ELEC3020: Lecture 6-1

Buses

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

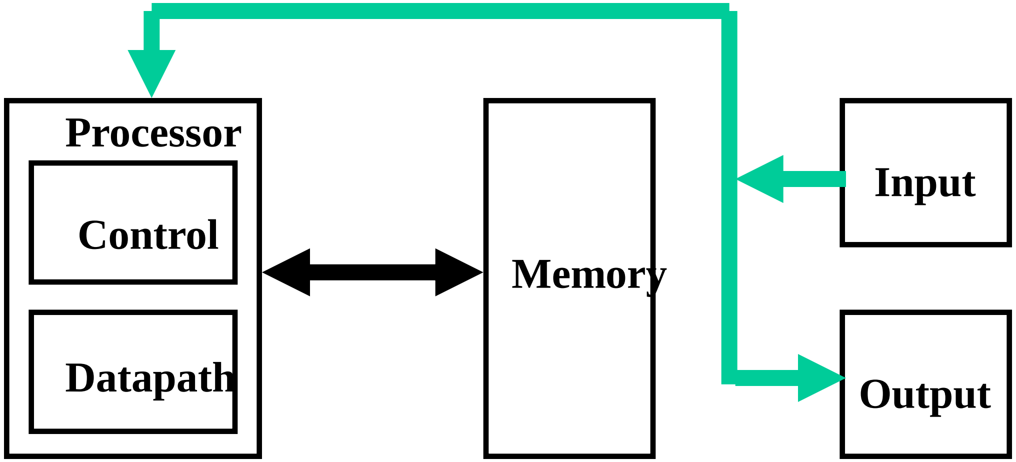
What is a bus?

- Slow vehicle that many people ride together
  - well, true...
- A bunch of wires...

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

A Bus is:

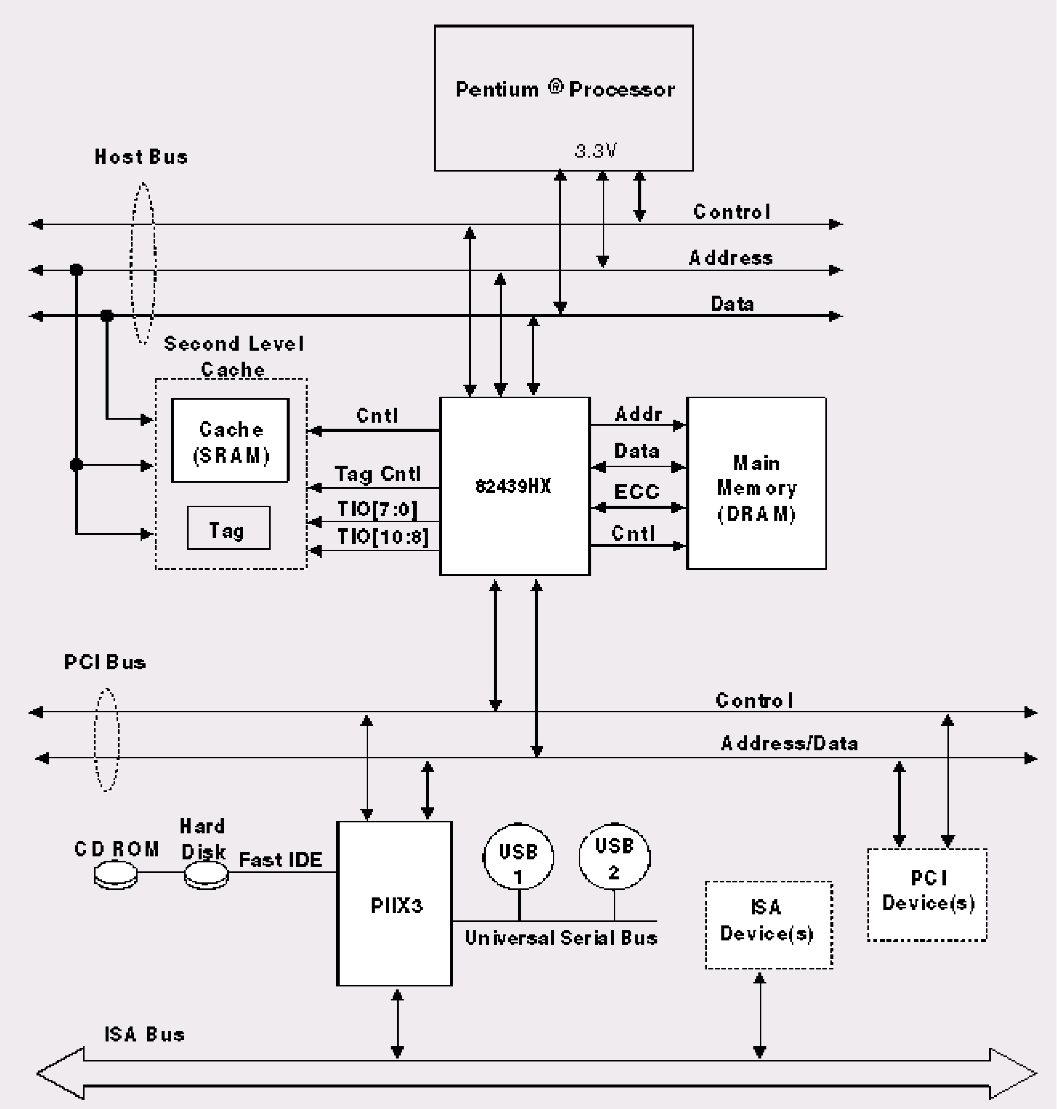
- shared communication link
- single set of wires used to connect multiple subsystems



- A Bus is also a fundamental tool for composing large, complex systems
  – systematic means of abstraction

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Early Pentium System Organization



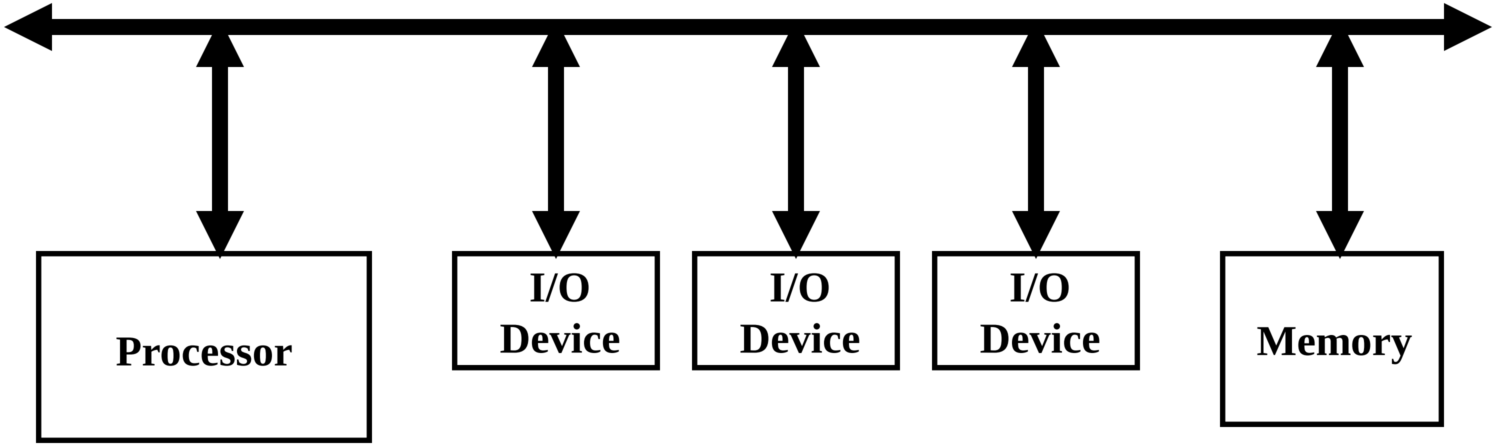
Processor/Memory Bus

PCI Bus

I/O Busses

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

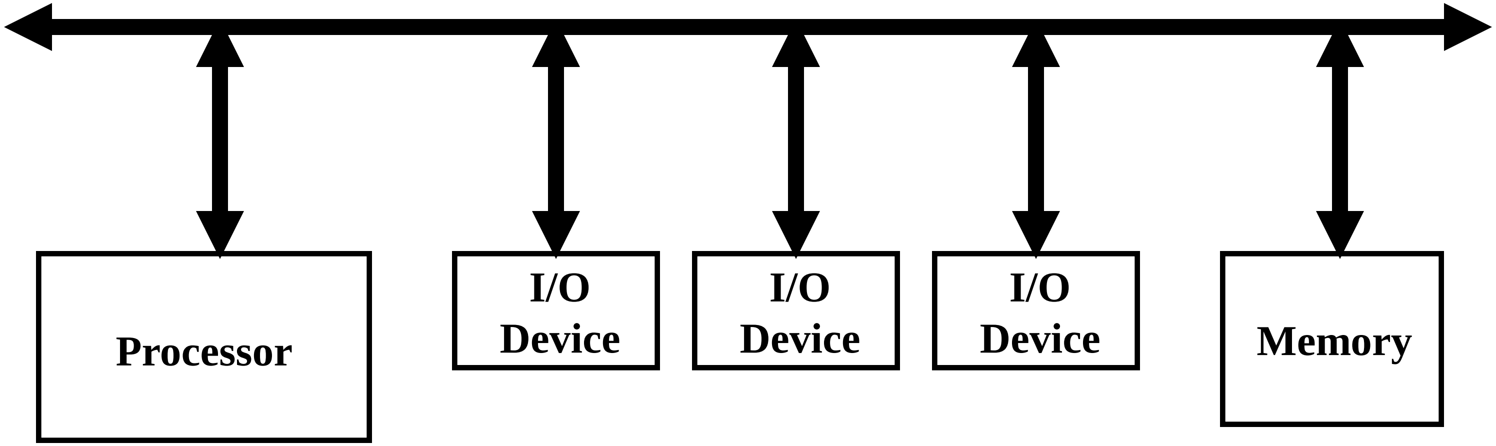
Advantages of Buses



- Versatility:
  - New devices can be added easily
  - Peripherals can be moved between computer systems that use the same bus standard
- Low Cost:
  - A single set of wires is shared in multiple ways
- Manage complexity by partitioning the design

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Disadvantage of Buses



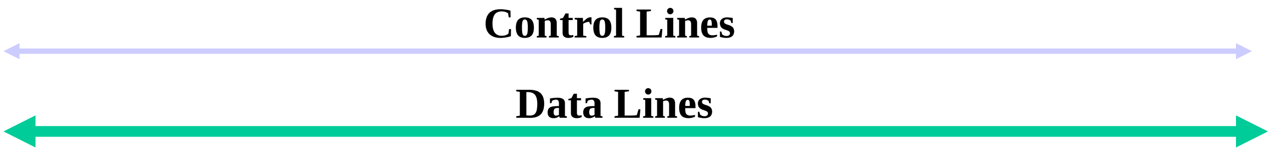
- It creates a communication bottleneck
  - The bandwidth of that bus can limit the maximum I/O throughput
- The maximum bus speed is largely limited by:
  - The length of the bus
  - The number of devices on the bus
  - The need to support a range of devices with:
    - Widely varying latencies
    - Widely varying data transfer rates

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Electrical issues

- Buses are slow compared to point-to-point connections as:
  - The various drives and receivers add capacitive loading, slowing the speed of signals ($\propto \sqrt{\frac{1}{lc}}$)
  - The bus is probably poorly terminated and needs *settling time*.
  - Timing delay skew between lines forces slower clocks.
- Buses are power hungry as:
  - Signals must be large enough to overcome bus noise and poor termination.
  - The capacitive loading lowers the bus impedance ($\propto \sqrt{\frac{l}{c}}$), increasing drive currents.
- A single signal, with clock recovery, cannot suffer from skew but might have problems with *dispersion*.

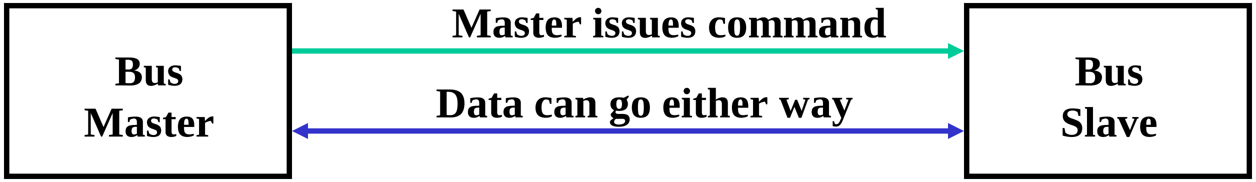
The General Organization of a Bus



- Control lines:
  - Signal requests and acknowledgments
  - Indicate what type of information is on the data lines
- Data lines carry information between the source and the destination:
  - Data and Addresses
  - Complex commands

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Master versus Slave



- A bus transaction includes two parts:
  - Issuing the command (and address) – request
  - Transferring the data – action
- Master is the one who starts the bus transaction by:
  - issuing the command (and address)
- Slave is the one who responds to the address by:
  - Sending data to the master if the master ask for data
  - Receiving data from the master if the master wants to send data

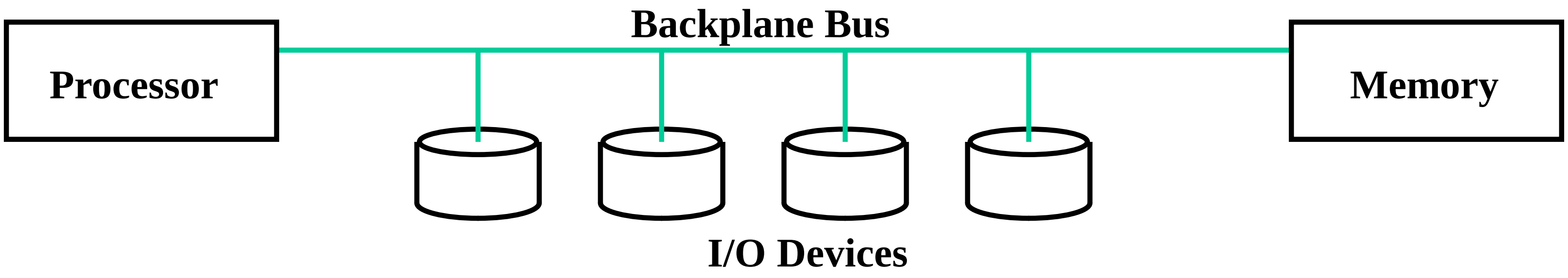
With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Types of Buses

- Processor-Memory Bus (design specific)
  - Short and high speed
  - Only need to match the memory system
    - Maximize memory-to-processor bandwidth
  - Connects directly to the processor
  - Optimized for cache block transfers
- I/O Bus (industry standard)
  - Usually is lengthy and slower
  - Need to match a wide range of I/O devices
  - Connects to the processor-memory bus or backplane bus
- Backplane Bus (standard or proprietary)
  - Backplane: an interconnection structure within the chassis
  - Allow processors, memory, and I/O devices to coexist
  - Cost advantage: one bus for all components

