Buses and Interconnection Networks

ECE 485/585

Mark G. Faust

Buses and Interconnection Networks

- Topics
  - Introduction
    - Why study buses?
    - What are buses?
    - Overview and definitions
  - Buses
    - Bandwidth
    - Arbitration
    - Enumeration and Configuration
    - Compatibility (Backward and Forward)
    - PC Bus Evolution
      - PC (8-bit ISA)
      - ISA, EISA
      - PCI
      - AGP
      - PCI Express
    - Parallel buses
    - Serial buses

Why study buses?

- Bus design plays a huge role in computer architecture
  - Performance, Cost
  - Flexibility, Reliability
  - Quality of Service

Speed
Data width
Complexity

Processor Family	Data Bus Width
8088	8-bits
8086, 286, 386SX	16-bits
386DX, 486DX	32-bits
Pentium	64-bits

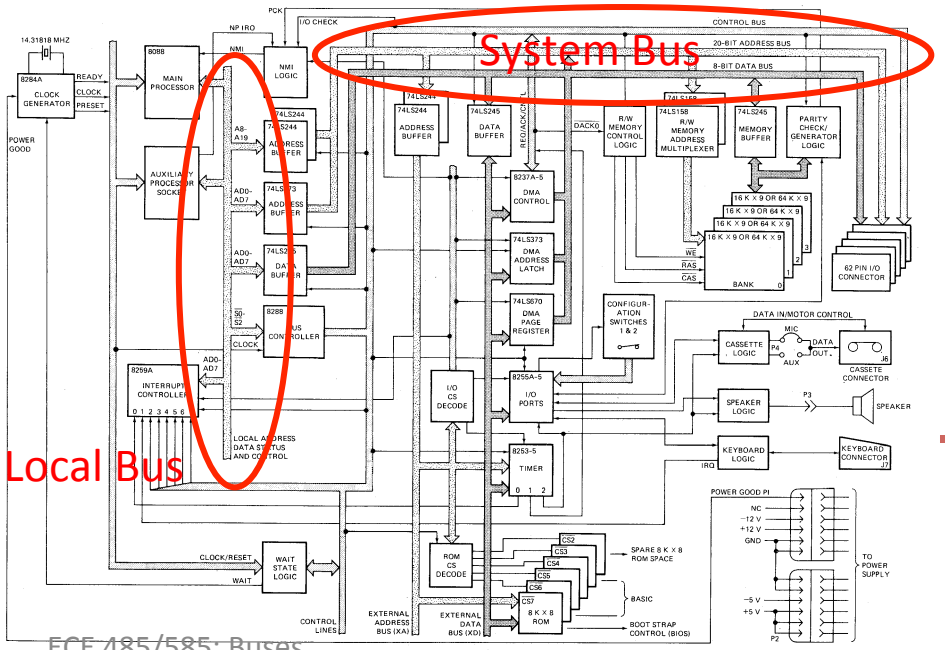
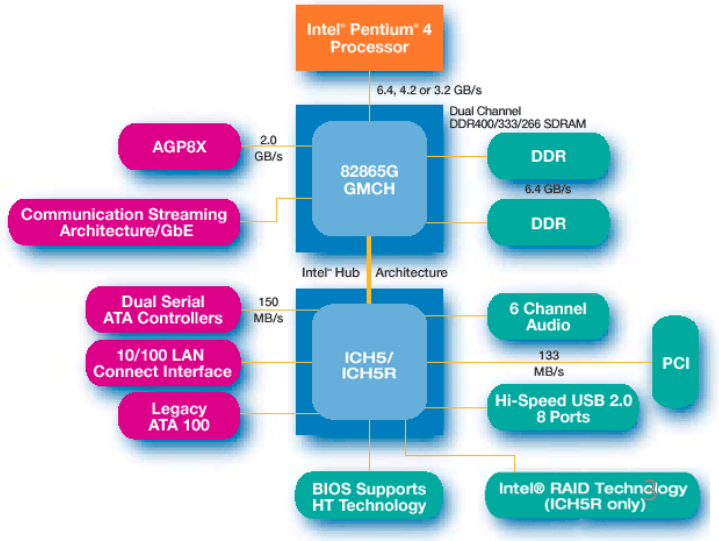


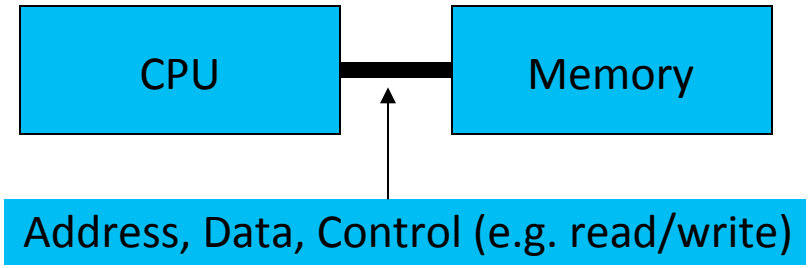
FIGURE 11-2 Block diagram of circuitry on IBM PC motherboard. (*IBM Corporation*)



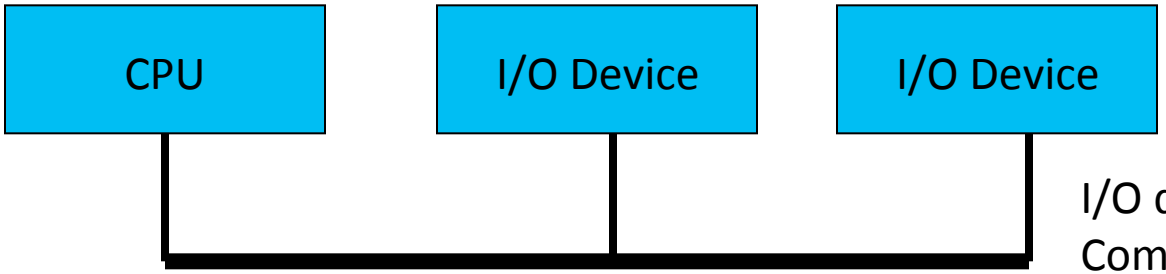
Why study buses?

- Throughout microprocessor-based systems
  - On-chip buses (e.g. processor/cache, intra-CPU, multi-core)
  - Chip-to-Chip
    - Memory bus (connecting one or more processors to memory subsystem)
    - Inter-processor communication
  - Board-to-Board
  - Chassis-to-Chassis
    - I/O buses to connect peripherals (I/O or peripheral buses)
  - System-to-System communication (networking)
  - Interconnecting multiple networks (internetworking)
- Our Objectives
  - Learn basic terminology and concepts related to buses
  - Examine evolution of buses in microprocessor-based systems
  - Study representative buses in current widespread use
  - Focus on common issues (e.g. arbitration, error handling, enumeration)

Bus Basics



CPU initiates all transactions (bus master)
Control and addresses from CPU
Data bidirectional



I/O device may initiate transaction
Competition for bandwidth, initiation
Process to discover all devices on bus

- The bus provides
  - Physical interconnection structure that's shared among agents
  - Rules for how different agents can connect to the bus
  - Rules for communicating across the bus (protocol)
- Bus arbitration
  - Determines which of two or more agents becomes bus master

Bus Definition Requires:

- Mechanical specification
  - Connectors, Plugs, Receptacles
  - Cable length
- Electrical specification
  - Voltage, termination
- Timing and signaling specification
  - Clocking, Timing
  - Handshaking
- Protocol
  - Packet numbering/sequencing
  - Arbitration
  - Error handling
  - Addressing
  - Initialization/Configuration

LLC (logical link)
MAC (media access control)

Layer	Examples
Application	FTP, Telnet, HTTP
Presentation	MIME
Session	Named Pipes
Transport	TCP
Network	IP
Data Link	
Physical Mechanical	RS-232, USB, 802.11, Bluetooth, Ethernet

OSI layer model

Issues in Bus Design

- Topology
  - Multi-drop (broadcast) vs. Point-to-Point
- Data Width
  - Serial vs. Parallel (number of bits)
- Speed
- Multiplexing
  - Will signals have more than one use (e.g. address/data lines)?
- Clocking
  - Asynchronous
  - Synchronous
  - Source Synchronous (possibly embedded in data)
- Arbitration
  - How does a device (agent) become the bus master?
- Addressing
  - How are agents uniquely identified?
  - How many agents/endpoints can be defined?
- Error detection and handling
- Configuration
  - Static: one CPU ←→ one memory
  - Dynamic at initialization: one CPU/multiple I/O devices fixed at boot time
  - Hot-pluggable: plug and play

Bus Design: Concepts and Terms

- Bus cycle time
  - Number of transfers/second
- Bandwidth
  - Amount of data/transfer
- Total Bus Bandwidth
  - Maximum number of bytes/second
- Theoretical bandwidth
  - Achievable without considering bus conflict, errors, protocol overhead
- Actual bandwidth
  - Realizable bandwidth when these factors taken into consideration
- Latency
  - Delay from initiation to start of response
- Concurrency
  - Ability to do more than one thing at a time (just as in CPU)
  - Achieved via replication or pipelining
    - Replication: e.g. wider bus → more bytes transferred at a time
    - Pipelining: multiple transactions simultaneously by using different parts of the bus

Units

- Bits vs. Bytes
  - B (upper case) is used for bytes
    - 60MB/s is 60 megabytes per second
  - b (lower case) is used for bits
    - 480Mbps is 480 megabits per second
- What is 1K?
  - Communications/data transfer (decimal)
    - $1K = 10^3$ $1M = 10^6$
    - $1MB/s = 10^6$ bytes/second
  - Memory (binary)
    - $1K = 2^{10} = 1024$ $1M = 2^{10} \times 2^{10} = 2^{20} = 1{,}048{,}576$
  - Disk
    - If you're an engineer – binary
    - If you're a marketing/sales person – decimal
    - 128 GB drive (137×10^9 B) [2^{28} sectors x 512 bytes/sector]

Bandwidth

- Theoretical bandwidth
  - Speed x Data Width
    - 32 bits x 66 MHz = 4 x 66 MB/second = 264 MB/s
- Actual bandwidth
  - Protocol – how many cycles for a transfer?
    - Cycle to send address
    - Cycle to fetch data (device latency)
    - Cycle to transmit data
  - Arbitration, Error phases?
  - Delayed Response, Retries
  - "Payload" Efficiency

