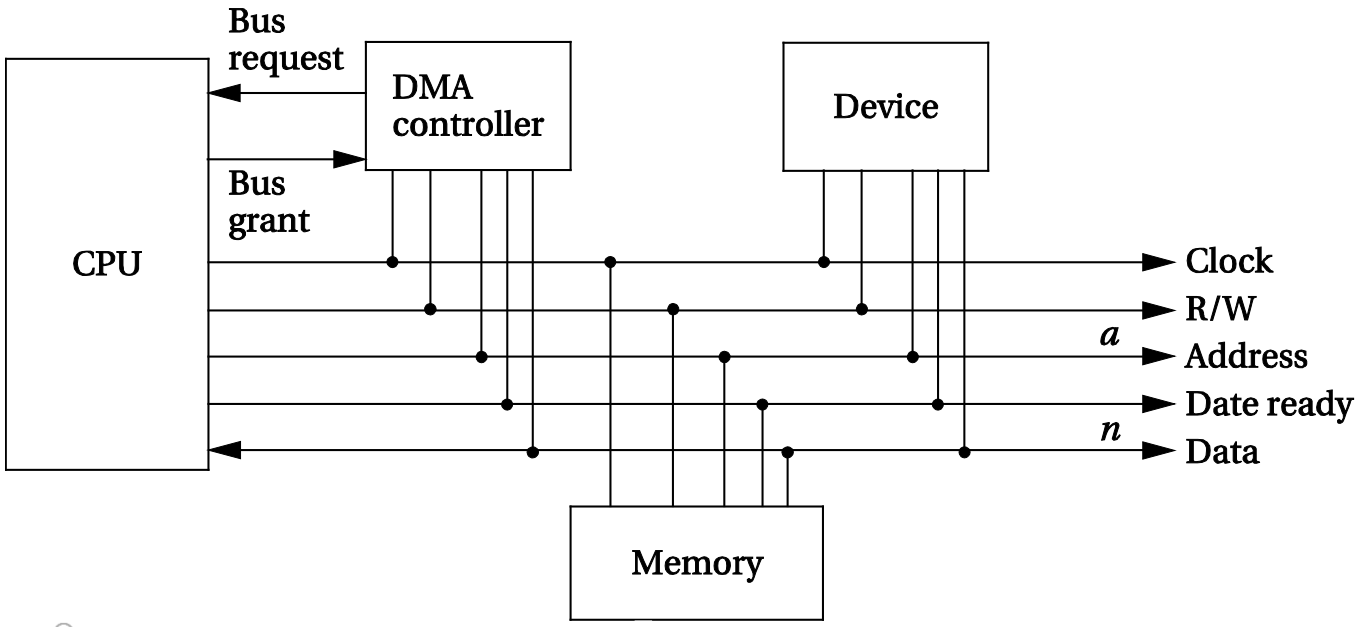
- Polling and interrupt-driven I/O
  - CPU transfers data between memory and I/O data registers
  - Time consuming for high-speed devices
- Direct memory access (DMA)
  - OS provides starting address in memory
  - I/O controller transfers to/from memory autonomously
  - Controller interrupts on completion or error

Bus mastership

- *Bus master* controls operations on the bus.
  - CPU is default bus master.
- Other devices may request bus mastership.
  - Separate set of handshaking lines.
  - CPU can’t use bus when it is not master.
- Situations for multiple bus masters:
  - DMA data transfers
  - Multiple CPUs with shared memory
    - One CPU might be graphics/network processor