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

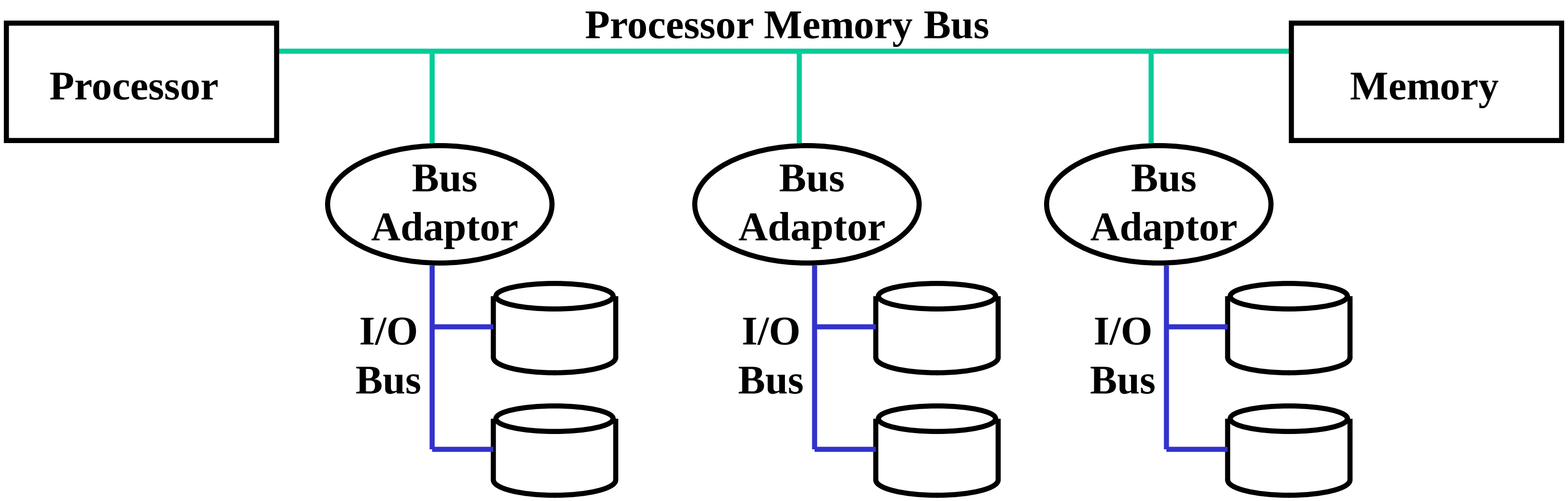
One Bus System: Backplane Bus



- A single bus (the backplane bus) is used for:
  - – Processor to memory communication
  - – Communication between I/O devices and memory
- Advantages: Simple and low cost
- Disadvantages: slow and the bus can become a major bottleneck
- Examples: S100, IBM PC – AT

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

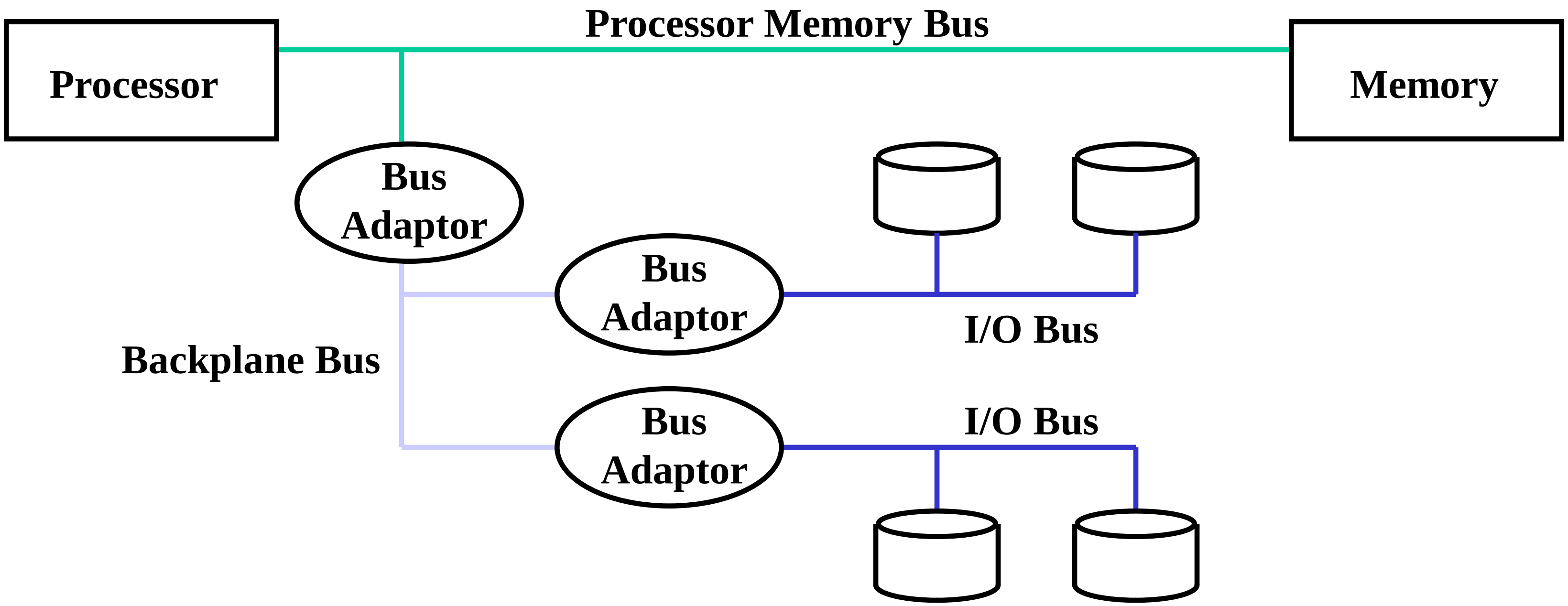
Two-Bus System



- I/O buses tap into the processor-memory bus via bus adaptors:
  - Processor-memory bus: mainly for processor-memory traffic
  - I/O buses: provide expansion slots for I/O devices
- Apple Macintosh-II
  - NuBus: Processor, memory, and a few selected I/O devices
  - SCCI Bus: the rest of the I/O devices

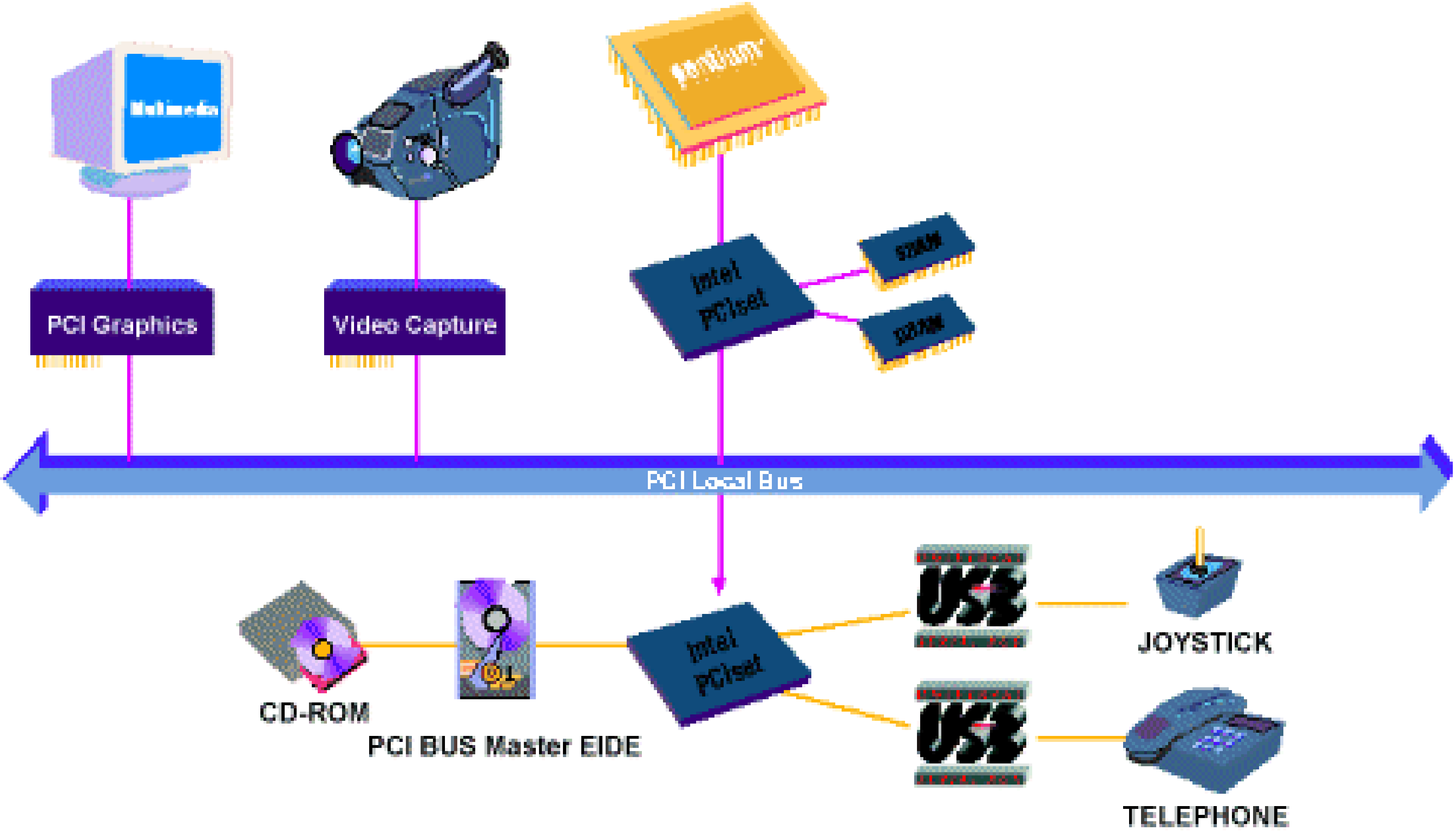
With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Three-Bus System



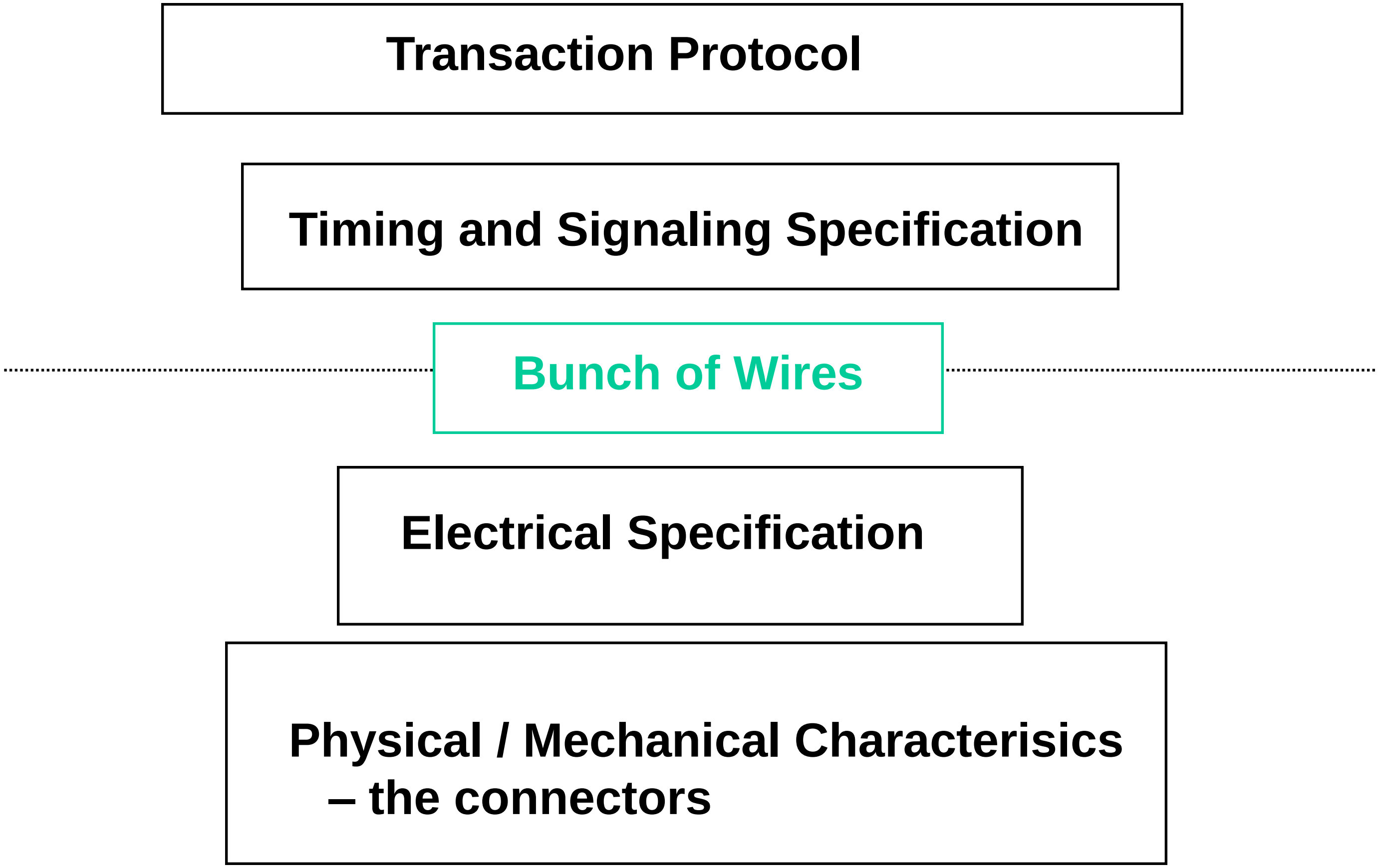
- A small number of backplane buses tap into the processor-memory bus
  - Processor-memory bus is used for processor memory traffic
  - I/O buses are connected to the backplane bus
- Advantage: loading on the processor bus is greatly reduced

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others



With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

What defines a bus?



Transaction Protocol

Timing and Signaling Specification

Bunch of Wires

Electrical Specification

Physical / Mechanical Characterisics
– the connectors

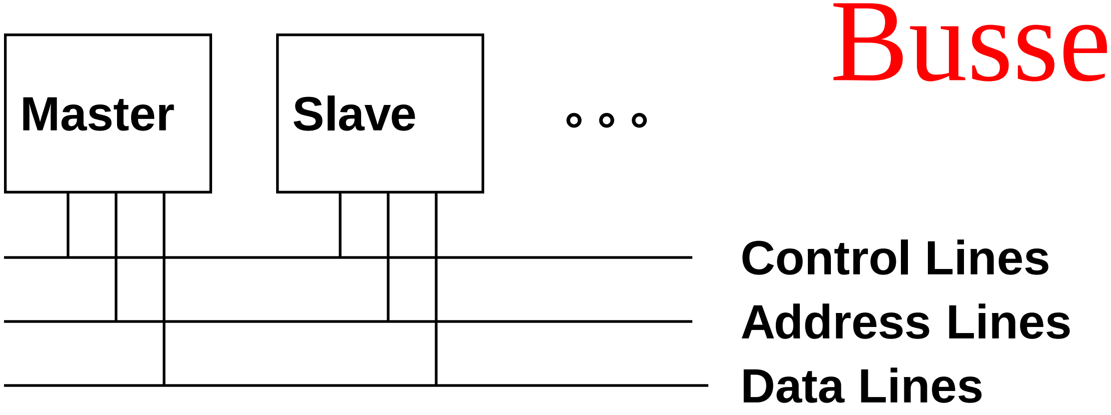
With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Synchronous and Asynchronous Bus

- Synchronous Bus:
  - Includes a clock in the control lines
  - A fixed protocol for communication that is relative to the clock
  - Advantage: involves very little logic and can run very fast
  - Disadvantages:
    - Every device on the bus must run at the same clock rate
    - To avoid clock skew, they cannot be long if they are fast
- Asynchronous Bus:
  - It is not clocked
  - It can accommodate a wide range of devices
  - It can be lengthened without worrying about clock skew
  - It requires a handshaking protocol

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Busses so far



Bus Master: has ability to control the bus, initiates transaction

Bus Slave: module activated by the transaction

Bus Communication Protocol: specification of sequence of events and timing requirements in transferring information.