Back of Envelope Bus Bandwidth Calculations

- ISA Bus
  - 8 MHz, 2 bytes wide, 4 cycles/transfer
  - Assuming no wait states

$$\frac{8\ Mcycles/\sec\ \times 2\ \text{bytes/transfer}}{4\ cycles/transfer} = 4MB/sec$$

- 100 Base T Ethernet
  - 100 Mbps

$$\frac{100\ Mbps}{8\ bits/byte} = 12MB/sec$$

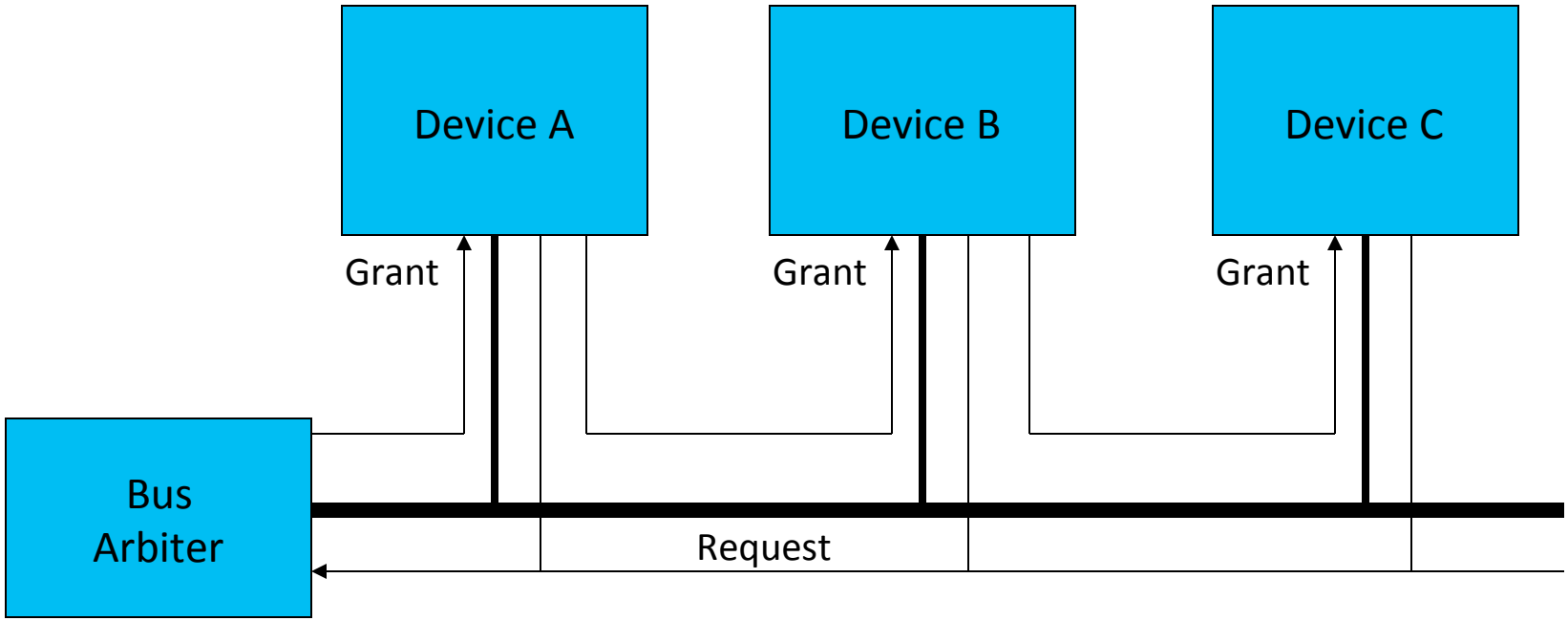
- Video Card
  - 800 x 600 pixels, R/G/B = 8 bits each, 30 frames/sec

$$800 \times 600 \times 3\frac{bytes}{frame} \times 30\ frames/sec = 43.2MB/sec$$

Bus Arbitration

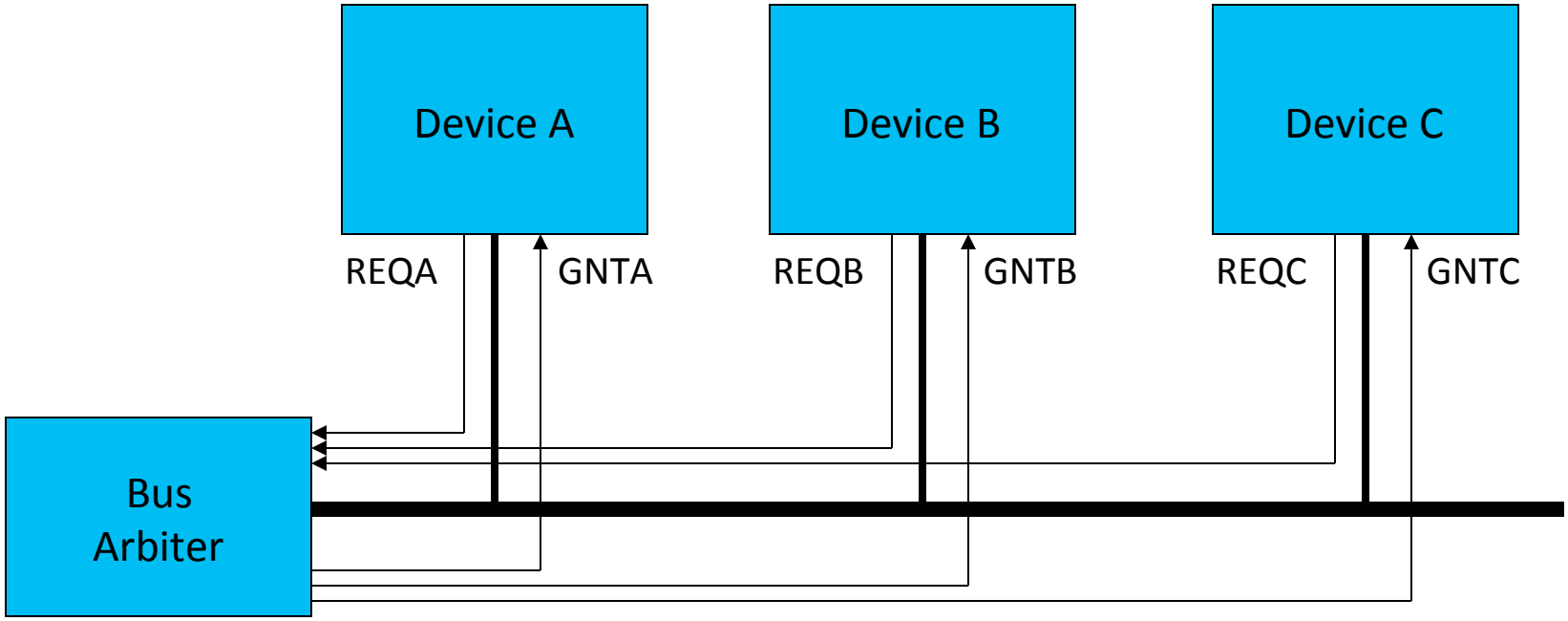
- What happens when more than one agent wants control of the bus (i.e. to become bus master)?
- Bus arbitration decides which agent gets to control the bus
- Arbitration can be
  - Centralized
    - Daisy chained (VME)
    - Dedicated lines (PCI)
  - Decentralized
    - Self-selection (Apple Mac NuBus, FSB, SCSI)
    - Collision detection (Ethernet)

Centralized Daisy Chain Arbitration



- Advantage
  - Simplicity
- Disadvantages
  - Low priority devices can be locked out indefinitely
  - Daisy chain grant signal limits bus speed