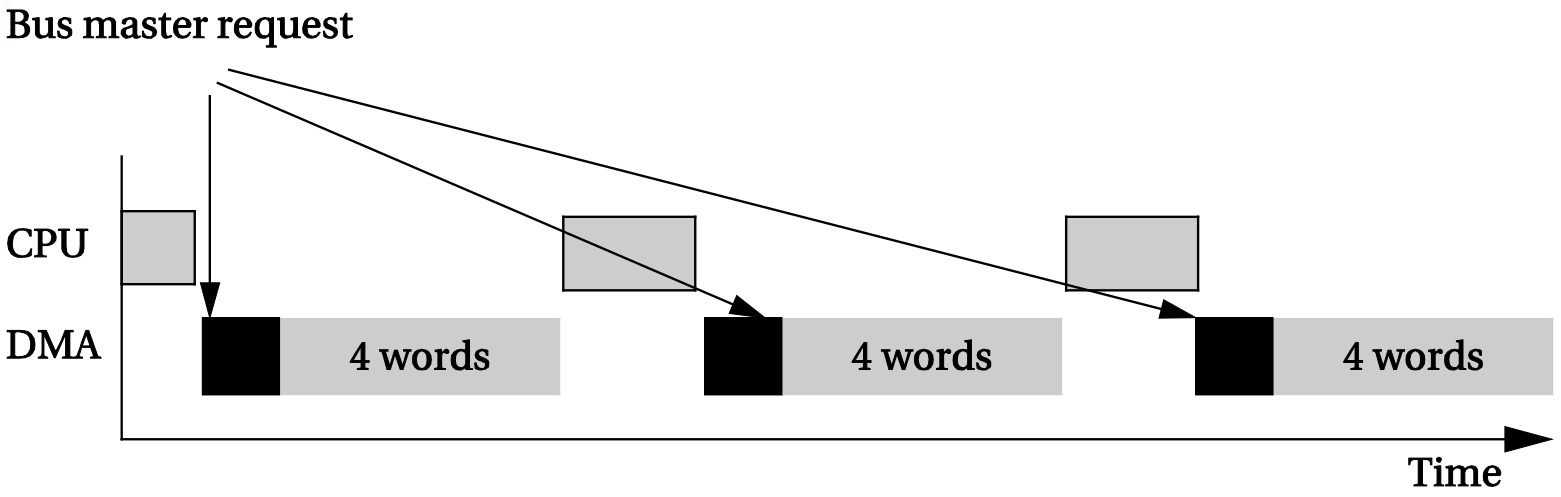
DMA organization

- *Direct memory access (DMA)* performs data transfers without executing instructions.
  - CPU configures transfer in DMA controller
  - DMA controller fetches & writes data.
- DMA controller is a separate unit.
  - CPU is the default bus master



DMA operation

- CPU sets DMA registers for start address, length.
- DMA controller has to acquire control of the bus
  - Bus request – ask for control of bus
  - Bus grant – acknowledgement that request granted
- Once DMA is bus master, it transfers automatically.
  - May run continuously until complete.
  - May use every n^{th} bus cycle.
  - CPU cannot use bus while DMA controller is master



DMA/Cache Interaction

- If DMA writes to a memory block that is cached
  - Cached copy becomes stale
- If write-back cache has dirty block, and DMA reads memory block
  - Reads stale data
- Need to ensure cache coherence
  - Flush blocks from cache if they will be used for DMA
  - Or use non-cacheable memory locations for I/O

DMA/VM Interaction

- OS uses virtual addresses for memory
  - DMA blocks may not be contiguous in physical memory
- Should DMA use virtual addresses?
  - Would require controller to do translation
- If DMA uses physical addresses
  - May need to break transfers into page-sized chunks
  - Or chain multiple transfers
  - Or allocate contiguous physical pages for DMA

Measuring I/O Performance

- I/O performance depends on
  - Hardware: CPU, memory, controllers, buses
  - Software: operating system, database management system, application
  - Workload: request rates and patterns
- I/O system design can trade-off between response time and throughput
  - Measurements of throughput often done with constrained response-time

Transaction Processing Benchmarks

- Transactions
  - Small data accesses to a DBMS
  - Interested in I/O rate, not data rate
- Measure throughput
  - Subject to response time limits and failure handling
  - ACID (Atomicity, Consistency, Isolation, Durability)
  - Overall cost per transaction
- Transaction Processing Council (TPC) benchmarks (www.tcp.org)
  - TPC-APP: B2B application server and web services
  - TCP-C: on-line order entry environment
  - TCP-E: on-line transaction processing for brokerage firm
  - TPC-H: decision support — business oriented ad-hoc queries

File System & Web Benchmarks

- SPEC System File System (SFS)
  - Synthetic workload for NFS server, based on monitoring real systems
  - Results
    - Throughput (operations/sec)
    - Response time (average ms/operation)
- SPEC Web Server benchmark
  - Measures simultaneous user sessions, subject to required throughput/session
  - Three workloads: Banking, Ecommerce, and Support

I/O vs. CPU Performance

- Amdahl's Law
  - Don't neglect I/O performance as parallelism increases compute performance
- Example
  - Benchmark takes 90s CPU time, 10s I/O time
  - Double the number of CPUs/2 years
    - I/O unchanged

Year	CPU time	I/O time	Elapsed time	% I/O time
now	90s	10s	100s	10%
+2	45s	10s	55s	18%
+4	23s	10s	33s	31%
+6	11s	10s	21s	47%