Asynchronous Bus Transfers: control lines (req, ack) serve to orchestrate sequencing.

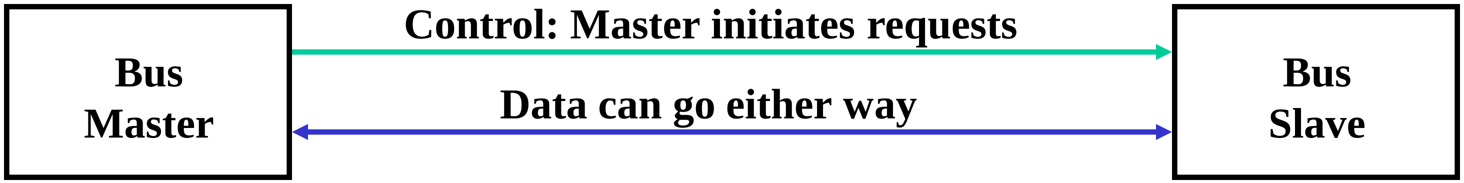
Synchronous Bus Transfers: sequence relative to common clock.

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Bus Transaction

- Arbitration
- Request
- Action

Arbitration



- One of the most important issues in bus design:
  - How is the bus reserved by a devices that wishes to use it?
- Chaos is avoided by a master-slave arrangement:
  - Only the bus master can control access to the bus:

    It initiates and controls all bus requests
  - A slave responds to read and write requests
- The simplest system:
  - Processor is the only bus master
  - All bus requests must be controlled by the processor
  - Major drawback: the processor is involved in every transaction

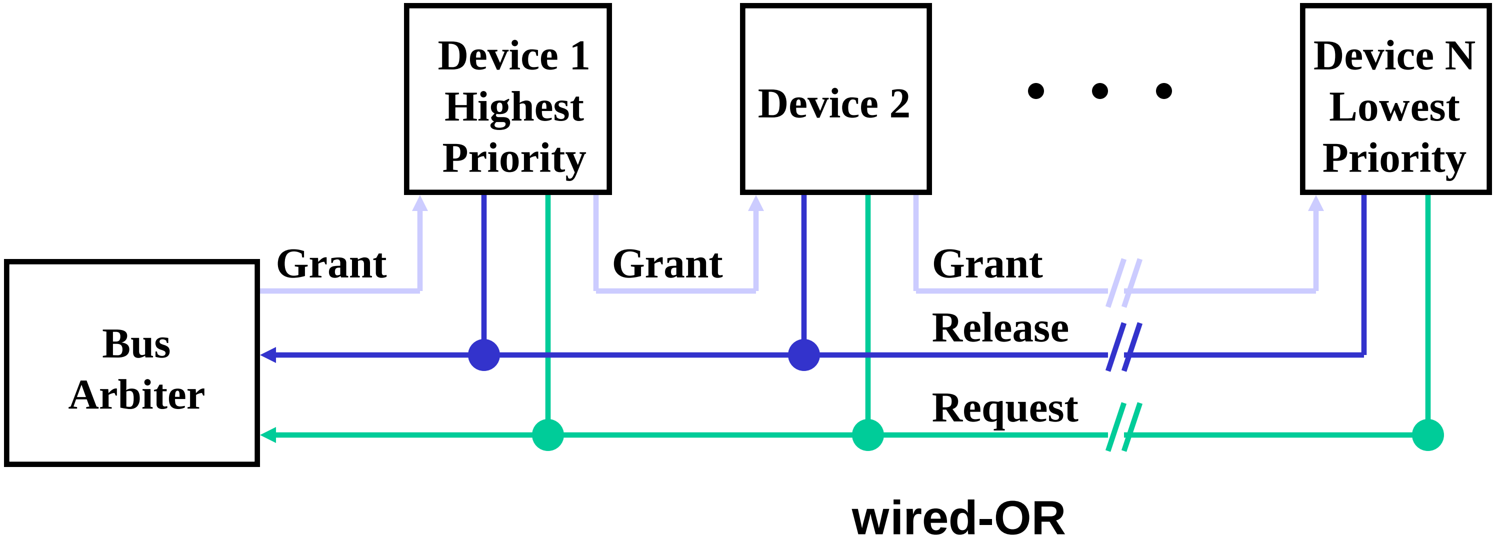
With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Multiple Bus Masters: Why Arbitration

- Bus arbitration scheme:
  - A bus master wanting to use the bus asserts the bus request
  - A bus master cannot use the bus until its request is granted
  - A bus master must signal to the arbiter after finish using the bus
- Bus arbitration schemes usually try to balance two factors:
  - Bus priority: the highest priority device should be serviced first
  - Fairness: Even the lowest priority device should never be completely locked out from the bus
- Bus arbitration schemes can be divided into four broad classes:
  - Daisy chain arbitration: single device with all request lines.
  - Centralized, parallel arbitration: see next-next slide
  - Distributed arbitration by self-selection: each device wanting the bus places a code indicating its identity on the bus.
  - Distributed arbitration by collision detection: Ethernet uses this.

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

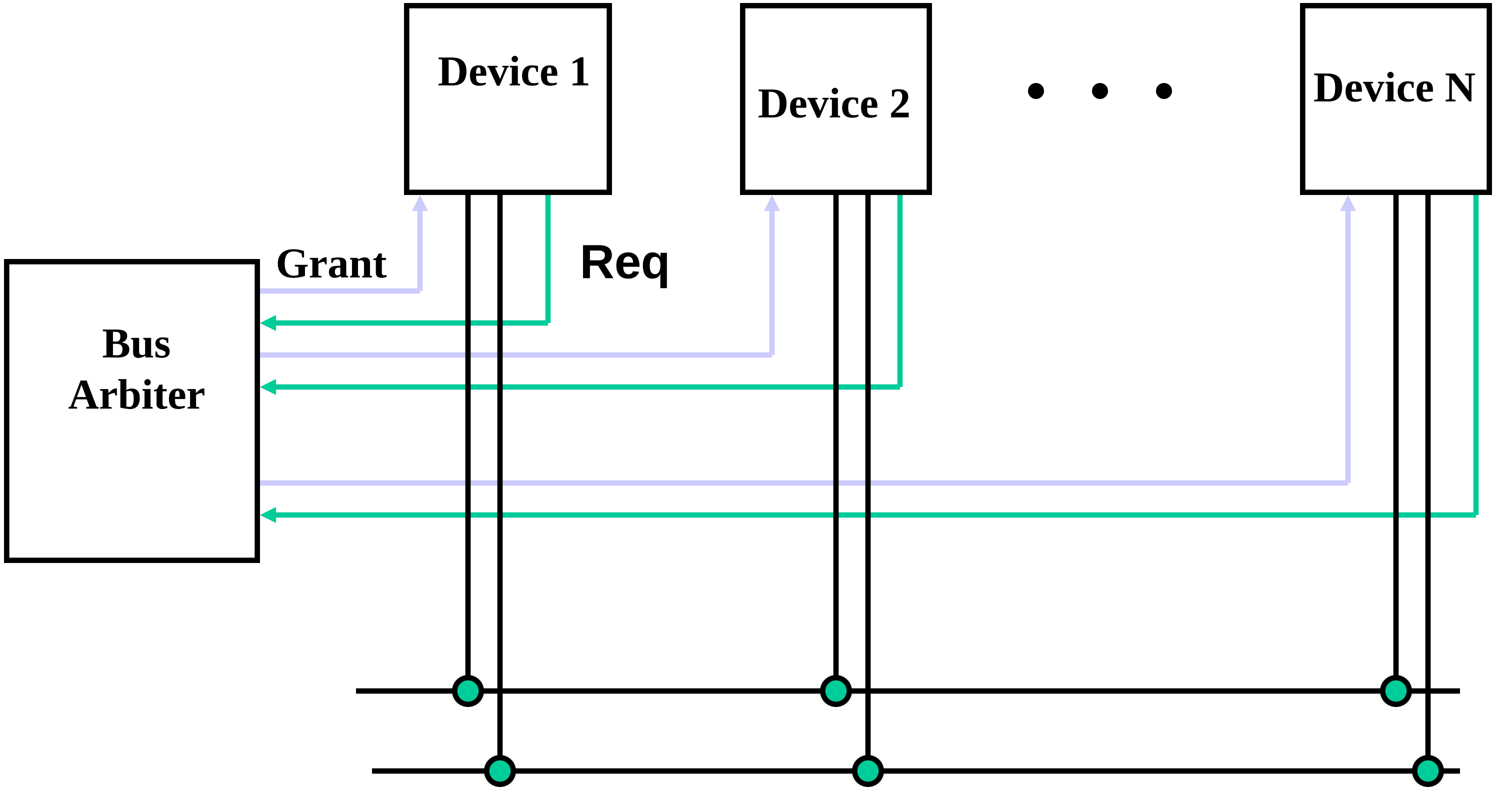
Daisy Chain Bus Arbitration



- Advantage: simple
- Disadvantages:
  - Cannot assure fairness:
    A low-priority device may be locked out indefinitely
  - The use of the daisy chain grant signal also limits the bus speed

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

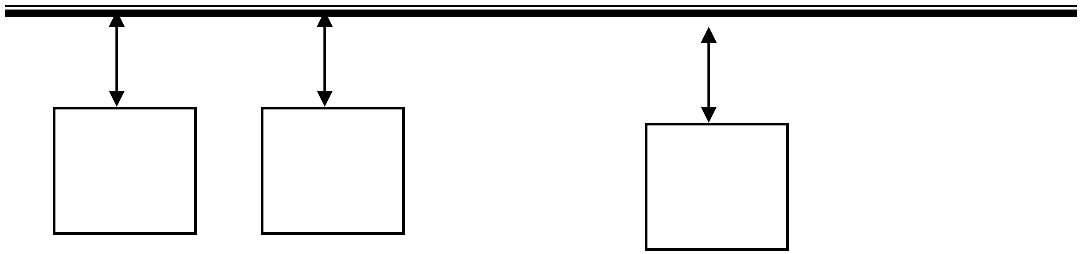
Centralized Parallel Arbitration



- Used in essentially all processor-memory busses and in high-speed I/O busses

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

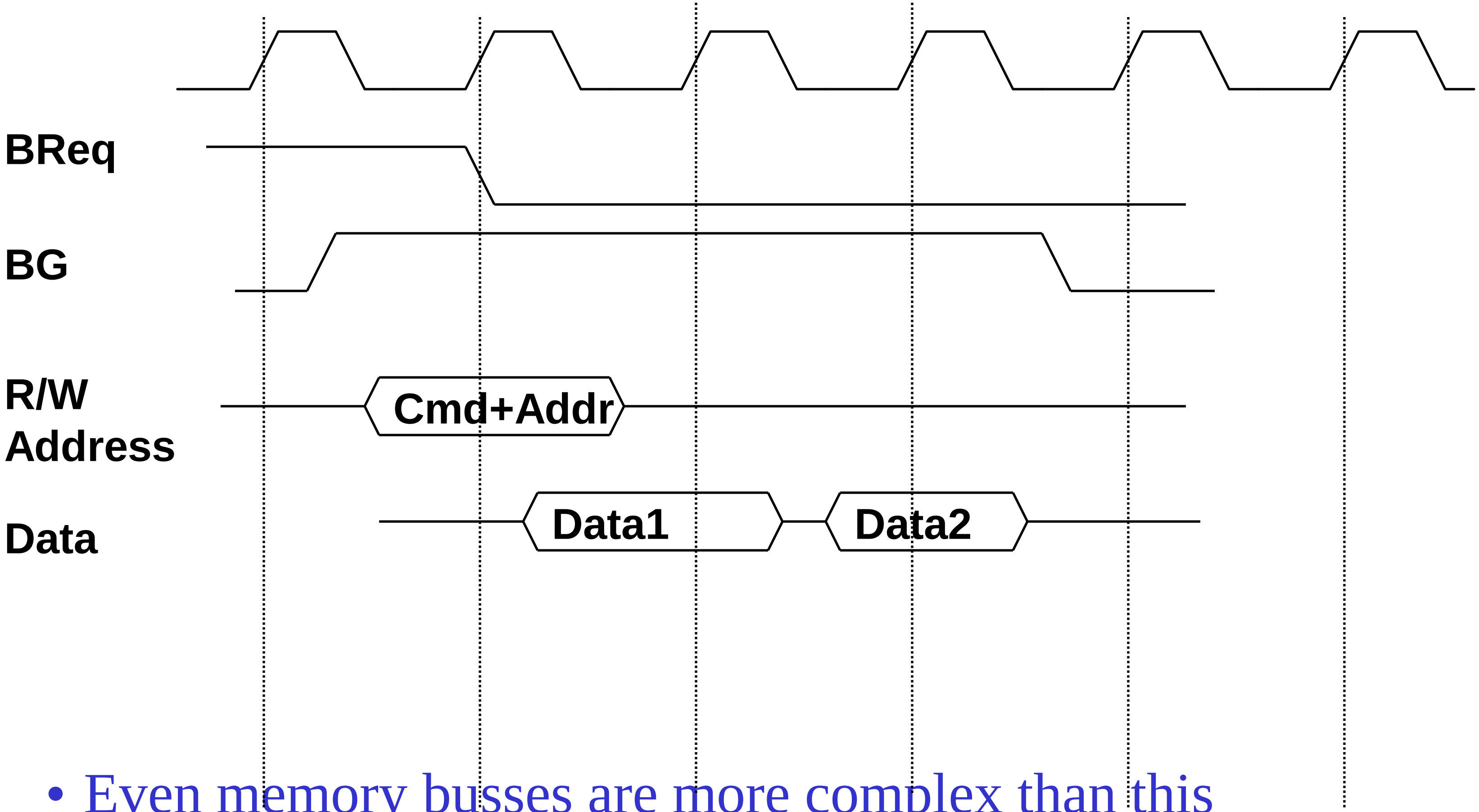
Simplest bus paradigm



- All agents operate syncronously
- All can source / sink data at same rate
- => simple protocol
  - just manage the source and target