Centralized Parallel Bus Arbitration



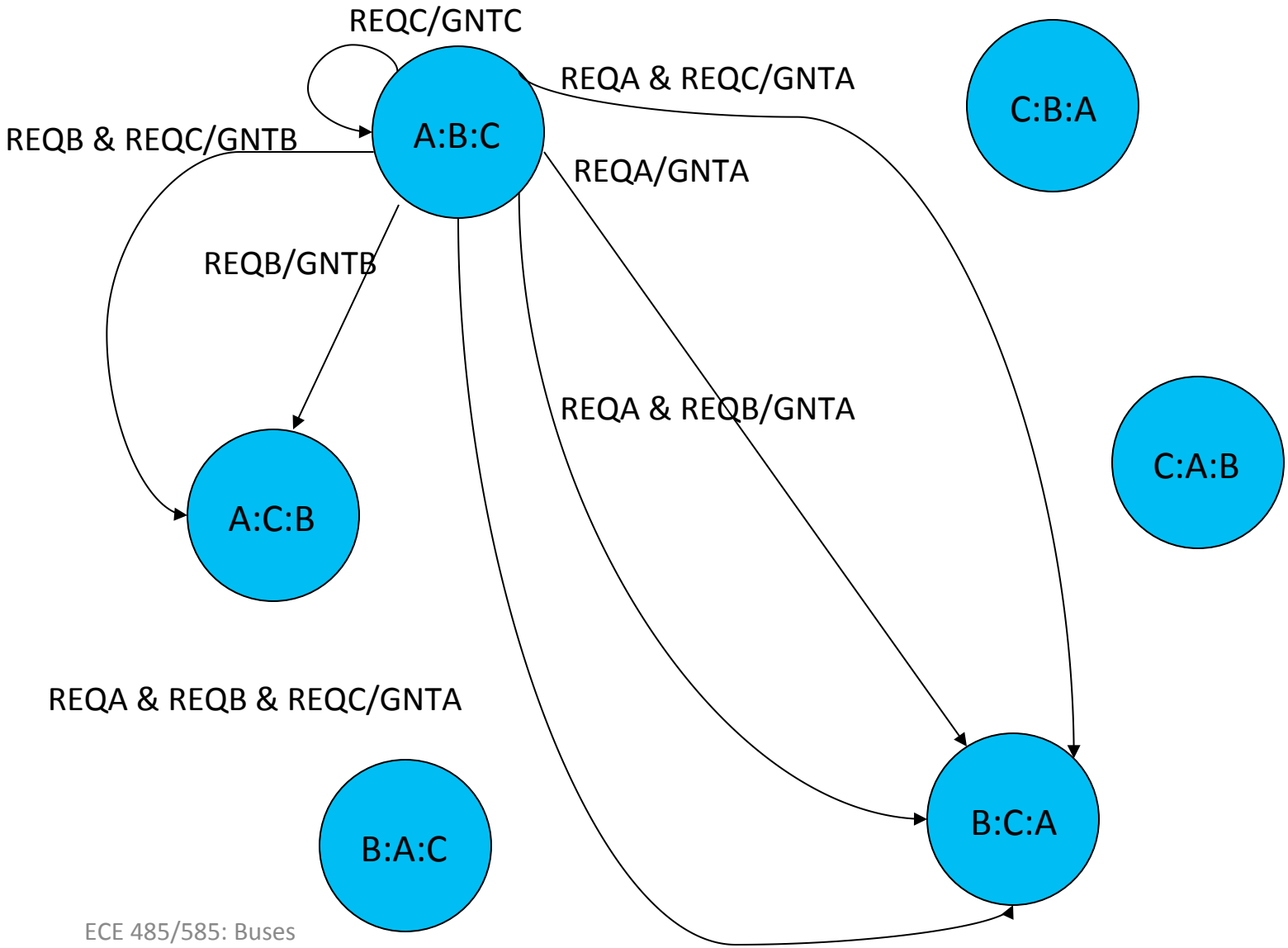
Arbitration Policy

- Goals
  - Priority
    - Highest priority agent should be serviced first
    - Priority can be static or dynamic
  - Fairness
    - Even lowest priority agent should not be locked out
    - Guaranteed worst case latency

An Unbiased Arbiter

- Consider three bus agents (A,B,C)
- Assign an initial priority (e.g. A:B:C)
- After a request is granted the agent's priority is set to the lowest priority
- If a single request is received it is granted
- If two or more requests are received the agent which is highest priority is granted access

Implementation of Unbiased Arbiter



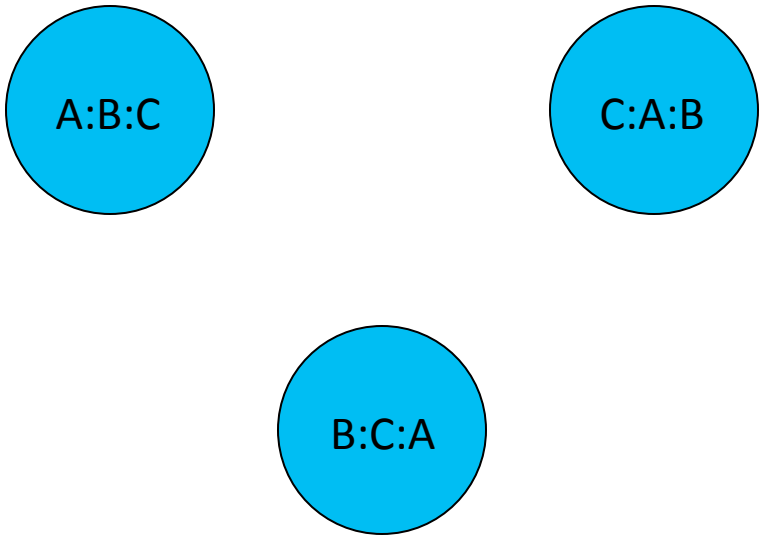
Next State/Output Table

State	Requests						
	A	B	C	A,B	A,C	B,C	A,B,C
A:B:C	A B:C:A	B A:C:B	C A:B:C	A B:C:A	A B:C:A	B A:C:B	A B:C:A
A:C:B	A C:B:A	B A:C:B	C A:B:C	A C:B:A	A C:B:A	C A:B:C	A C:B:A
B:A:C	A B:C:A	B A:C:B	C B:A:C	B A:C:B	A B:C:A	B A:C:B	B A:C:B
B:C:A	A B:C:A	B C:A:B	C B:A:C	B C:A:B	C B:A:C	B C:A:B	B C:A:B
C:A:B	A C:B:A	B C:A:B	C A:B:C	A C:B:A	C A:B:C	C A:B:C	C A:B:C
C:B:A	A C:B:A	B C:A:B	C B:A:C	B C:A:B	C B:A:C	C B:A:C	C B:A:C

Rotating Priority

Number of states grows as n! where n is number of bus agents. A bus with 8 agents requires 8! = 40,320 states. A simpler solution…

Simplify priority: Just three states. Priority when have conflicting requests is same as before. Transition to state with winning agent as last priority.

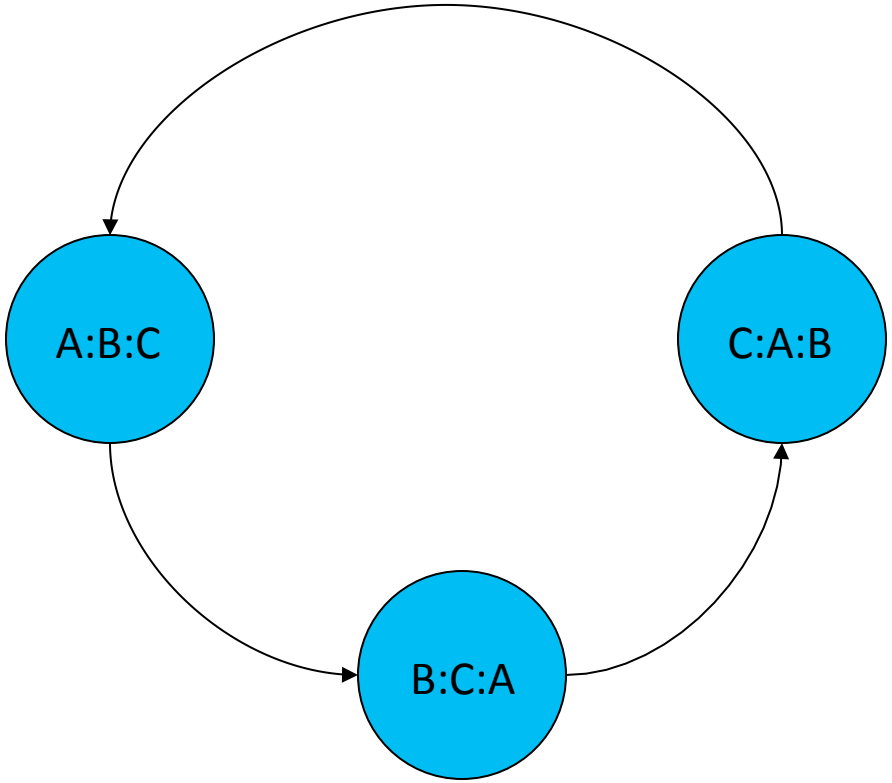


Next State/Output Table

State/Request	A	B	C	A,B	A,C	B,C	A,B,C
A:B:C	A B:C:A	B C:A:B	C A:B:C	A B:C:A	A B:C:A	B C:A:B	A B:C:A
B:C:A	A B:C:A	B C:A:B	C A:B:C	B C:A:B	C A:B:C	B C:A:B	B C:A:B
C:A:B	A B:C:A	B C:A:B	C A:B:C	A B:C:A	C A:B:C	C A:B:C	C A:B:C

No request can be outstanding for more than n cycles where n is the number of agents (states). Therefore, worst case latency is no worse than previous scheme.

Even Simpler: Rotating Priority (Round Robin)



Observe the logic in computing the next state – it's just a counter (or circular shift register).

Device Connection

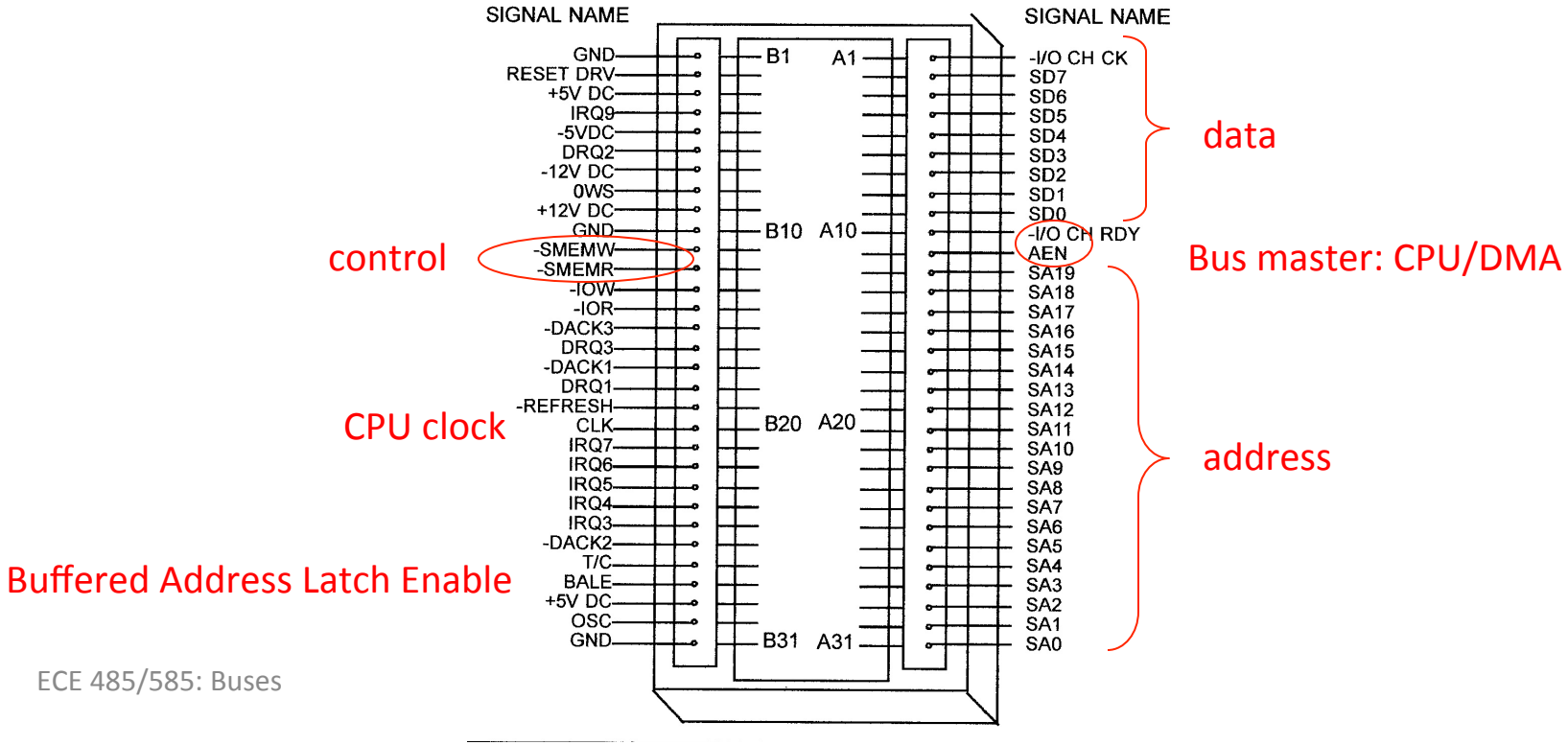
- Need to determine what devices are present
  - Identify requirements and capabilities
    - latency, bandwidth needs, address space
  - Determine or assign (base) address
  - Allocate resources (e.g. buffers)
- Completely static
  - One processor/one memory
  - Processor and fixed address I/O device (hardwired decoding)
- Fixed at boot time (e.g. PCI)
  - Device discovery
  - BIOS enumeration
  - Alternative (initial) means of "addressing" devices
- Plug and play (e.g. USB)
  - Dynamic discovery