I/O System Design

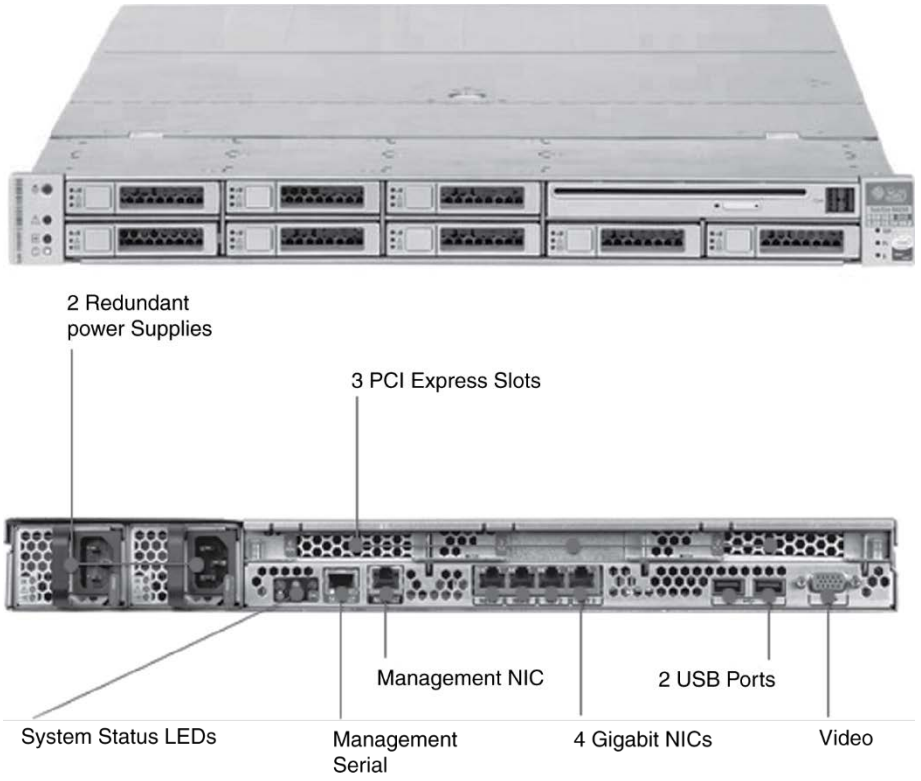
- Satisfying latency requirements
  - For time-critical operations
  - If system is unloaded
    - Add up latency of components
- Maximizing throughput
  - Find “weakest link” (lowest-bandwidth component)
  - Configure to operate at its maximum bandwidth
  - Balance remaining components in the system
- If system is loaded, simple analysis is insufficient
  - Need to use queuing models or simulation

Server Computers

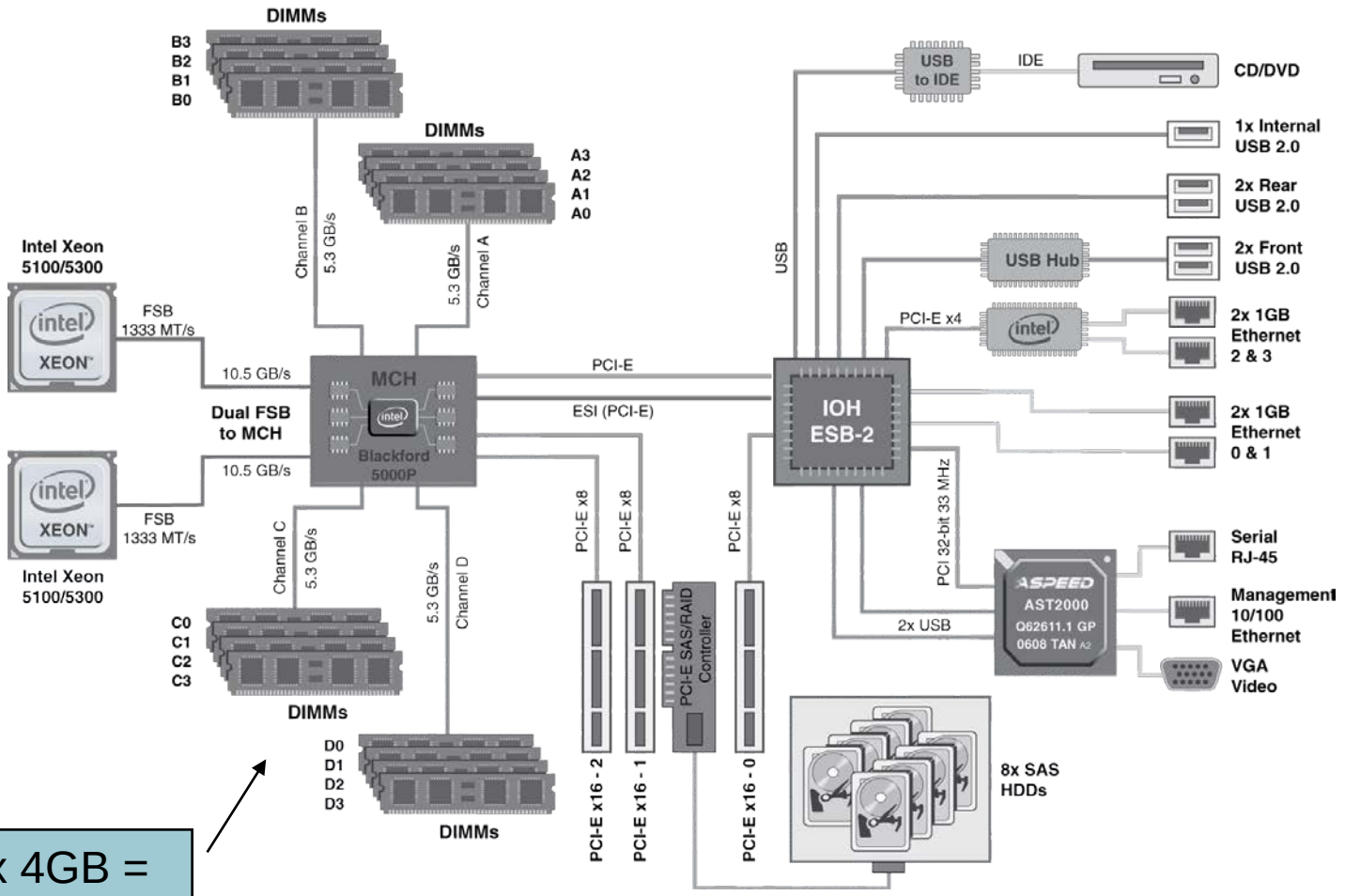
- Applications are increasingly run on servers
  - Web search, office apps, virtual worlds, …
- Requires large data center servers
  - Multiple processors, networks connections, massive storage
  - Space and power constraints
- Server equipment built for 19” racks
  - Multiples of 1.75” (1U) high