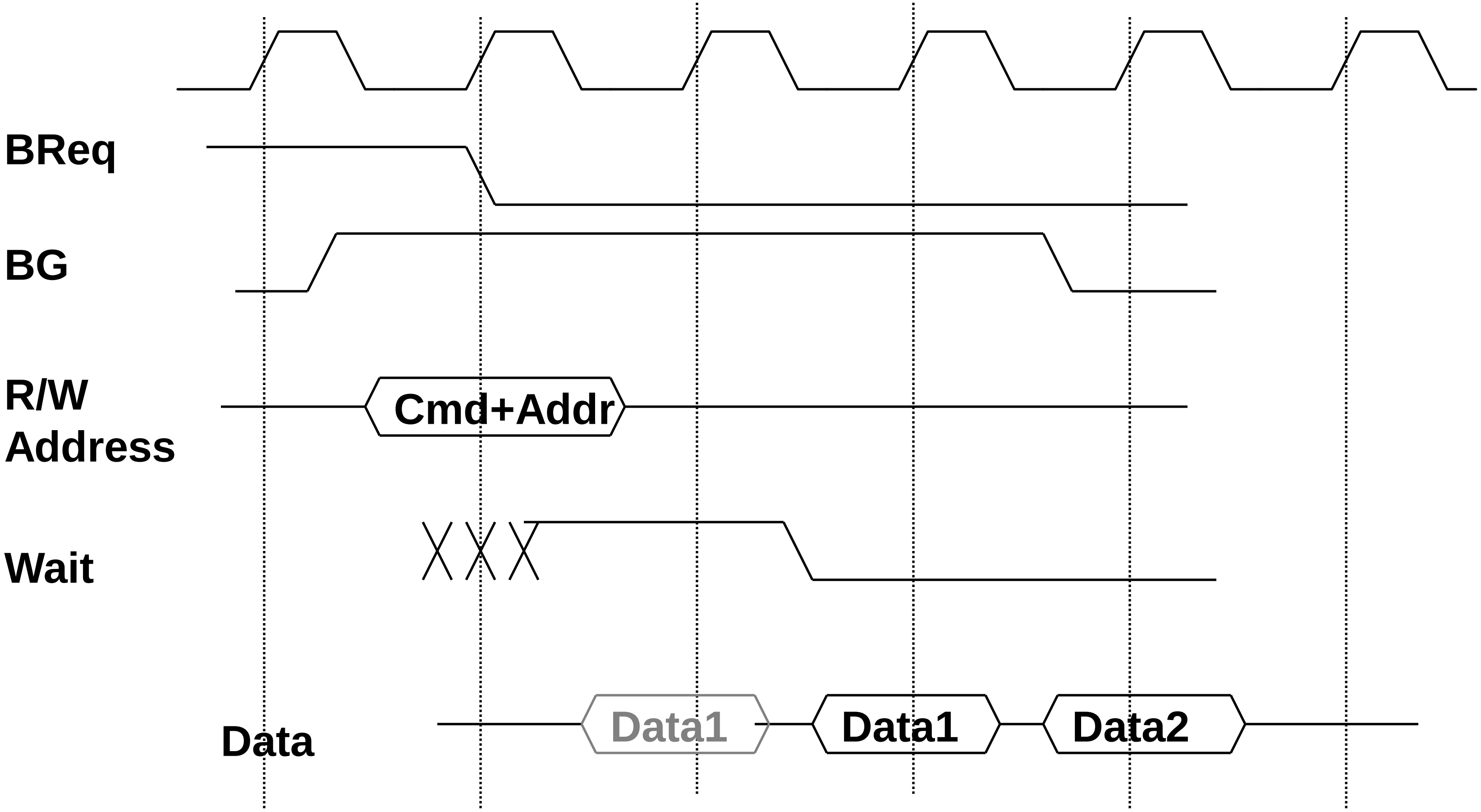
With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Simple Synchronous Protocol



- Even memory busses are more complex than this
  - memory (slave) may take time to respond
  - it need to control data rate

Typical Synchronous Protocol



- Slave indicates when it is prepared for data xfer
- Actual transfer goes at bus rate

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Increasing the Bus Bandwidth

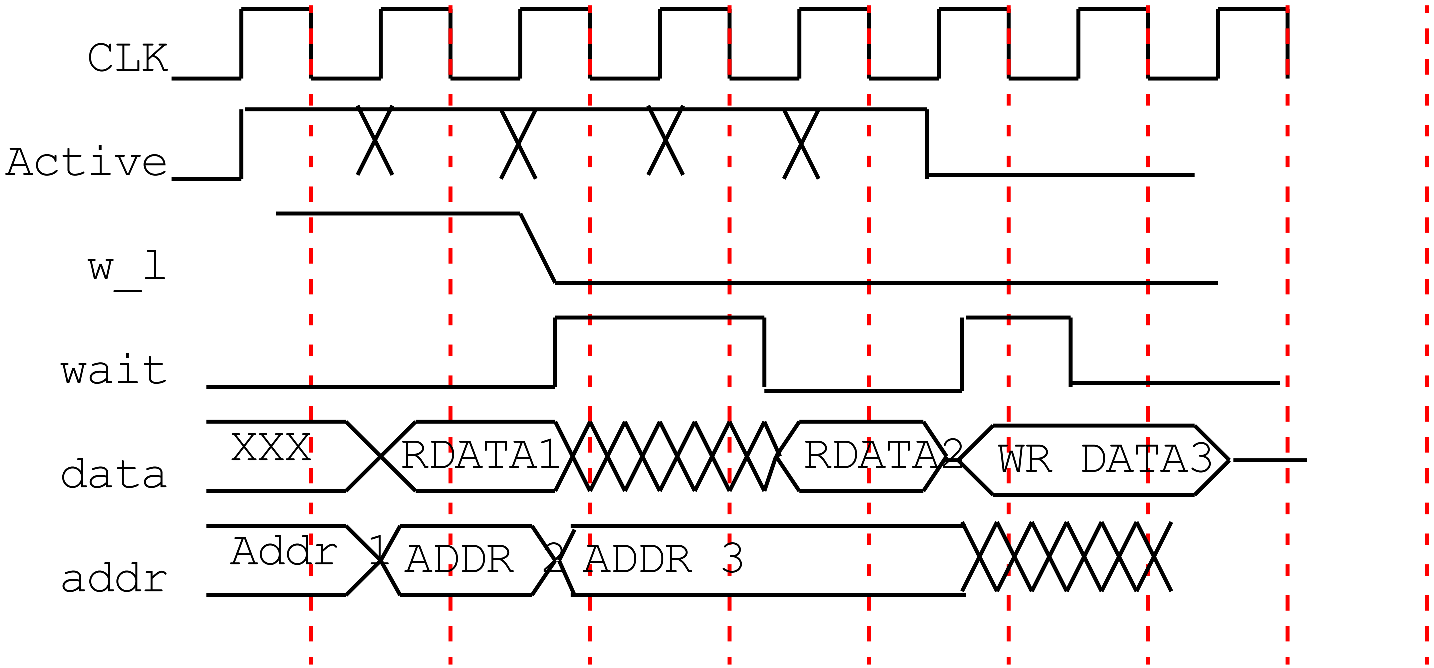
- Separate versus multiplexed address and data lines:
  - Address and data can be transmitted in one bus cycle if separate address and data lines are available
  - Cost: (a) more bus lines, (b) increased complexity
- Data bus width:
  - By increasing the width of the data bus, transfers of multiple words require fewer bus cycles
  - Example: SPARCstation 20's memory bus is 128 bit wide
  - Cost: more bus lines
- Block transfers:
  - Allow the bus to transfer multiple words in back-to-back bus cycles
  - Only one address needs to be sent at the beginning
  - The bus is not released until the last word is transferred
  - Cost: (a) increased complexity
    (b) decreased response time for request

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Pipelined Bus Protocols

Attempt to initiate next address phase during current data phase

Single master example from your lab (proc-to-cache)



With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Speeding up a Multimaster Bus

- Overlapped arbitration
  - perform arbitration for next transaction during current transaction
- Bus parking
  - master can holds onto bus and performs multiple transactions as long as no other master makes request
- Overlapped address / data phases (prev. slide)
  - requires one of the above techniques
- Split-phase (or packet switched) bus
  - completely separate address and data phases
  - arbitrate separately for each
  - address phase yield a tag which is matched with data phase
- ”All of the above” in modern memory busses

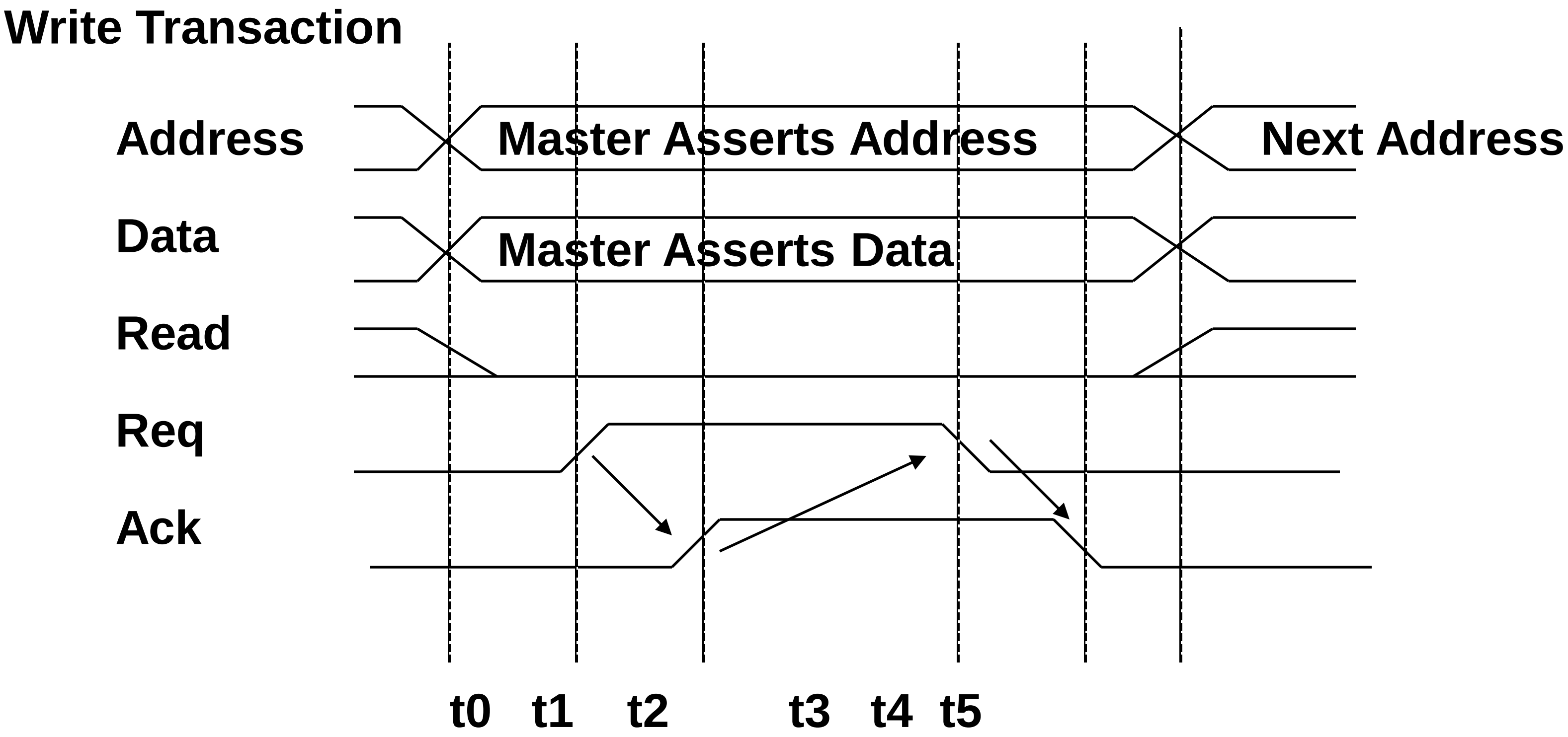
With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

The I/O Bus Problem

- Designed to support wide variety of devices
  - full set not know at design time
- Allow data rate match between arbitrary speed deviced
  - fast processor – slow I/O
  - slow processor – fast I/O

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

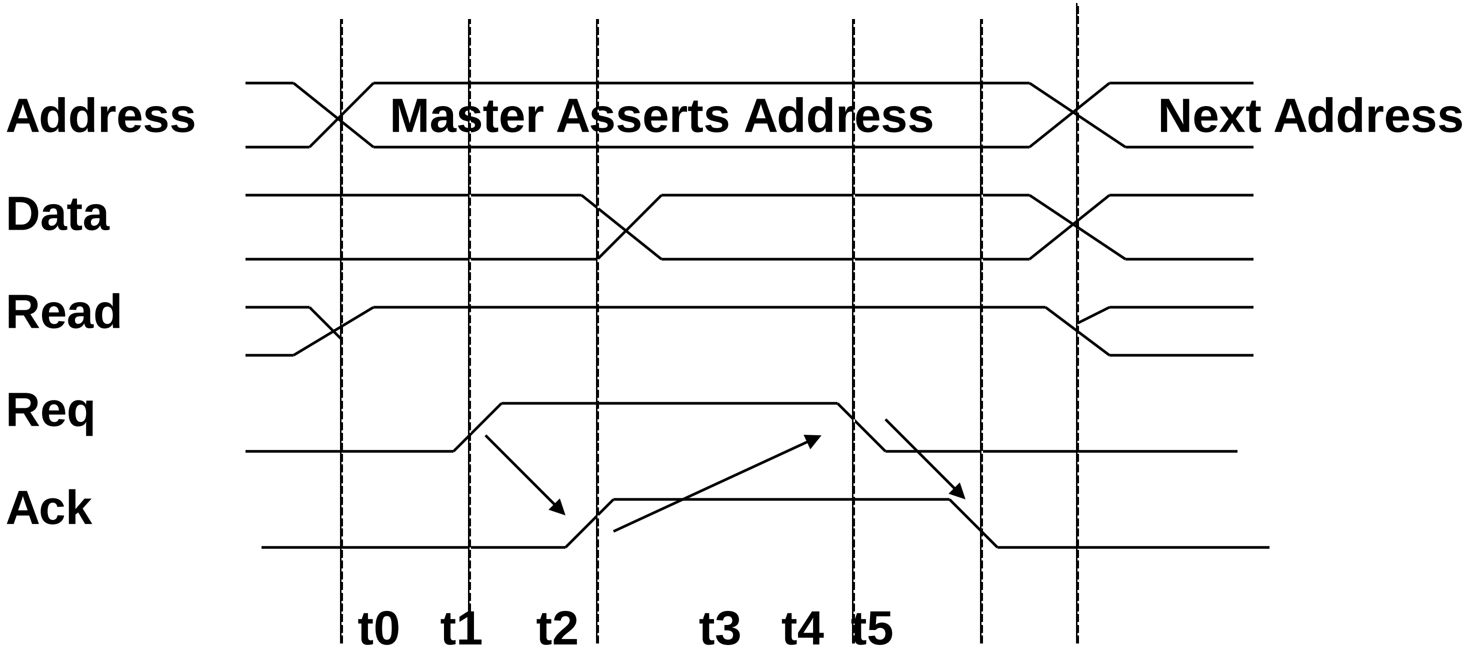
Asynchronous Handshake



- t0 : Master has obtained control and asserts address, direction, data
- Waits a specified amount of time for slaves to decode target
- t1: Master asserts request line
- t2: Slave asserts ack, indicating data received
- t3: Master releases req
- t4: Slave releases ack

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

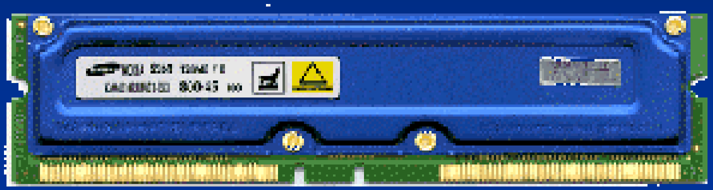
Read Transaction

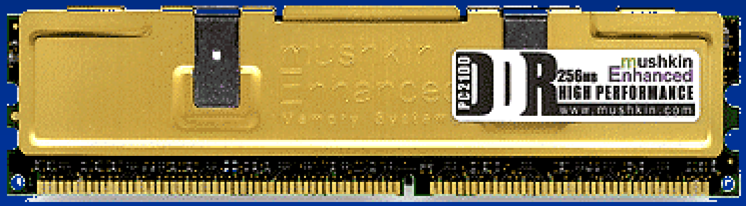


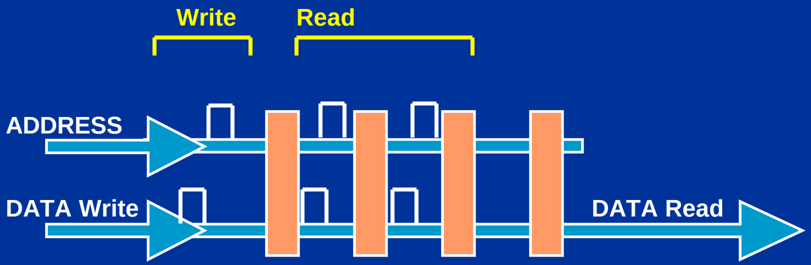
- t0 : Master has obtained control and asserts address, direction, data
- Waits a specified amount of time for slaves to decode target\
- t1: Master asserts request line
- t2: Slave asserts ack, indicating ready to transmit data
- t3: Master releases req, data received
- t4: Slave releases ack