Compatibility

- High cost of change
- Large installed base/investment
  - Hardware, software, expertise
- Backward Compatibility
  - New system/device will operate in context of old environment
    - High Speed USB 2.0 device must operate on USB 1.0 connectors, hubs
- Forward Compatibility
  - New system designed so that old devices/systems will operate in the new environment
    - PCI-X bus allows PCI devices to connect

Early PC Bus

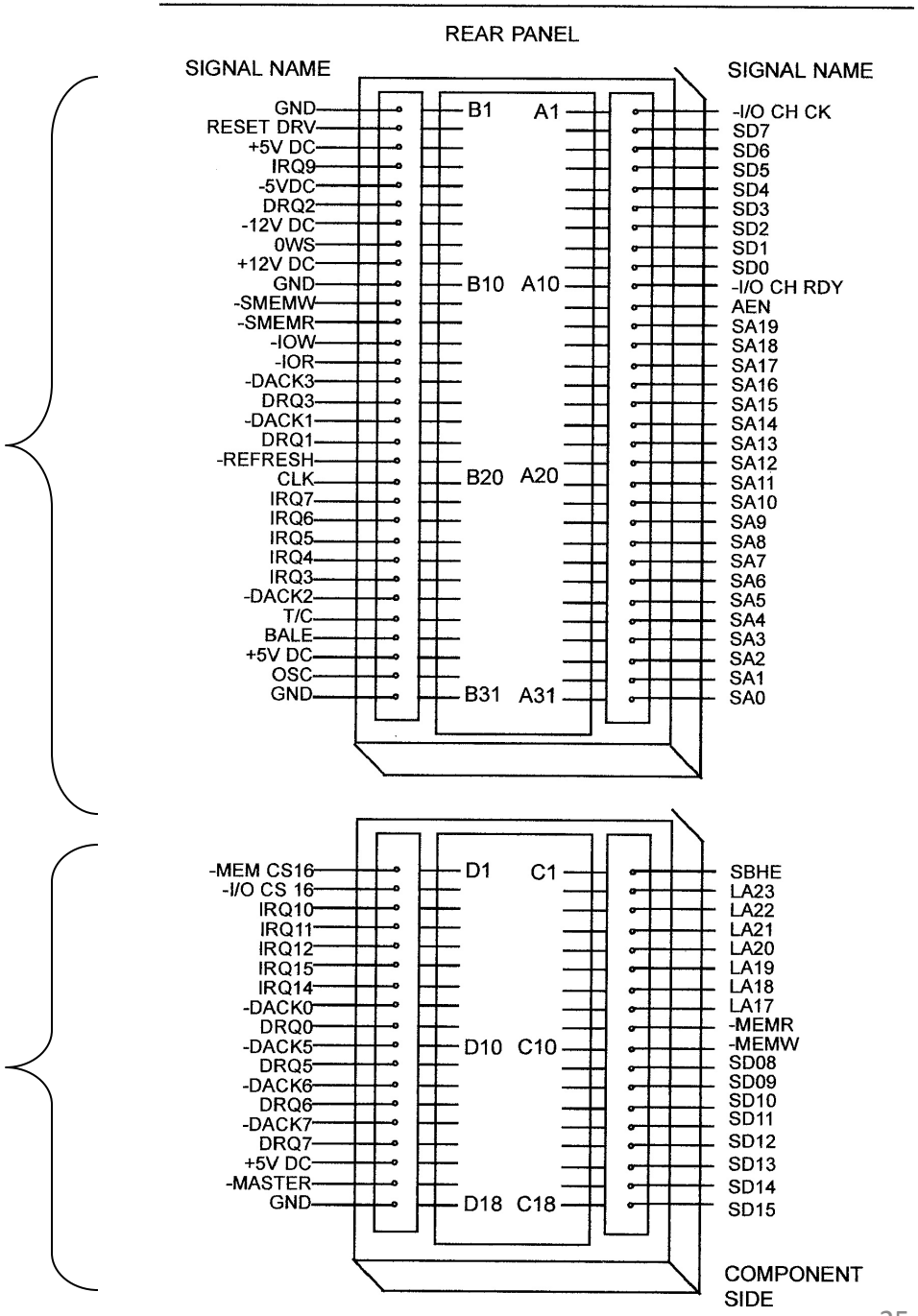
- IBM PC (1981) Intel 8088-based
  - 8-bit data bus
- IBM PC/AT (1984) Intel 80286-based
  - 16-bit data bus
  - Known as ISA (Industry Standard Architecture) Bus because of IBM copyright on PC/AT
  - IBM PC (PC/XT) Bus then referred to as 8-bit portion of ISA bus



ISA Bus

ISA 8-bit portion

ISA 16-bit addition
- additional address bits
- additional 8-bits data



ISA Shortcomings

- Data path limited to 16-bits
  - Intel 386/486 provided 32-bits
- Address bus limited to 24-bits
  - Intel 386/486 provided 32-bits
- DMA limitations
- Electrical considerations
  - Crosstalk, Capacitance, Susceptibility to EMI

ISA Successor

- IBM: Micro Channel Architecture (MCA)
  - Proprietary
- Consortium Introduced Extended ISA (EISA)
  - 32-bit Data Transfers
  - 32-bit Addressing
  - Same speed (8 MHz)
  - Backward compatible
  - Effectively doubled the bandwidth

VESA Local Bus

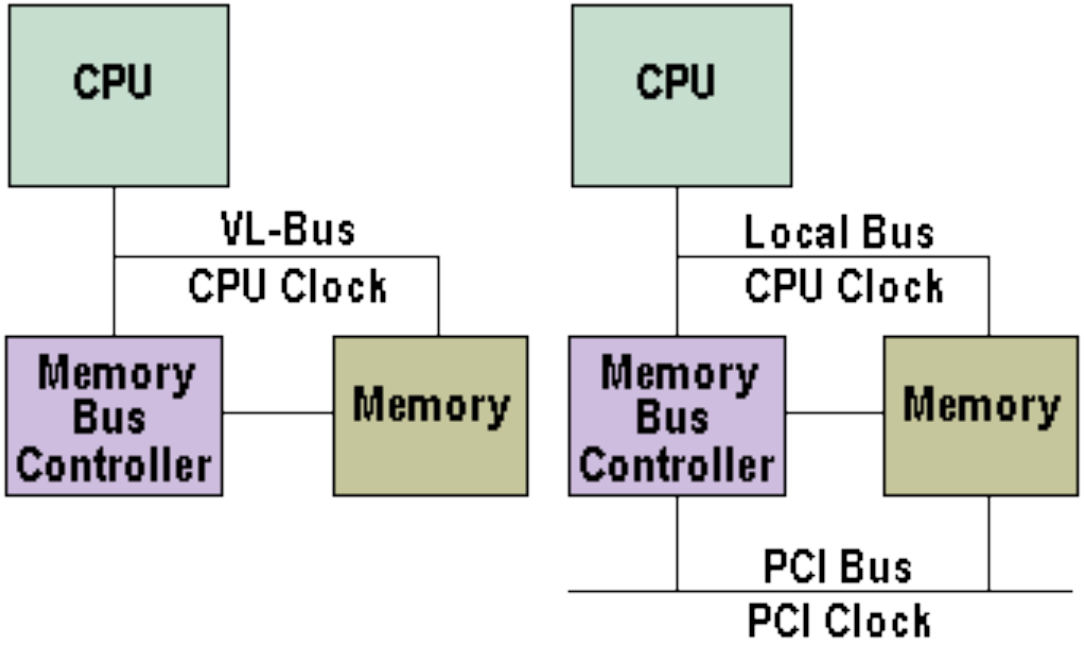
- Video Electronics Standards Association (VESA)
  - Introduced 1992
  - Took advantage of “local” CPU bus
  - 33 MHz (later 50 MHz in VLB 2.0)
  - 32-bits wide (later 64-bits in VLB 2.0)
  - Required I/O cards to run at same speed as processor
  - Introduction of Pentium led to death of VLB