Rack-Mounted Servers

Sun Fire x4150 1U server



Sun Fire x4150 1U server



4 cores each

16 x 4GB = 64GB DRAM

I/O System Design Example

- What I/O rate can be sustained?
  - For random reads, and for sequential reads
- Given a Sun Fire x4150 system with
  - Workload: 64KB disk reads
    - Each I/O op requires 200,000 user-code instructions and 100,000 OS instructions
  - Each CPU: 10^9 instructions/sec
  - FSB: 10.6 GB/sec peak
  - DRAM DDR2 667MHz: 5.336 GB/sec
  - PCI-E 8× bus: 8 × 250MB/sec = 2GB/sec
  - Disks: 15,000 rpm, 2.9ms avg. seek time, 112MB/sec transfer rate

Design Example (cont)

- I/O rate for CPUs
  - Per core: 10^9/(100,000 + 200,000) = 3,333
  - 8 cores: 26,667 ops/sec
- Random reads, I/O rate for disks
  - Assume actual seek time is average/4
  - Time/op = seek + latency + transfer
    = 2.9ms/4 + 4ms/2 + 64KB/(112MB/s) = 3.3ms
  - 303 ops/sec per disk, 2424 ops/sec for 8 disks
- Sequential reads
  - 112MB/s / 64KB = 1750 ops/sec per disk
  - 14,000 ops/sec for 8 disks

Design Example (cont)

- PCI-E I/O rate (RAID -> North Bridge)
  - 2GB/sec / 64KB = 31,250 ops/sec
- DRAM I/O rate (MCB -> DRAM)
  - 5.336 GB/sec / 64KB = 83,375 ops/sec
- FSB I/O rate (North Bridge -> CPU)
  - Assume we can sustain half the peak rate
  - 5.3 GB/sec / 64KB = 81,540 ops/sec per FSB
  - 163,080 ops/sec for 2 FSBs
- Weakest link: disks
  - 2424 ops/sec random, 14,000 ops/sec sequential
  - Other components have ample headroom to accommodate these rates

Pitfall: Peak Performance

- Peak I/O rates are nearly impossible to achieve
  - Usually, some other system component limits performance
  - E.g., transfers to memory over a bus
    - Collision with DRAM refresh
    - Arbitration contention with other bus masters
  - E.g., PCI bus: peak bandwidth ~133 MB/sec
    - In practice, max 80MB/sec sustainable

Pitfall: Offloading to I/O Processors

- Overhead of managing I/O processor request may dominate
  - Quicker to do small operation on the CPU
  - But I/O architecture may prevent that
- I/O processor may be slower
  - Since it's supposed to be simpler
- Making it faster makes it into a major system component
  - Might need its own coprocessors!

Concluding Remarks

- I/O performance measures
  - Throughput, response time
  - Dependability and cost also important
- Buses used to connect CPU, memory, I/O controllers
  - Polling, interrupts, DMA
- I/O benchmarks
  - TPC, SPECSFS, SPECWeb
- RAID
  - Improves performance and dependability