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

RAMBUS DDR RAM



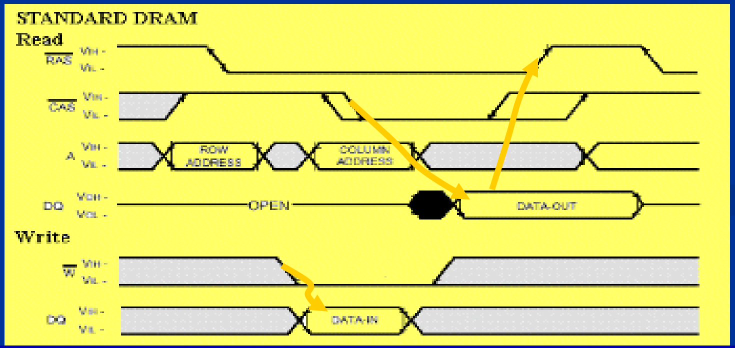


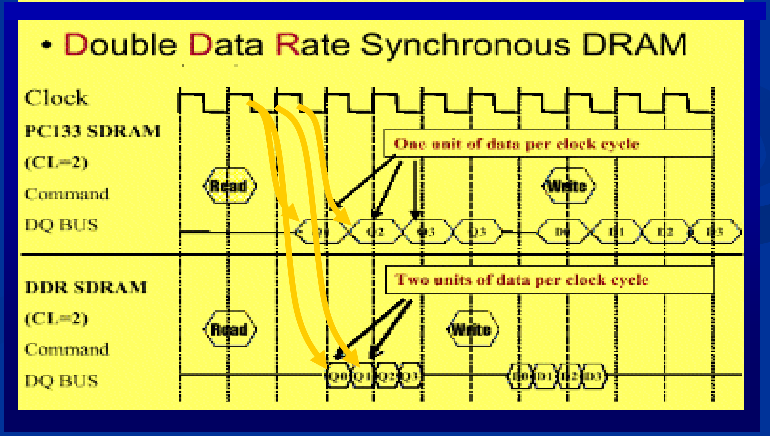


Works as a traveling wave line for addresses and Data

**Extremely difficult to build for wide memory !!
and such limited up to now to 16 bit.**

That’s why Rambus started with 8 bit wide memories





- Double Data Rate Synchronous DRAM

RAMBUS   DDR RAM

↔

	Clock/Data Rate	Memory Bus Width	Latency (row acces time)	Bandwidth
PC-100	100MHz/100MHz	64-bit	50 ns	800 MBytes/s
PC-133	133MHz/133MHz	64-bit	48 ns	1.064 GBytes/s
RAMBUS PC700	100MHz/100MHz	16-bit	75 ns	1.424 GBytes/s
RAMBUS PC800	100MHz/100MHz	16-bit	73 ns	1.6 GBytes/s
PC1600 DDR 200	100MHz/100MHz	64-bit	50 ns	1.6 GBytes/s
PC2100 DDR 266	100MHz/100MHz	64-bit	47 ns	2.128 GBytes/s
DDR II	100MHz/100MHz	64-bit	40 ns	3.2 GBytes/s

RAMBUS◆DDR SDRAM: The Future

RAMBUS

Technical

Samples in 130 nm have been produced.

Clock:	1066	MHz,
Data Rate:	2132	MHz
Bandwidth:	4,264	GBytes/s

Yes but it needs a chipset that can handle this.

No possibilities wide memories or for interleaving memory banks

DDR SDRAM

Technical

DDR 2400 is available

Clock:	266	MHz,
Data Rate:	532	MHz
Bandwidth:	4.254	GBytes/s

DDR 2700 is on JEDEC approval

Clock:	333	MHz,
Data Rate:	666	MHz
Bandwidth:	5.328	GBytes/s

Yes but it needs a chipset that can handle this.

Possibilities to make wide memories or to make interleaving memory banks

Original PCI, around 1993

Bus	PCI
Originator	Intel
Clock Rate (MHz)	33
Addressing	Physical
Data Sizes (bits)	8,16,24,32,64
Master	Multi
Arbitration	Central
32 bit read (MB/s)	33
Peak (MB/s)	111 (222)
Max Power (W)	25

High Speed I/O Bus

- Examples
  - graphics
  - fast networks
- Limited number of devices
- Data transfer bursts at full rate
- DMA transfers important
  - small controller spools stream of bytes to or from memory
- Either side may need to squelch transfer
  - buffers fill up

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

PCI Read/Write Transactions

- All signals sampled on rising edge
- Centralized Parallel Arbitration
  - overlapped with previous transaction
- All transfers are (unlimited) bursts
- Address phase starts by asserting FRAME#
- Next cycle "initiator" asserts cmd and address
- Data transfers happen on when
  - IRDY# asserted by master when ready to transfer data
  - TRDY# asserted by target when ready to transfer data
  - transfer when both asserted on rising edge
- FRAME# deasserted when master intends to complete only one more data transfer

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

PCI Read Transaction

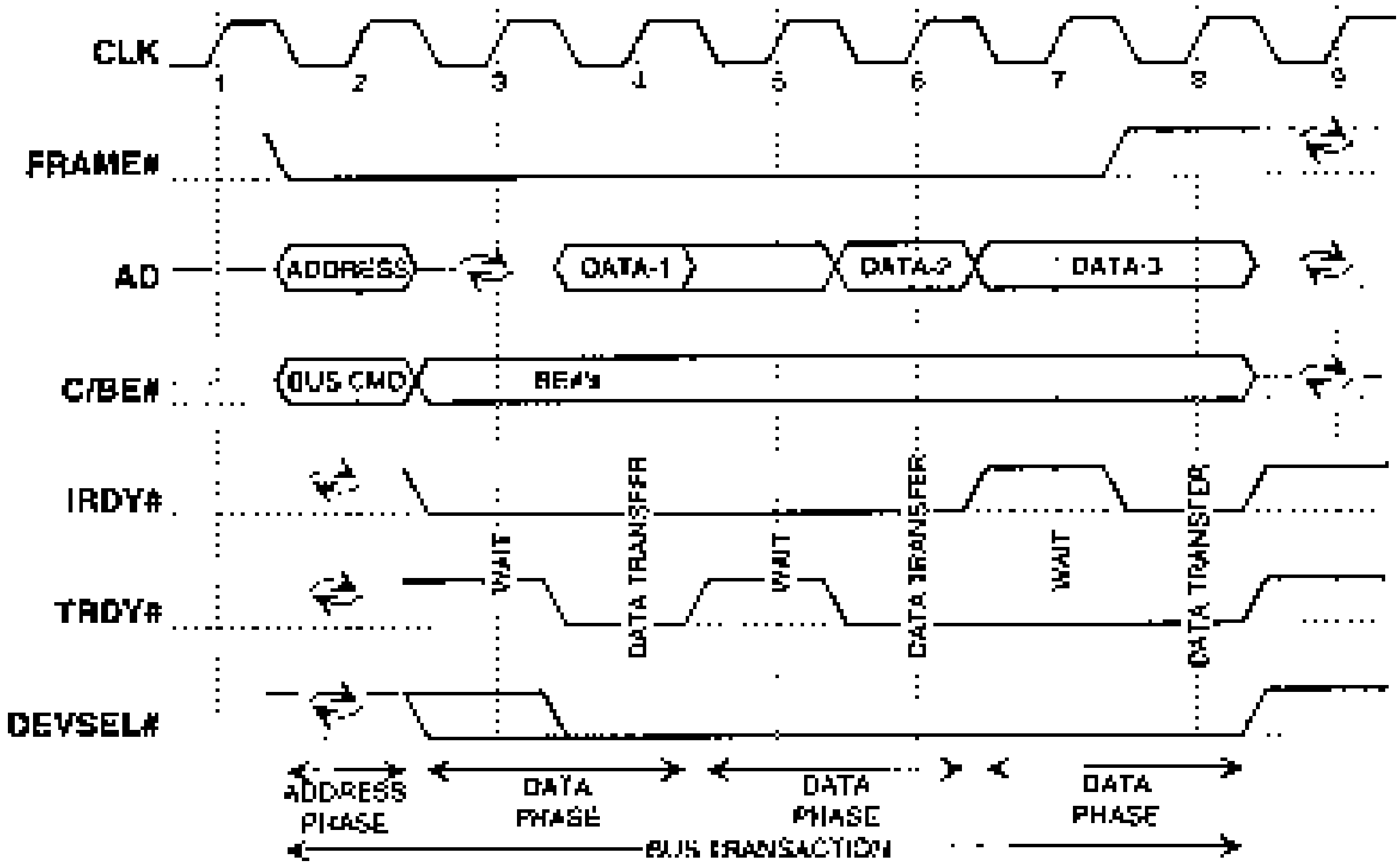


Figure 3-1: Basic Read Operation

– **Turn-around cycle on any signal driven by more than one agent**

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

PCI Write Transaction

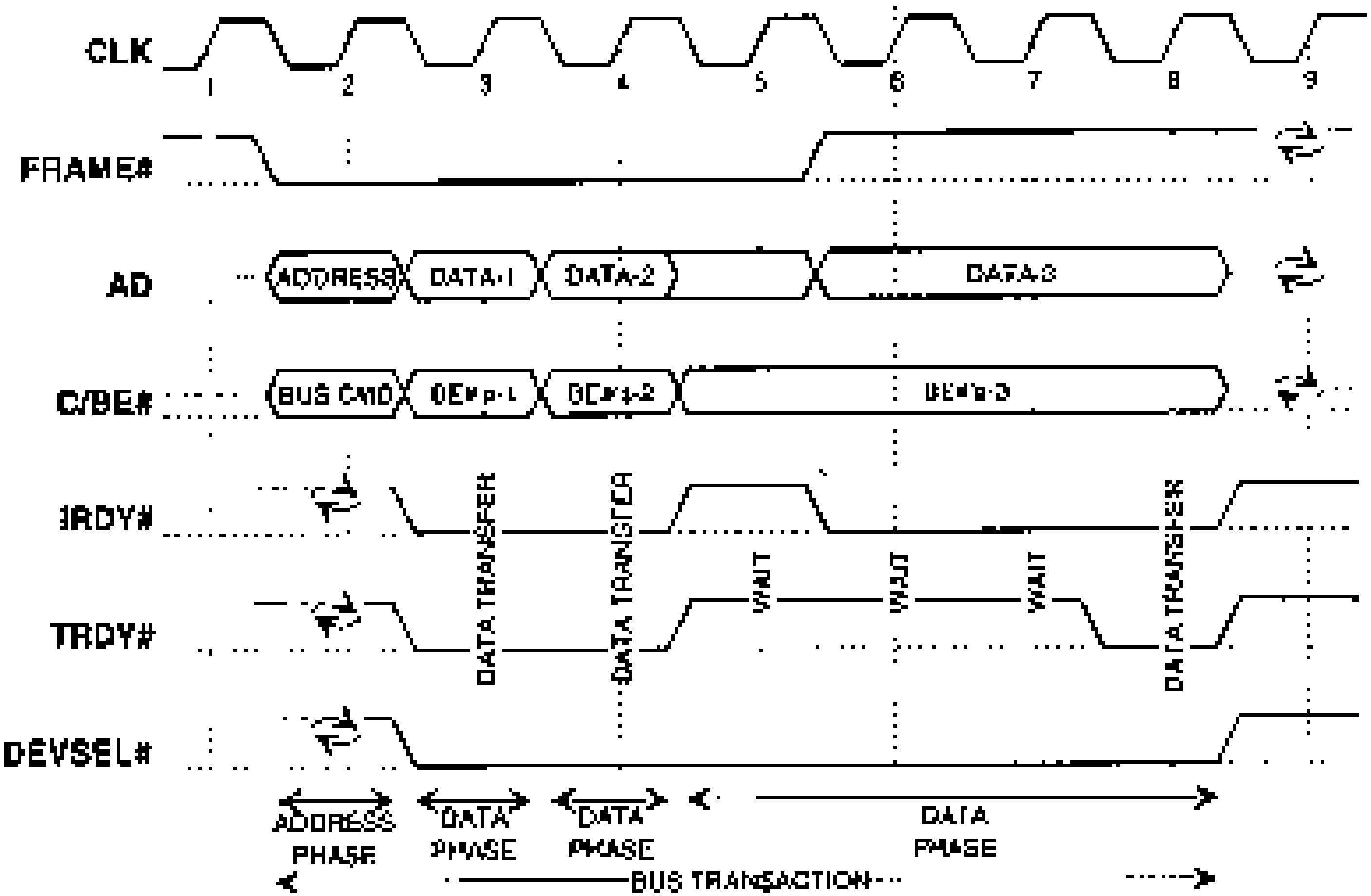


Figure 3-2: Basic Write Operation

PCI Optimizations

- Push bus efficiency toward 100% under common simple usage
    - like RISC
- Bus Parking
    - retain bus grant for previous master until another makes request
    - granted master can start next transfer without arbitration
- Arbitrary Burst length
    - intiator and target can exert flow control with xRDY
    - target can disconnect request with STOP (abort or retry)
    - master can disconnect by deasserting FRAME
    - arbiter can disconnect by deasserting GNT
- Delayed (pended, split-phase) transactions
    - free the bus after request to slow device

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Additional PCI Issues

- Interrupts: support for controlling I/O devices
- Cache coherency:
  - support for I/O and multiprocessors
- Locks:
  - support timesharing, I/O, and MPs
- Configuration Address Space

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Summary of Bus Options

•*Option*	*High performance*	*Low cost*
•Bus width	Separate address & data lines	Multiplex address & data lines
•Data width	Wider is faster (e.g., 32 bits)	Narrower is cheaper (e.g., 8 bits)
•Transfer size	Multiple words has less bus overhead	Single-word transfer is simpler
•Bus masters	Multiple (requires arbitration)	Single master (no arbitration)
•Clocking	Synchronous	Asynchronous
•Protocol	pipelined	Serial