PC Buses

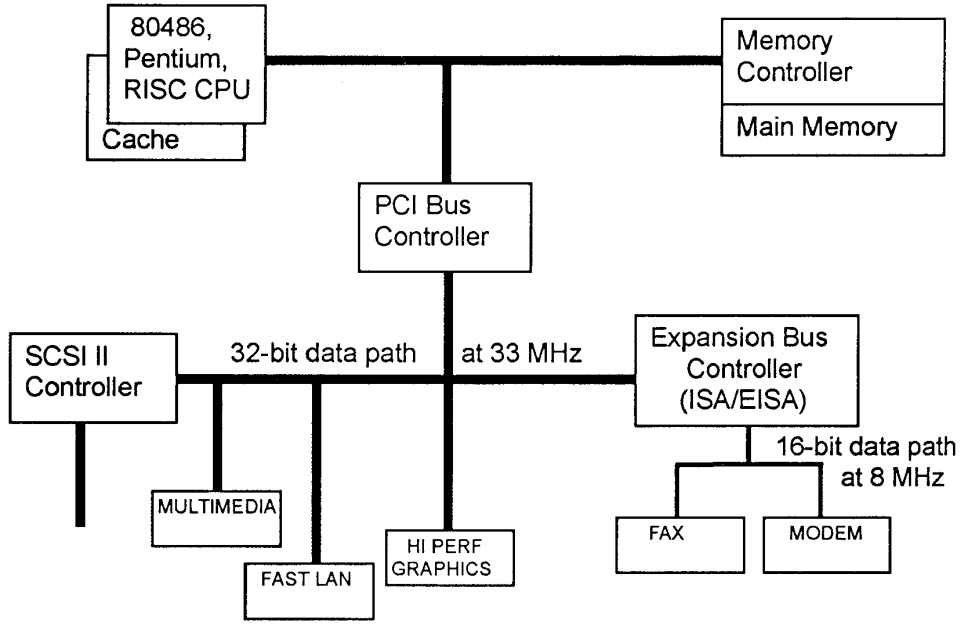
- Legacy Local/System Buses
  - PC Bus (ISA 8-bit)
  - IBM MCA (Micro Channel Architecture)
  - ISA (PC/AT)
  - EISA
  - VESA Local Bus (VLB)
- Contemporary Local/System Buses
  - PCI
  - PCI-X
  - AGP (no new designs)
  - Front Side Bus (Intel)
- I/O Buses and Others
  - USB 2.0, 3.0
  - IDE
  - SCSI
  - FireWire
  - PCMCIA
  - Infiniband
  - HyperChannel
  - PCI-Express (PCI-e)
  - RapidIO
  - Quickpath

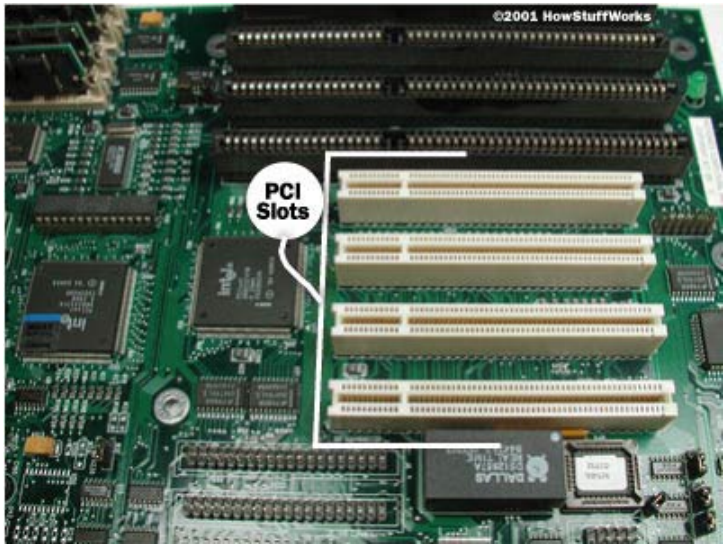
PCI Bus

- Peripheral Component Interconnect
- Introduced by Intel (1992)
- Turned over to PCI SIG
- Not Intel Microprocessor Specific
- Most common contemporary PC bus



Early PCI System Architecture





PCI Devices

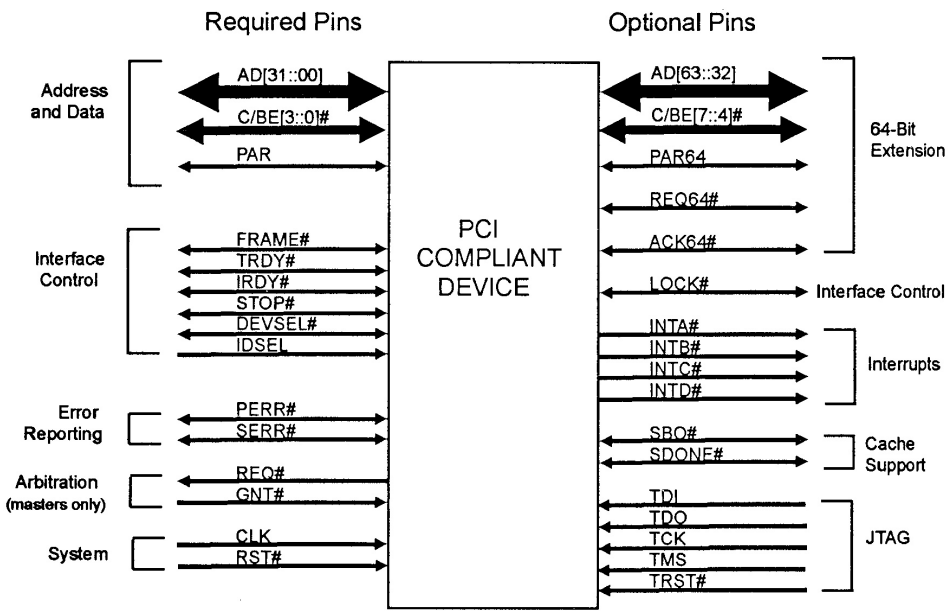
- Device Enumeration
  - Bus:Device:Function
    - Buses numbered sequentially from 0
    - Every device has at least function 0 (1..7 optional)
- Access to PCI Devices
  - Memory space
  - I/O space
  - Configuration space
    - 256 bytes configuration space per function

PCI Transactions

- Every transaction involves two agents*
  - Initiator (bus master) "bus mastering" – DMA!
  - Target (bus slave)
- A device can snoop without participating in a transaction
- Transactions
  - Three types: {I/O, Memory, Configuration}
  - Transaction Phases
    - Arbitration ("hidden")
    - One Address Phase
    - One or more Data Phases
      - All transactions are essentially bursts (1,2,4 bytes/phase)

* Except "Special Cycles" which are broadcast cycles

Simplified PCI Pin Out

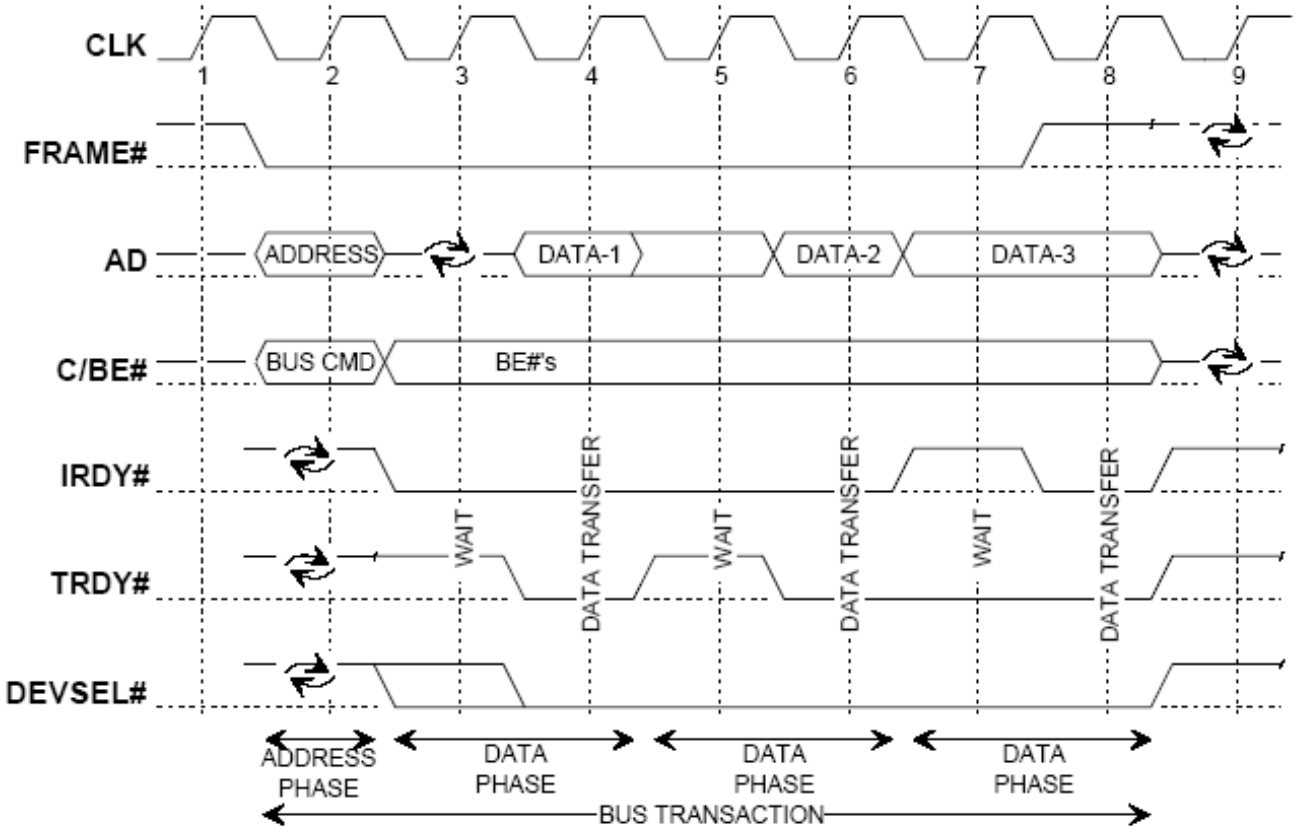


Signal	Function
CLK	Positive edge sampled. 33/66 MHz, 133 for PCI-X
AD[31:0]	Multiplexed address and data
FRAME#	H → L start of new transaction L → H following data phase is last
C/BE[3:0]#	Multiplexed command and byte enable
DEVSEL#	Asserted by target device when its address detected
IRDY#	Initiator ready
TRDY#	Target ready
STOP#	Request from target to terminate current transaction
LOCK#	Used by PCI bridge to ensure atomicity of commands
REQ#	Request bus ownership
GNT#	Grant bus ownership
PAR	Parity for AD[31:0] and C/BE[3:0]#. Even parity. Signal delayed by one cycle
PERR#	Parity error detected
RST#	Reset all registers to initial state
INTA# - INTD#	Level sensitive interrupt lines (permits sharing)
JTAG	Group includes TCK, TDI, TDO, TMS, TRST#