Approaches to Increasing I/O Bandwidth

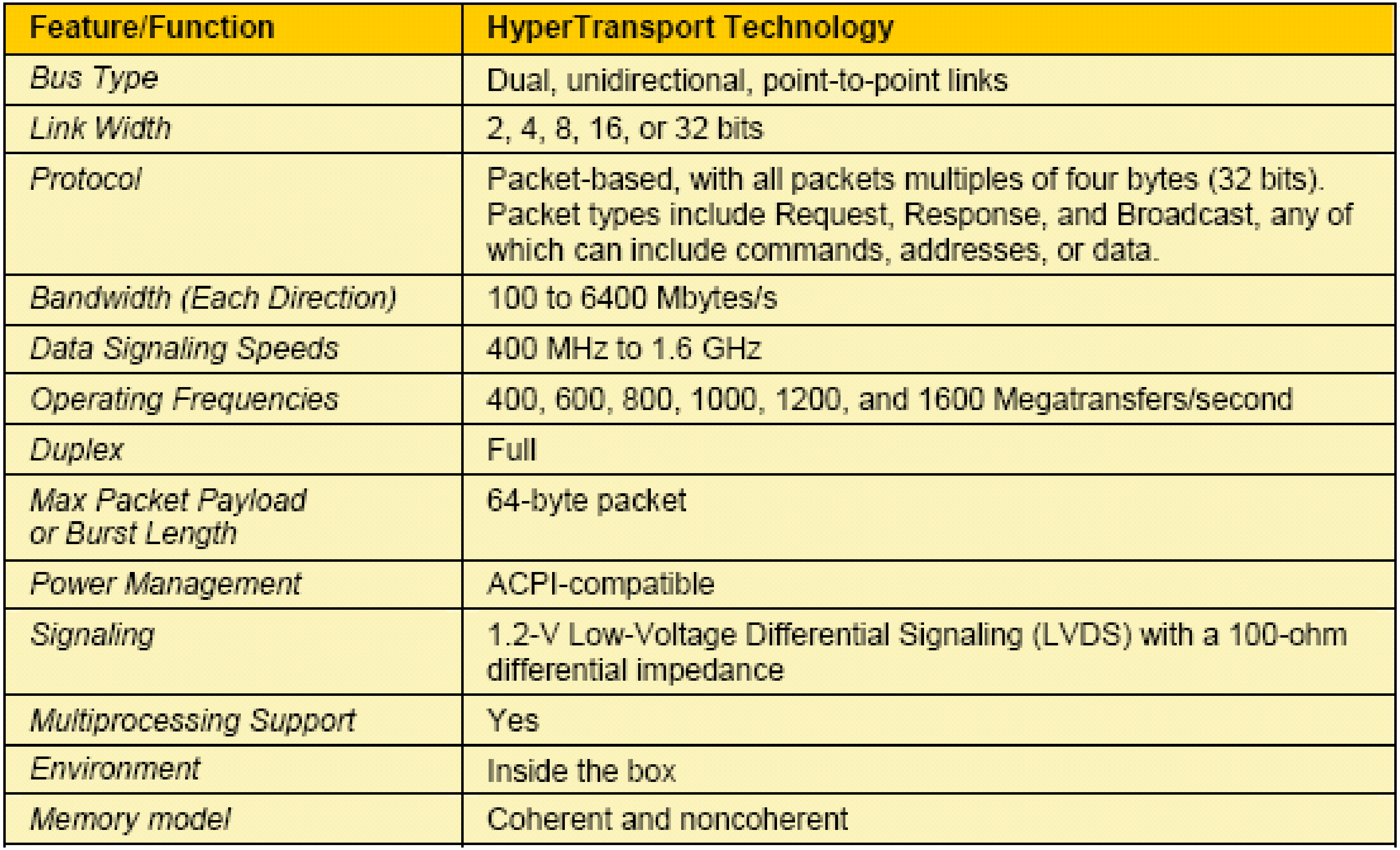
- **Increase bus widths: [DAN] this quickly causes problems with board layout and skew**
- **Increase clock speeds/source synchronous clocking**
- **Multi-rate clocking: SDR -> DDR -> QDR**
- **Circuit techniques**
- **Lower voltage, differential signalling**
- **Enhancements from process improvements**
- **But, eventually run into signal integrity issues**

With thanks to Dave Patterson (http.cs.berkeley.edu/~patterson) and others

Bus performance limitations

- **Phase-match between clock/data**
- **Difficult to match trace lengths at board level**
- **Particularly when width >= 64-bits**
- **V/T budgets demand tight skew control**
- **Practical limit ~ 800 Mbps**
- **Connectors make things worse**
- **Impedance discontinuities**

AMD HyperTransport (Opteron interconnect)



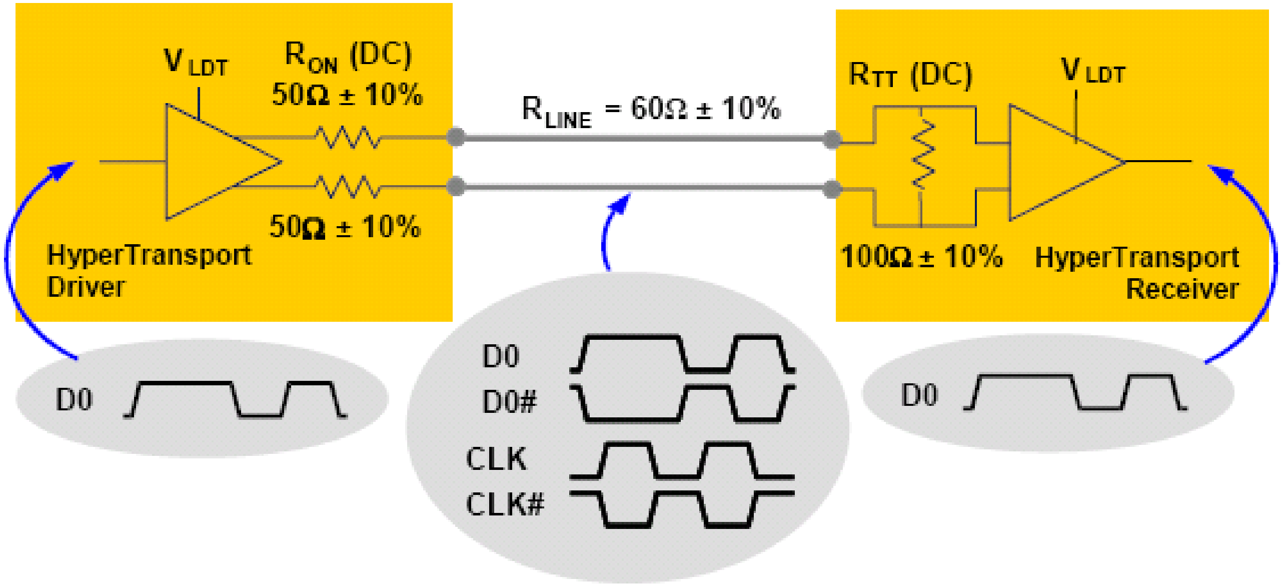
Feature/Function	HyperTransport Technology
Bus Type	Dual, unidirectional, point-to-point links
Link Width	2, 4, 8, 16, or 32 bits
Protocol	Packet-based, with all packets multiples of four bytes (32 bits). Packet types include Request, Response, and Broadcast, any of which can include commands, addresses, or data.
Bandwidth (Each Direction)	100 to 6400 Mbytes/s
Data Signaling Speeds	400 MHz to 1.6 GHz
Operating Frequencies	400, 600, 800, 1000, 1200, and 1600 Megatransfers/second
Duplex	Full
Max Packet Payload or Burst Length	64-byte packet
Power Management	ACPI-compatible
Signaling	1.2-V Low-Voltage Differential Signaling (LVDS) with a 100-ohm differential impedance
Multiprocessing Support	Yes
Environment	Inside the box
Memory model	Coherent and noncoherent

In HyperTransport technology:

- The *physical layer* defines the physical and electrical characteristics of the protocol. This layer interfaces to the physical world and includes data, control, and clock lines.
- The *data link layer* includes the initialization and configuration sequence, periodic cyclic redundancy check (CRC), disconnect/reconnect sequence, information packets for flow control and error management, and doubleword framing for other packets.
- The *protocol layer* includes the commands, the virtual channels in which they run, and the ordering rules that govern their flow.
- The *transaction layer* uses the elements provided by the protocol layer to perform actions, such as reads and writes.
- The *session layer* includes rules for negotiating power management state changes, aswell as interrupt and system management activities.

HyperTransport Physical Layer

HyperTransport technology uses low-voltage differential signalling with differential impedance (ZOD) of 100 ohms for CAD, Clock, and Control signals, as illustrated in Figure 4. Characteristic line impedance is 60 ohms. The driver supply voltage is 1.2 volts, instead of the conventional 2.5 volts for standard LVDS. Differential signalling and the chosen impedance provide a robust signalling system for use on low-cost printed circuit boards. Common four-layer PCB materials with specified di-electric, trace, and space dimensions and tolerances or controlled impedance boards are sufficient to implement a HyperTransport I/O link. The differential signalling permits trace lengths up to 24 inches for 800 Mbit/s operation.



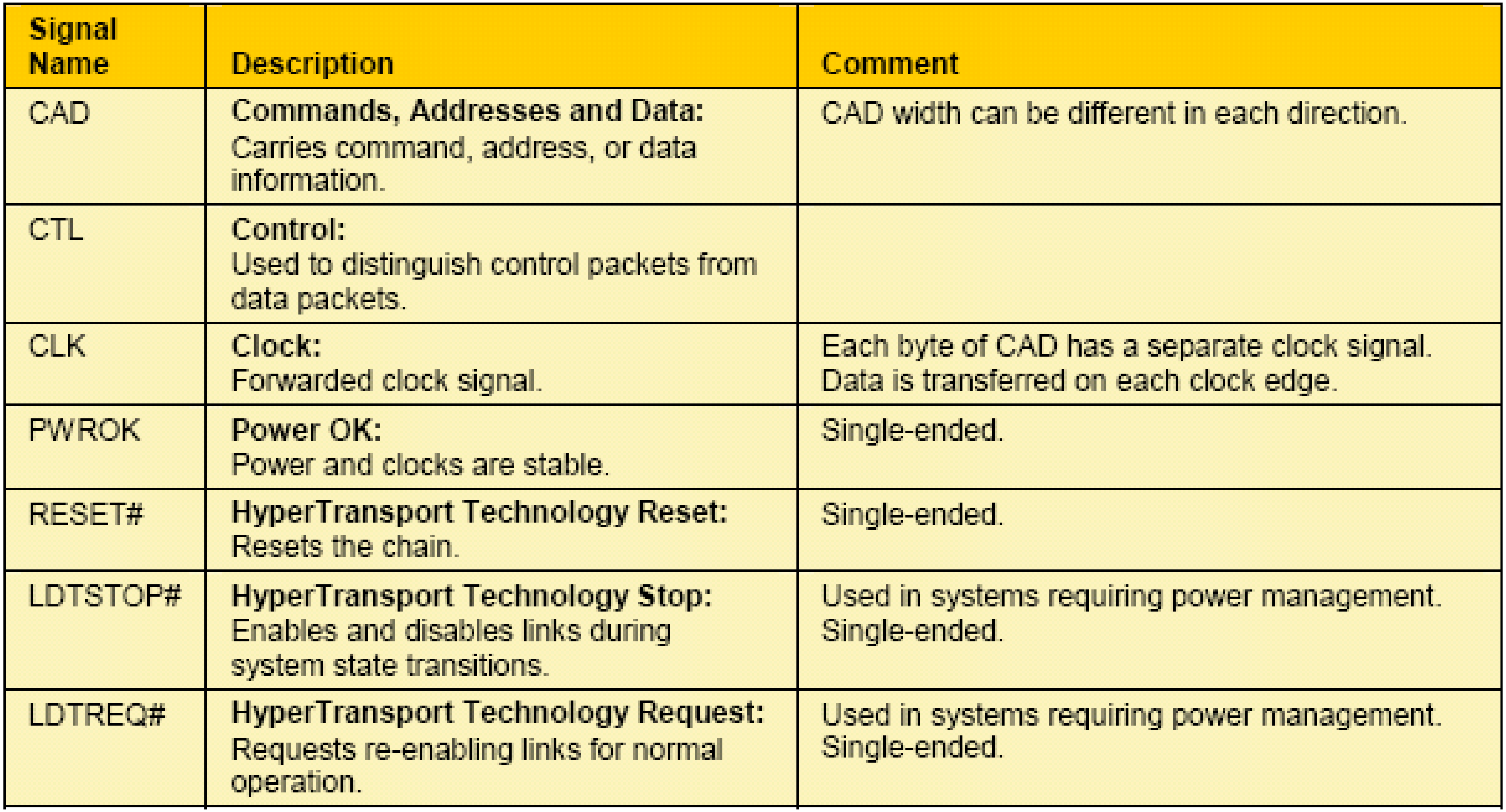
Device Configurations

- HyperTransport technology creates a packet-based link implemented on two independent, unidirectional sets of signals. It provides a broad range of system topologies built with three generic device types:
  - *Cave*—A single-link device at the end of the chain.
  - *Tunnel*—A dual-link device that is not a bridge.
  - *Bridge*—Has a primary link upstream link in the direction of the host and one or more secondary links.
- Example configurations include:
  - A cave device connected directly to a host bridge.
  - A chain of tunnel devices connected to a host bridge.
  - Multiple chains of tunnel devices connected to a bridge, which is then connected to a host bridge.
  - Multiple chains of tunnel devices connected to a switch, which is then connected to a host bridge.
  - Any combination of the above.

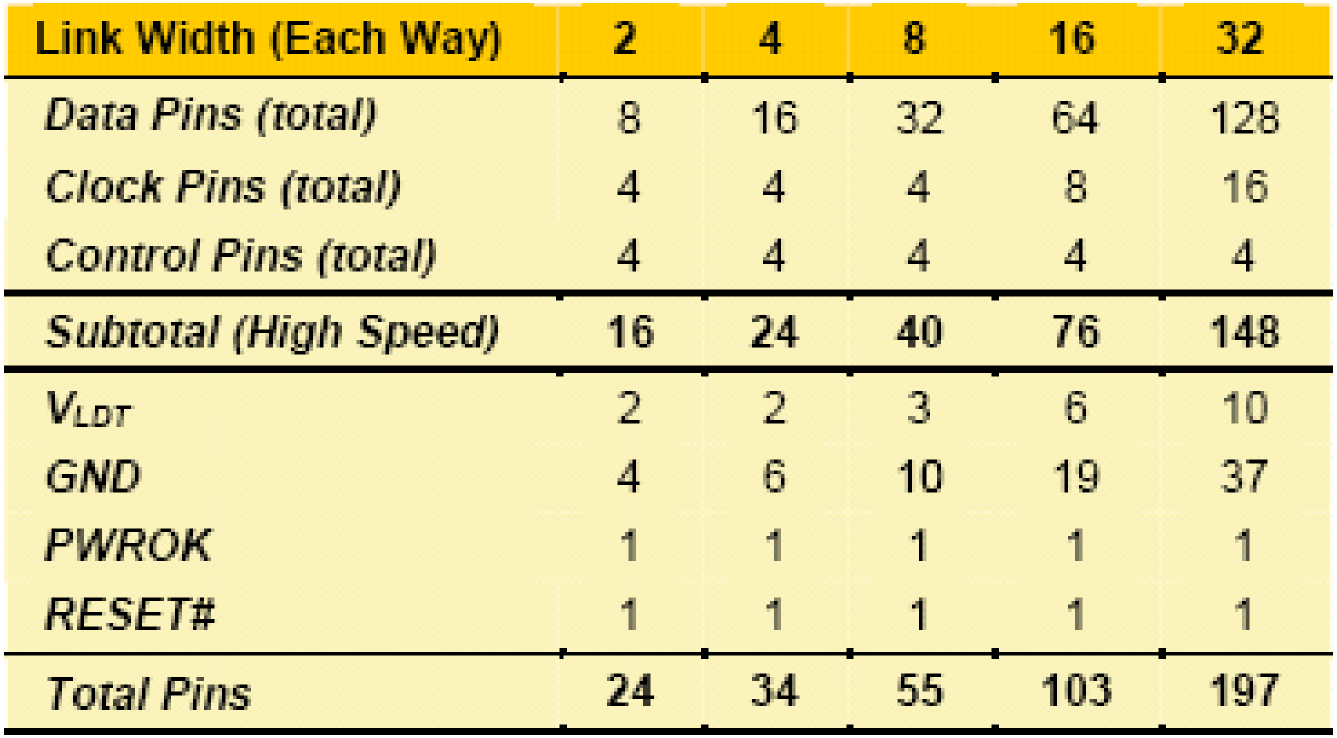
[http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/HyperTransport_IO_Link_Whitepaper_25012A.pdf]

Commands, addresses, and data (CAD) all share the same bits.

- Each data path includes a Control (CTL) signal and one or more Clock (CLK) signals.
- The CTL signal differentiates commands and addresses from data packets.
- For every grouping of eight bits or less within the data path, there is a forwarded CLK signal. Clock forwarding reduces clock skew between the reference clock signal and the signals traveling on the link. Multiple forwarded clocks limit the number of signals that must be routed closely in wider HyperTransport links.
- For most signals, there are two pins per bit.
- In addition to CAD, Clock, Control, VLDT power, and ground pins, each HyperTransport device has Power OK (PWROK) and Reset (RESET#) pins. These pins are single-ended because of their low-frequency use.
- Devices that implement HyperTransport technology for use in lower power applications such as notebook computers should also implement Stop (LDTSTOP#) and Request (LDTREQ#). These power management signals are used to enter and exit low-power states.

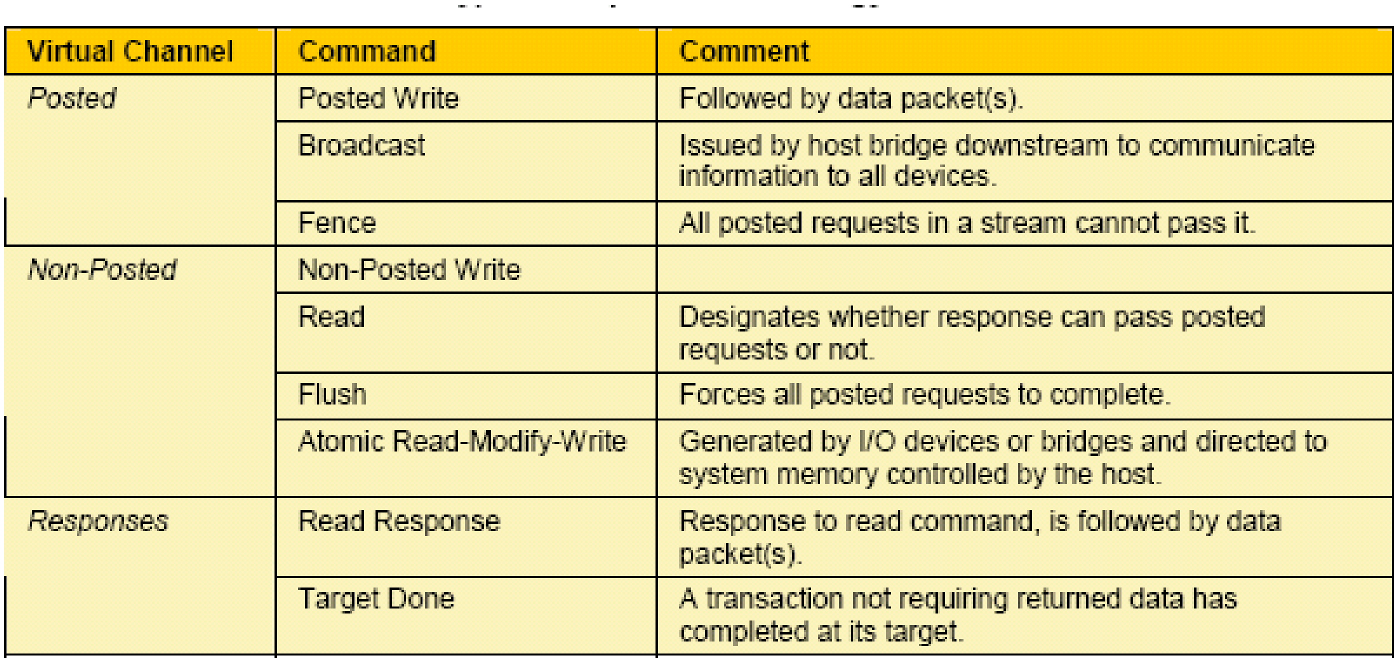


Signal Name	Description	Comment
CAD	**Commands, Addresses and Data:** Carries command, address, or data information.	CAD width can be different in each direction.
CTL	**Control:** Used to distinguish control packets from data packets.	
CLK	**Clock:** Forwarded clock signal.	Each byte of CAD has a separate clock signal. Data is transferred on each clock edge.
PWROK	**Power OK:** Power and clocks are stable.	Single-ended.
RESET#	**HyperTransport Technology Reset:** Resets the chain.	Single-ended.
LDTSTOP#	**HyperTransport Technology Stop:** Enables and disables links during system state transitions.	Used in systems requiring power management. Single-ended.
LDTREQ#	**HyperTransport Technology Request:** Requests re-enabling links for normal operation.	Used in systems requiring power management. Single-ended.



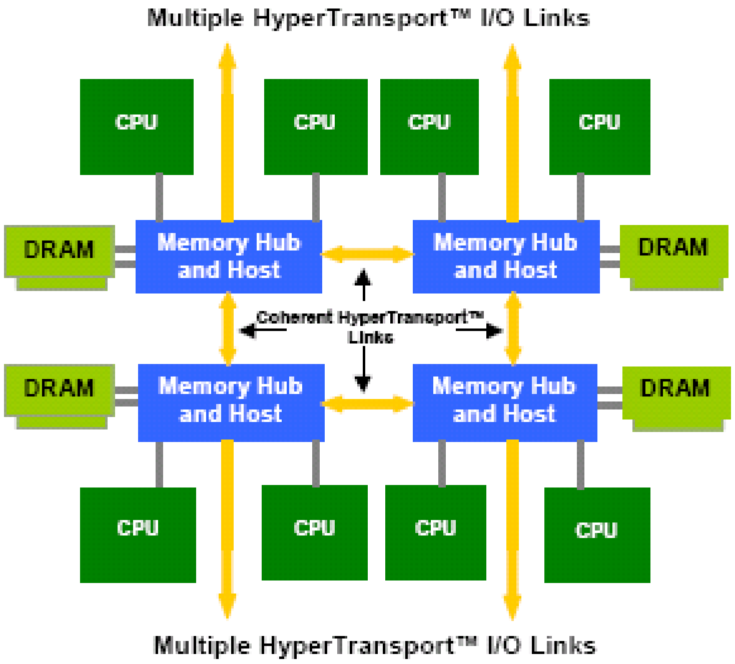
Link Width (Each Way)	2	4	8	16	32
Data Pins (total)	8	16	32	64	128
Clock Pins (total)	4	4	4	8	16
Control Pins (total)	4	4	4	4	4
Subtotal (High Speed)	**16**	**24**	**40**	**76**	**148**
V_{LDT}	2	2	3	6	10
GND	4	6	10	19	37
PWROK	1	1	1	1	1
RESET#	1	1	1	1	1
Total Pins	**24**	**34**	**55**	**103**	**197**

- All HyperTransport technology commands are either four or eight bytes long and begin with a 6-bit command type field. The most commonly used commands are Read Request, Read Response, and Write. A virtual channel contains requests or responses with the same ordering priority.



Virtual Channel	Command	Comment
Posted	Posted Write	Followed by data packet(s).
	Broadcast	Issued by host bridge downstream to communicate information to all devices.
	Fence	All posted requests in a stream cannot pass it.
Non-Posted	Non-Posted Write	
	Read	Designates whether response can pass posted requests or not.
	Flush	Forces all posted requests to complete.
	Atomic Read-Modify-Write	Generated by I/O devices or bridges and directed to system memory controlled by the host.
Responses	Read Response	Response to read command, is followed by data packet(s).
	Target Done	A transaction not requiring returned data has completed at its target.

AMD is using a superset of HyperTransport technology to handle cache coherency in its multiprocessor server systems. In the model of a four-processor server each of the Northbridges has two coherent HyperTransport links to create an array of four CPUs. Each core logic chipset has its own memory and a HyperTransport link available for I/O expansion.



AMD CTO: Four chips are better than two

By Michael Singer, CNET News.com
Published on ZDNet News: January 4, 2006, 12:00 PM PT

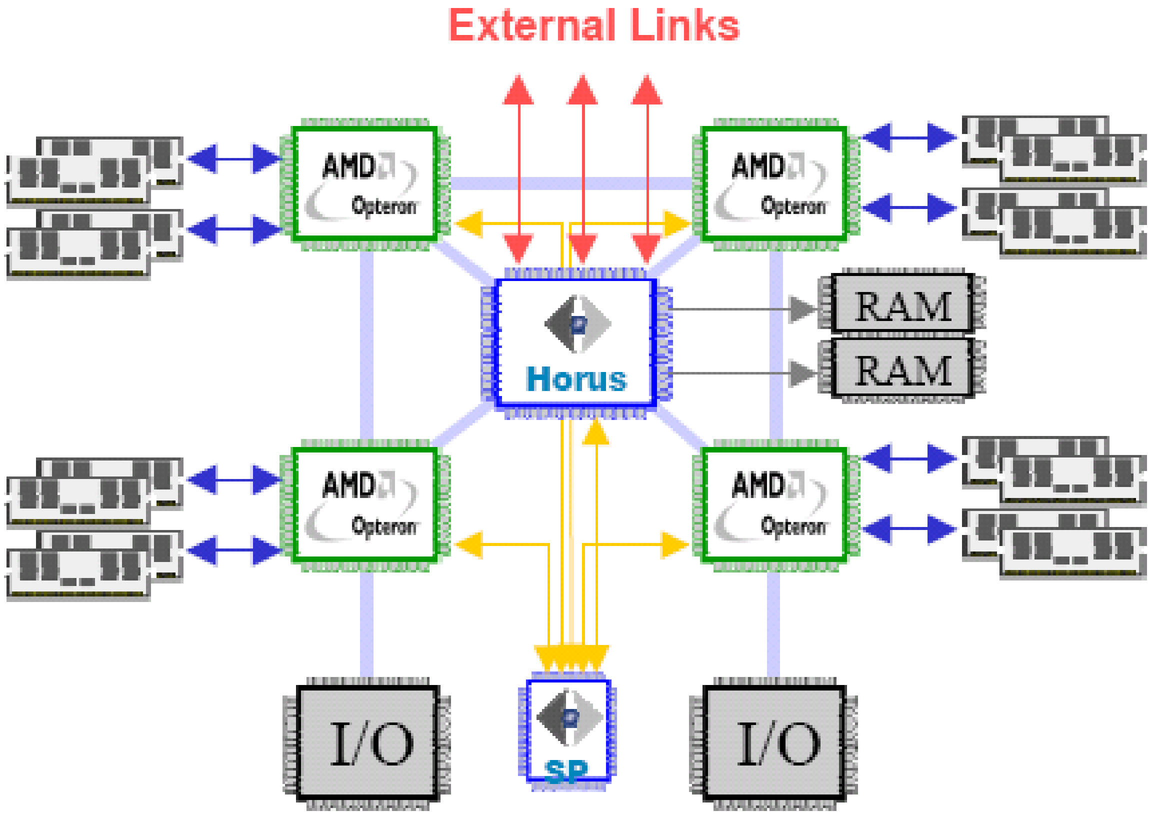
Q: Is it true that AMD is looking at licensing parts of its architecture?
Hester: The idea has been around for a while but it relates to understanding what our direct customers and the end customers want to do. The idea is to selectively license the coherent HyperTransport technology (a chip-to-chip interconnect supported by AMD and primarily used on a computer system board in distances up to 24 inches).

Isn't this the technology that you have been promoting for years as a better alternative to Intel's front-side bus systems or its upcoming Common System Interconnect?
Hester: Exactly.

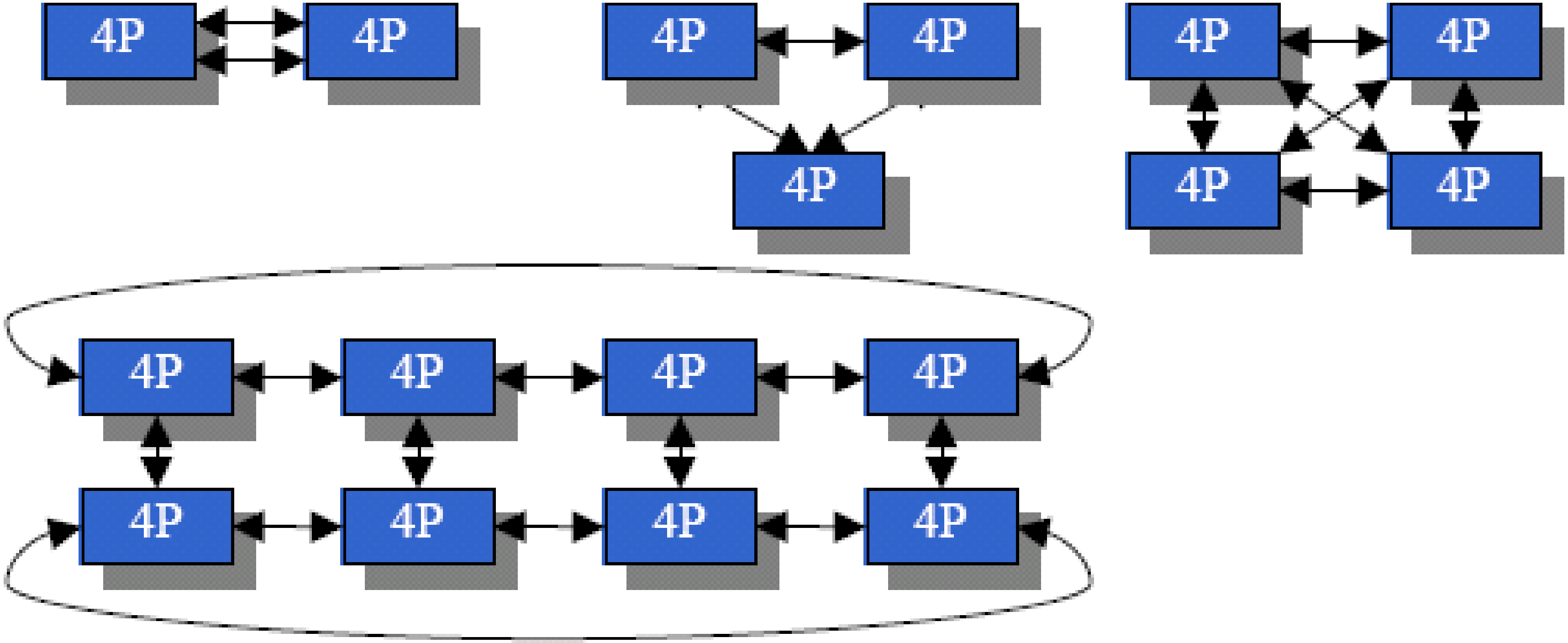
And you're going to license that to other people?
Hester: Right. So the example I'd give you is in the high-performance tactical computing area, where people like Cray and others would like to do vector floating-point units.