PCI Command Decoding

C/BE[3:0]	PCI Command
0000	Interrupt Acknowledge
0001	Special Cycle (Broadcast)
0010	I/O Read
0011	I/O Write
0100	Reserved
0101	Reserved (meaning in PCI-X)
0110	Memory Read
0111	Memory Write
1000	Reserved (meaning in PCI-X)
1001	Reserved (meaning in PCI-X)
1010	Configuration Read
1011	Configuration Write
1100	Memory Read Multiple
1101	Dual Address Cycle
1110	Memory Read Line
1111	Memory Write and Invalidate

Read Transaction



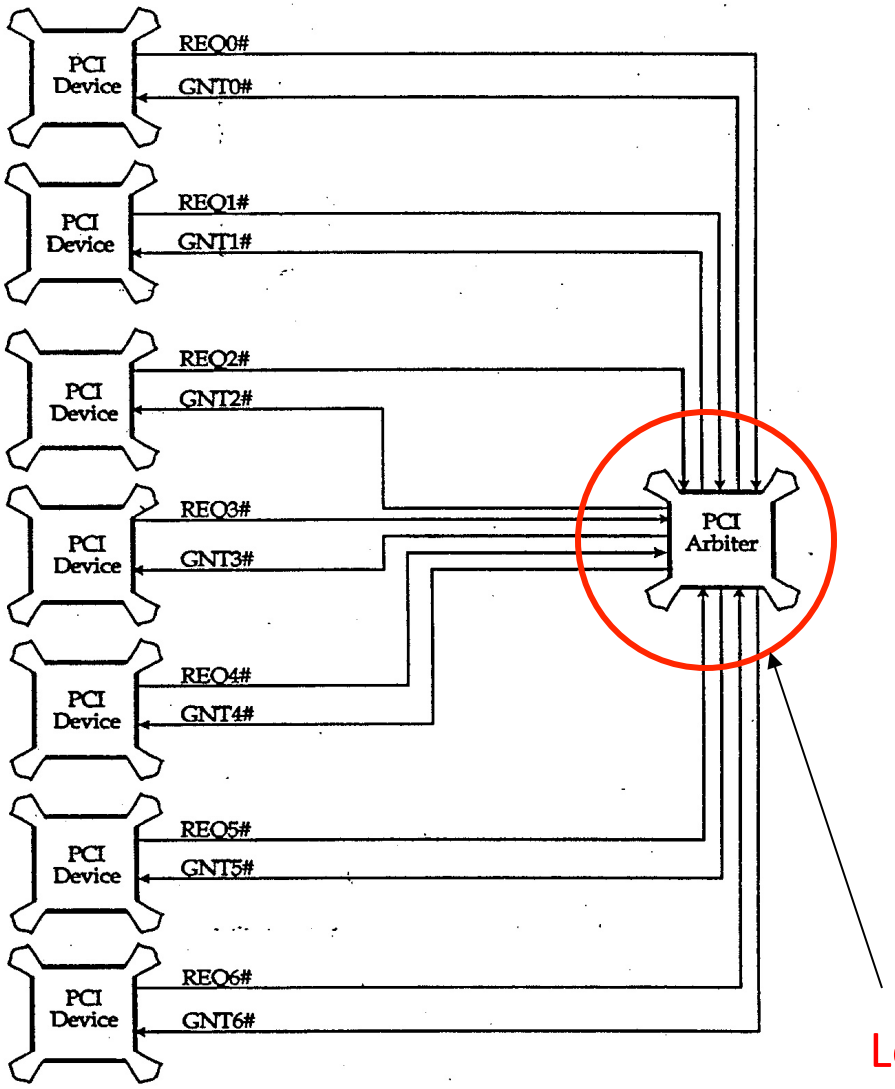
DEVSEL# must be asserted by target within 1 to 3 cycles after address phase

TRDY# must be asserted by target within 16 cycles after address phase

Read Transaction Details

- Cycle 1 - The bus is idle. (FRAME and IRDY de-asserted)
- Cycle 2 - The initiator asserts a valid address and places a read command on the C/BE# signals. This is the address phase.
- Cycle 3 - The initiator tri-states the address in preparation for the target driving read data. The initiator now drives valid byte enable information on the C/BE# signals. The initiator asserts IRDY# low indicating it is ready to capture read data. The target asserts DEVSEL# low (in this cycle or the next) as an acknowledgment it has positively decoded the address. The target drives TRDY# high indicating it is not yet providing valid read data.
- Cycle 4 - The target provides valid data and asserts TRDY# low indicating to the initiator that data is valid. IRDY# and TRDY# are both low during this cycle causing a data transfer to take place. The initiator captures the data. This is the first data phase.
- Cycle 5 - The target deasserts TRDY# high indicating it needs more time to prepare the next data transfer.
- Cycle 6 - The second data phase occurs as both IRDY# and TRDY# are low. The initiator captures the data provided by the target.
- Cycle 7 - The target provides valid data for the third data phase, but the initiator indicates it is not ready by deasserting IRDY# high.
- Cycle 8 - The initiator re-asserts IRDY# low to complete the third data phase. The initiator captures the data provided by the target. The initiator drives FRAME# high indicating this is the final data phase (master termination).
- Cycle 9 - FRAME#, AD, and C/BE# are tri-stated, as IRDY#, TRDY#, and DEVSEL# are driven inactive high for one cycle prior to being tri-stated.

PCI Arbitration



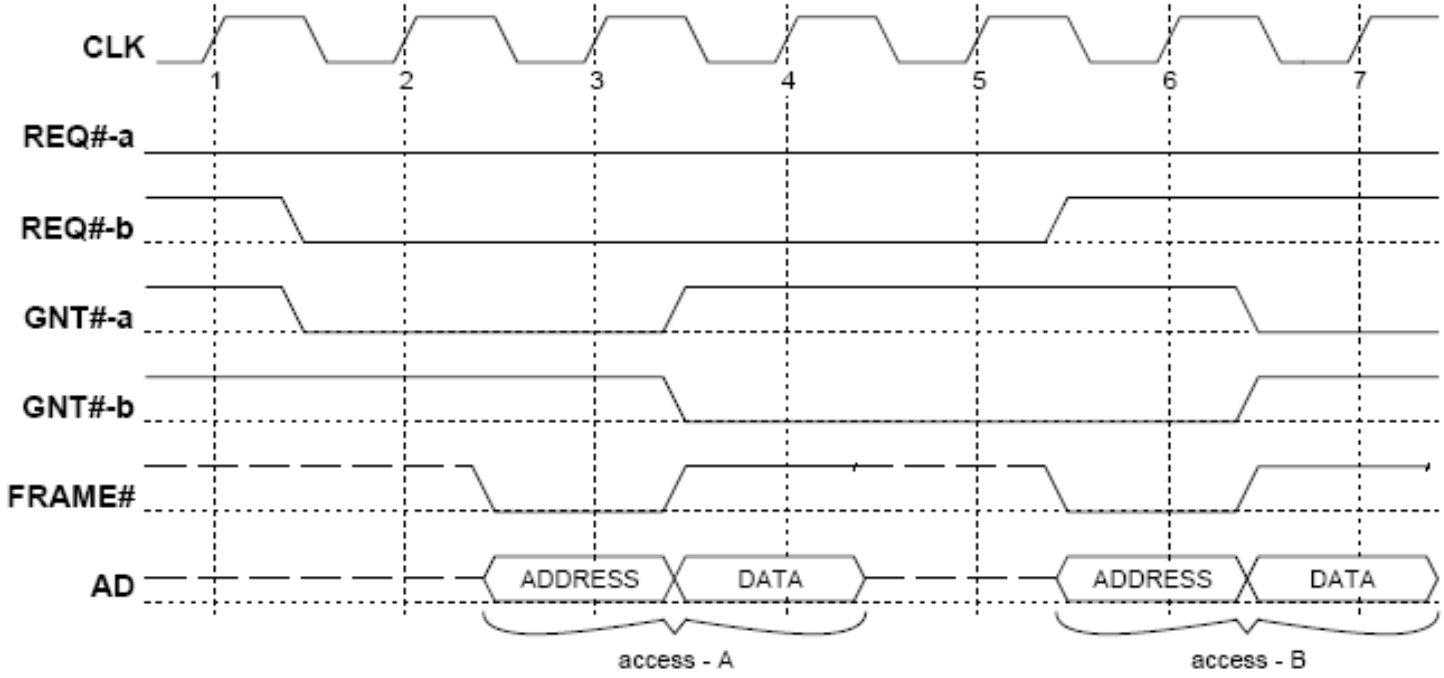
Details of arbitration policy not specified

Factors in priority and device latency timer

Hidden arbitration reduces latency

Located in PC Chipset (e.g. Southbridge)

Basic Arbitration

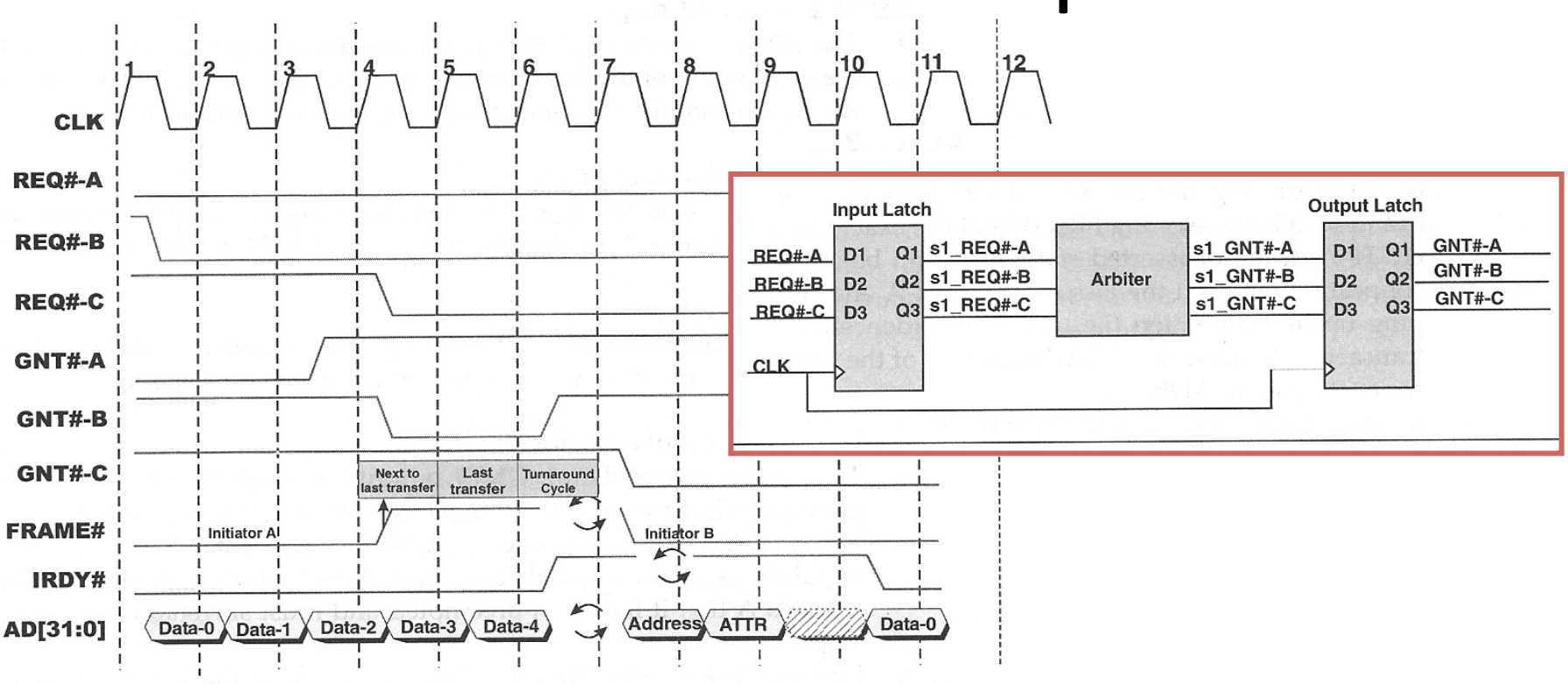


Device a has bus (and continues to assert REQ because it anticipates wanting to do further transactions)

Device b requests bus

Coincidentally, device a is through with the first of its transactions

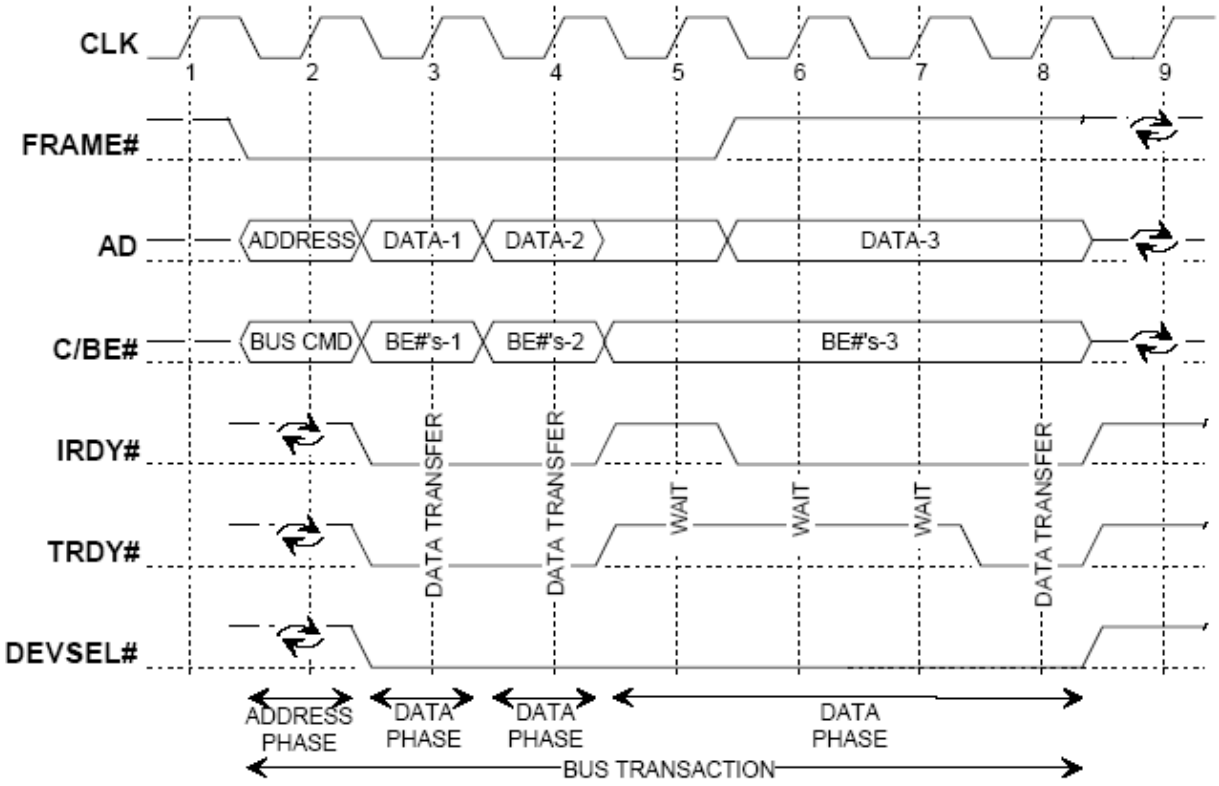
Arbitration Example



Each device (capable of being a bus master) maintains programmable latency timer. When a device is initiator it decrements the counter each cycle after asserting FRAME#. If it loses its GNT# signal during the transfer and the counter has reached its limit then the device must de-assert FRAME# and yield the bus to another device. If the limit has not been reached, it can continue until it has reached the limit.

In addition, during configuration each device (capable of being a bus master) reports its maximum permissible bus access grant delay (Max_Lat) and the minimum time it must have control over the bus (Min_Gnt). These are taken into account by the arbiter (in an unspecified manner).

Write Transaction



Write Transaction Details

- Cycle 1 - The bus is idle.
- Cycle 2 - The initiator asserts a valid address and places a write command on the C/BE# signals. This is the address phase.
- Cycle 3 - The initiator drives valid write data and byte enable signals. The initiator asserts IRDY# low indicating valid write data is available. The target asserts DEVSEL# low as an acknowledgment it has positively decoded the address (the target may not assert TRDY# before DEVSEL#). The target drives TRDY# low indicating it is ready to capture data. The first data phase occurs as both IRDY# and TRDY# are low. The target captures the write data.
- Cycle 4 - The initiator provides new data and byte enables. The second data phase occurs as both IRDY# and TRDY# are low. The target captures the write data.
- Cycle 5 - The initiator deasserts IRDY# indicating it is not ready to provide the next data. The target deasserts TRDY# indicating it is not ready to capture the next data.
- Cycle 6 - The initiator provides the next valid data and asserts IRDY# low. The initiator drives FRAME# high indicating this is the final data phase (master termination). The target is still not ready and keeps TRDY# high.
- Cycle 7 - The target is still not ready and keeps TRDY# high.
- Cycle 8 - The target becomes ready and asserts TRDY# low. The third data phase occurs as both IRDY# and TRDY# are low. The target captures the write data.
- Cycle 9 - FRAME#, AD, and C/BE# are tri-stated, as IRDY#, TRDY#, and DEVSEL# are driven inactive high for one cycle prior to being tri-stated.

Termination by Target

- Retry
  - STOP# is asserted before target ever asserts TRDY# (before any data is transmitted). Typically because target is unable to present any data within the allowed time period (16 cycles)
- Disconnect
  - STOP# is asserted by the target during or after first data phase. Master will repeat the transaction but with modified start address
- Target Abort
  - STOP# is asserted simultaneously with deactivation of DEVSEL#. Typically an unrecoverable error.
- Retry terminations result in "delayed transactions"
  - Typically used by slower devices
  - The target terminates the transaction with the initiator but continues to internally process the request, storing the results in a buffer
  - When the initiator retries the transaction, the target provides the requested data

PCI Configuration

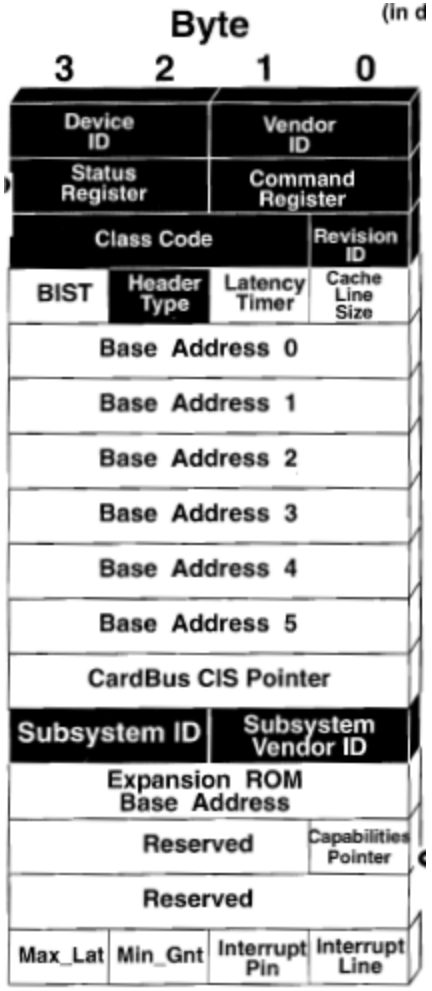
During configuration…

Enumeration determines PCI devices and functions connected, and their key characteristics, including amount of address space they require.