Being able to do that requires a co-processor, or attached processor elements that would attach into a standard system. We don't have any finalized plans yet, but if you look at the workloads in the data center, you're starting to see applications where, if you could accelerate XML and Java, a number of the vertical applications would perform significantly better. So instead of trying to build a machine that's just aimed at workloads, you can think about the attached processor or co-processor that works in conjunction with our AMD64 architecture to accelerate those workloads.

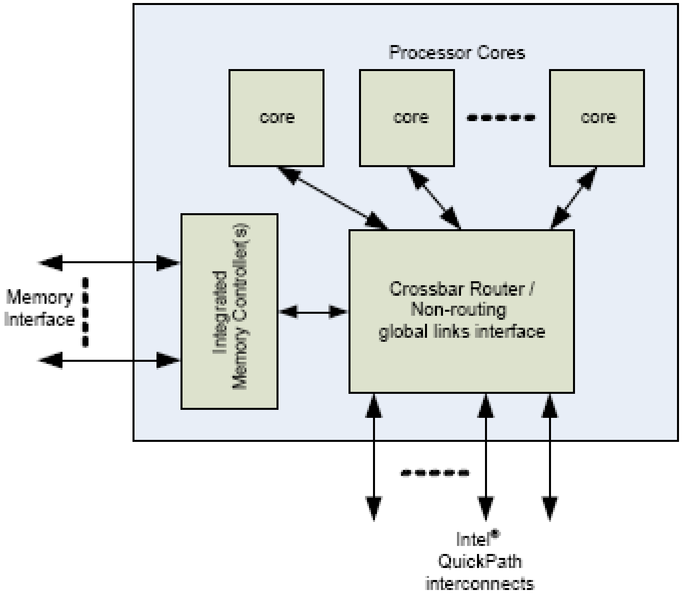
Horus


External Links
AMD Opteron
AMD Opteron
Horus
RAM
RAM
AMD Opteron
AMD Opteron
I/O
SP
I/O

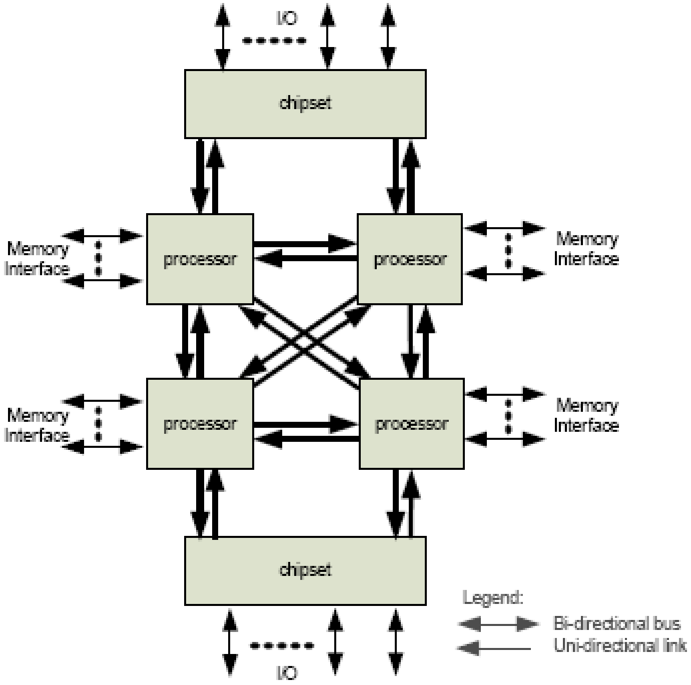
Horus



Intel QuickPath Interconnect (Core i7)


Processor Cores
core
core
core
Integrated Memory Controller(s)
Crossbar Router / Non-routing global links interface
Memory Interface
Intel® QuickPath interconnects

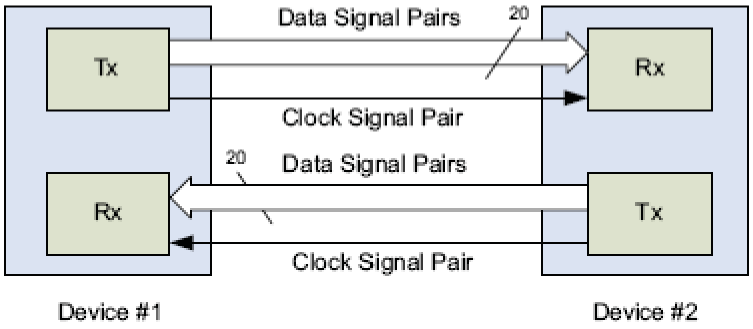
Processor

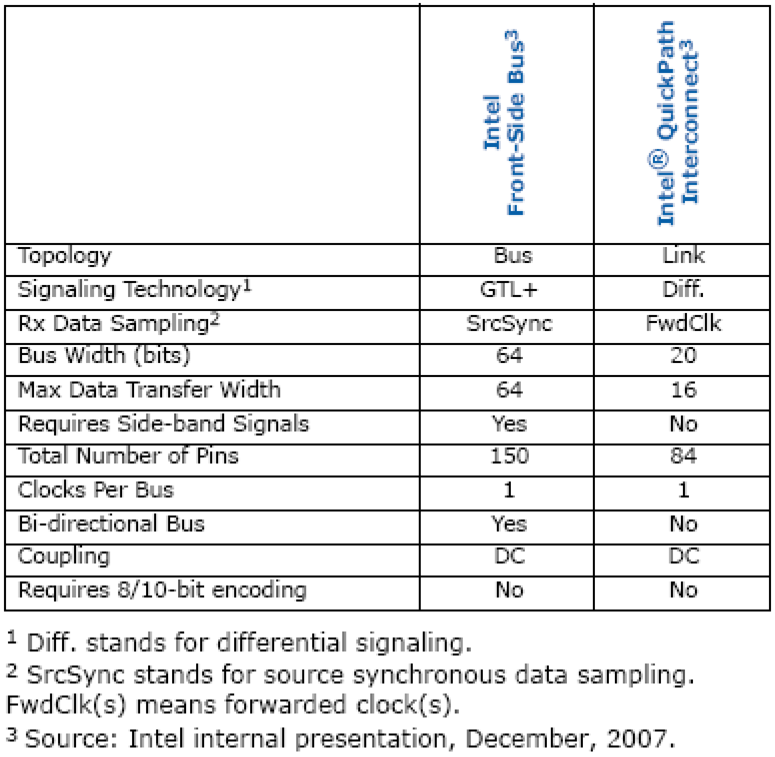


System

From: Intel’s An Introduction to the Intel® QuickPath Interconnect

Intel QuickPath Interconnect



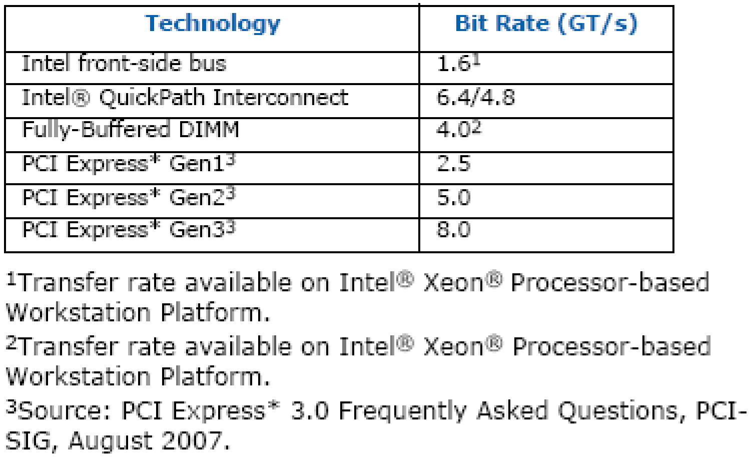


	Intel Front-Side Bus[3]	Intel® QuickPath Interconnect[3]
Topology	Bus	Link
Signaling Technology[1]	GTL+	Diff.
Rx Data Sampling[2]	SrcSync	FwdClk
Bus Width (bits)	64	20
Max Data Transfer Width	64	16
Requires Side-band Signals	Yes	No
Total Number of Pins	150	84
Clocks Per Bus	1	1
Bi-directional Bus	Yes	No
Coupling	DC	DC
Requires 8/10-bit encoding	No	No

[1] Diff. stands for differential signaling.

[2] SrcSync stands for source synchronous data sampling. FwdClk(s) means forwarded clock(s).

[3] Source: Intel internal presentation, December, 2007.



Technology	Bit Rate (GT/s)
Intel front-side bus	1.6[1]
Intel® QuickPath Interconnect	6.4/4.8
Fully-Buffered DIMM	4.0[2]
PCI Express* Gen1[3]	2.5
PCI Express* Gen2[3]	5.0
PCI Express* Gen3[3]	8.0

[1] Transfer rate available on Intel® Xeon® Processor-based Workstation Platform.

[2] Transfer rate available on Intel® Xeon® Processor-based Workstation Platform.

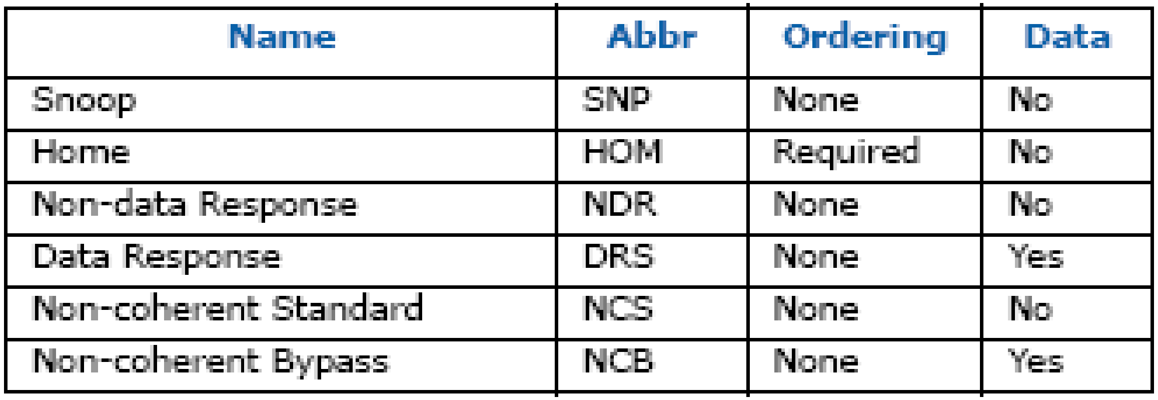
[3] Source: PCI Express* 3.0 Frequently Asked Questions, PCI-SIG, August 2007.

From: Intel's An Introduction to the Intel® QuickPath Interconnect

Intel QuickPath Interconnect

- A Phit contains all the information transferred by the Physical layer on a single clock edge. At full-width that would be 20 bits, at halfwidth 10 bits and at quarter-width 5 bits.
- A Flit is always 80 bits regardless of the link width, so the number of Phits needed to transmit a Flit will increase by a factor of two or four for half and quarter-width links, respectively.
- The Link layer uses a credit/debit scheme for flow control. During initialization, a sender is given a set number of credits to send packets, or Flits, to a receiver. Whenever a packet or Flit is sent to the receiver, the sender decrements its credit counters by one credit, which can represent either a packet or a Flit depending on the type of virtual network being used.

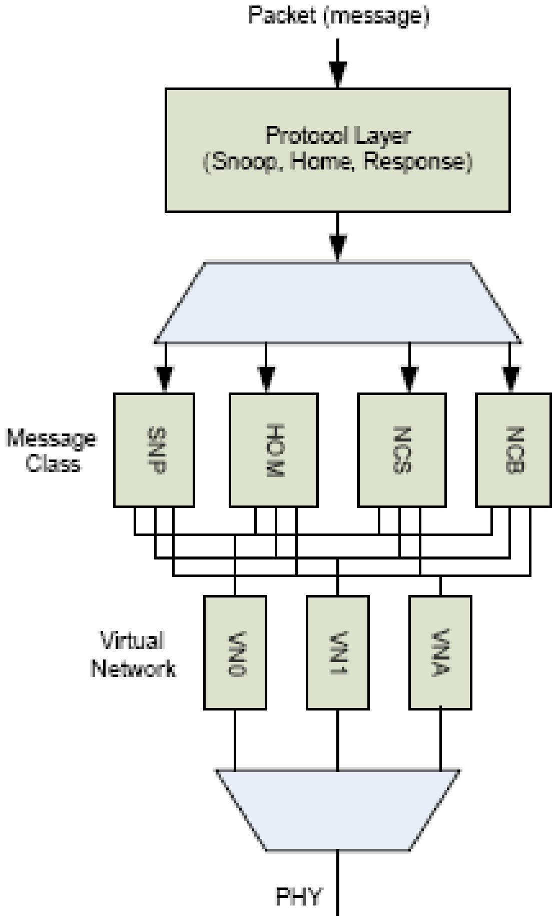
Messages:



Name	Abbr	Ordering	Data
Snoop	SNP	None	No
Home	HOM	Required	No
Non-data Response	NDR	None	No
Data Response	DRS	None	Yes
Non-coherent Standard	NCS	None	No
Non-coherent Bypass	NCB	None	Yes

QuickPath virtual network

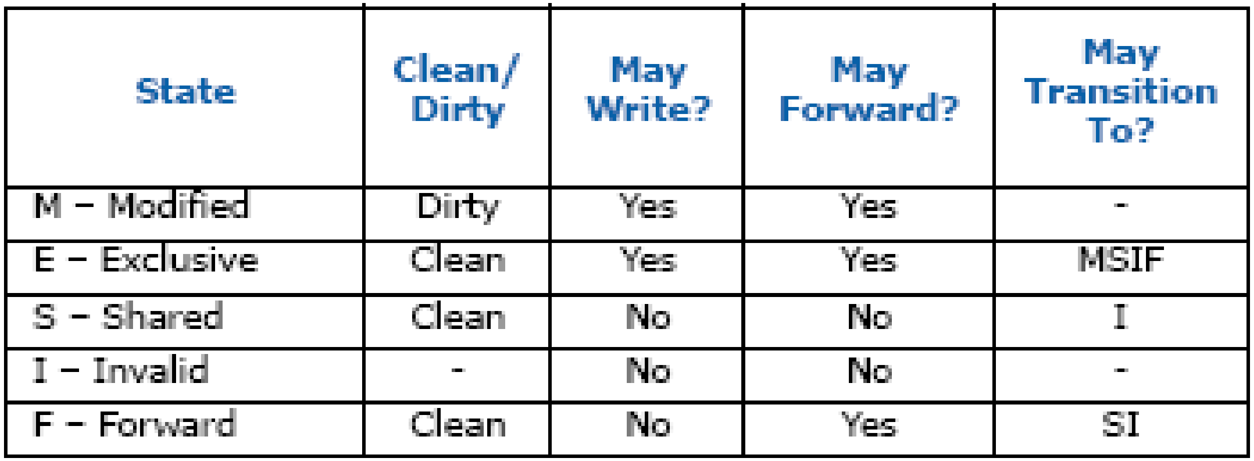
(up to 3 per link)



Note: Only 4 of 6 message classes shown.

QuickPath cache protocol: MESIF

- The Intel® QuickPath Interconnect implements a modified format of the MESI coherence protocol. The standard MESI protocol maintains every cache line in one of four states: modified, exclusive, shared, or invalid. A new read-only forward state has also been introduced to enable cache-to-cache clean line forwarding.



State	Clean/ Dirty	May Write?	May Forward?	May Transition To?
M – Modified	Dirty	Yes	Yes	-
E – Exclusive	Clean	Yes	Yes	MSIF
S – Shared	Clean	No	No	I
I – Invalid	-	No	No	-
F – Forward	Clean	No	Yes	SI