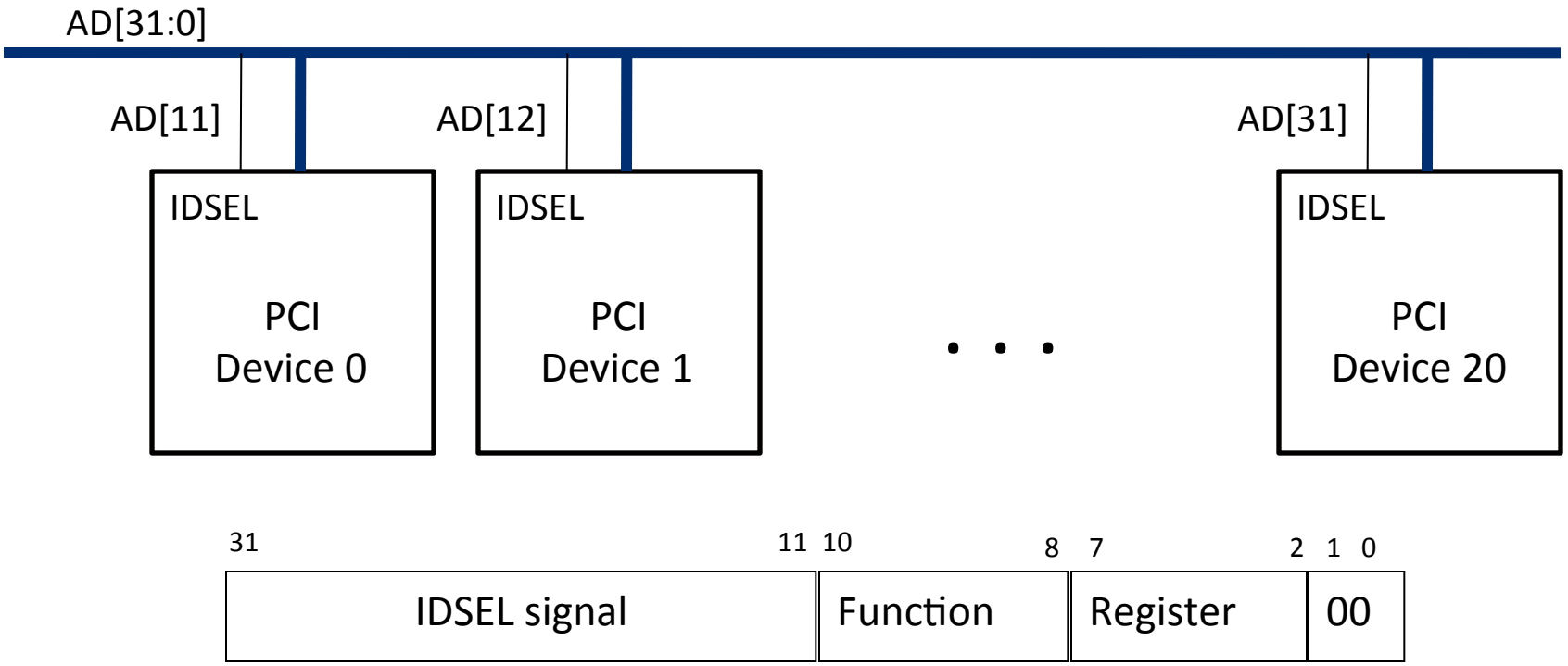
Each device (capable of being a bus master) also reports its maximum permissible bus access grant delay (Max_Lat) and the minimum time it must have control over the bus (Min_Gnt). These are taken into account by the arbiter (in an unspecified manner).

Each device (capable of being a bus master) maintains a programmable latency timer (set by the BIOS or OS after considering requirements of all devices). When a device is initiator it decrements the counter each cycle after asserting FRAME#. If it loses its GNT# signal during the transfer and the counter has reached its limit then the device must de-assert FRAME# and yield the bus to another device. If the limit has not been reached, it can continue until it has reached the limit.

The BIOS or OS also sets the base address for the address range the device will respond to.



Enumeration and Addressing Configurations



Address format

Bus Parking

- Arbiter can "park" the bus on a device when the bus is idle (no other devices requesting the bus)
- Device's GNT# is asserted (device does not assert REQ#)
- Device can gain ownership of bus immediately without cycle of delay
- Device must assert AD[31:0] and C/BE#[3:0]
- Usually parked on last device to be initiator
- Improves throughput by eliminating unnecessary arbitration cycles
- Avoids floating the bus (power consumption issues)

Selected PCI-X Enhancements

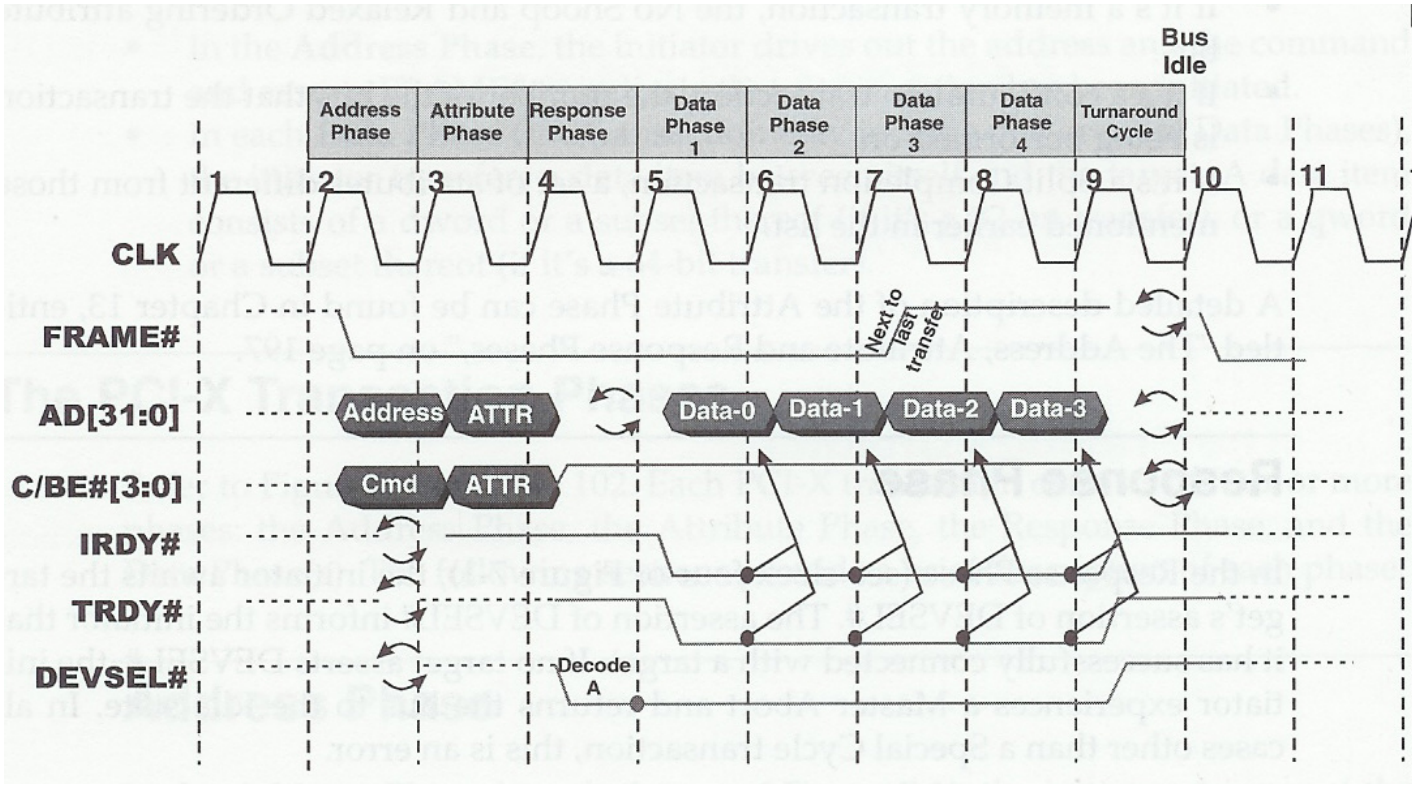
- Delayed Transactions consume bus bandwidth
  - Retry signaling
  - Repeated requests by initiator without knowing if target has data
- Split Transactions
  - If target would take >16 cycles to complete the request, it can signal a "Split Transaction" instead
  - Additional information supplied by requestor (initiator) during Attribute Phase
    - Requester ID (bus:device:function)
    - Transaction Number (Tag)

    } Sequence ID
  - Completer (target) initiates its own completion transaction when it has data ready (using Sequence ID as address)

Selected PCI-X Enhancements

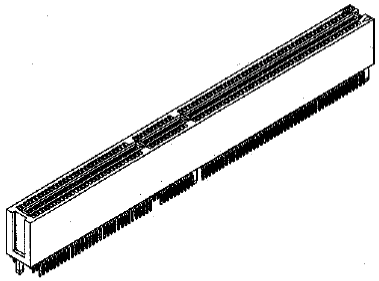
- PCI transactions were indeterminant length bursts of dwords (32-bits words)
  - PCI initiator terminates transfer by de-asserting FRAME#
  - PCI-X transactions are determinant length
    - Byte Count sent during Attribute Phase
- PCI permits both initiator and target to insert wait states
  - De-asserting IRDY# and TRDY#
  - PCI-X prohibits initiators from inserting wait states and only allows targets to insert wait states prior to the first data item
- PCI Implements Parity Error Detection on AD and C/BE#
  - PCI-X Implements ECC
    - Single bit error correction
    - Multi-bit error detection
- PCI-X Transactions have four phases
  - Address Phase
  - Attribute Phase
  - Response Phase
  - Data Phase(s)

PCI-X Read Transaction



PCI-X and PCI Compatibility

- Same connectors
  - Only one pin (B38) definition changes (GND to PCIXCAP)
- If PCI-X device detects PCI devices
  - Drops speed to match least capable device
  - Uses PCI protocol
- Achieves forward and backward compatibility



- PCB Bridges
  - Increase number of devices that can be connected
  - Divide PCI devices into segments so that all devices on segment have similar capabilities (PCI, PCI-X) and speed (66 MHz, 133 MHz) to optimize PCI bus bandwidth
  - Balancing bandwidth requirements

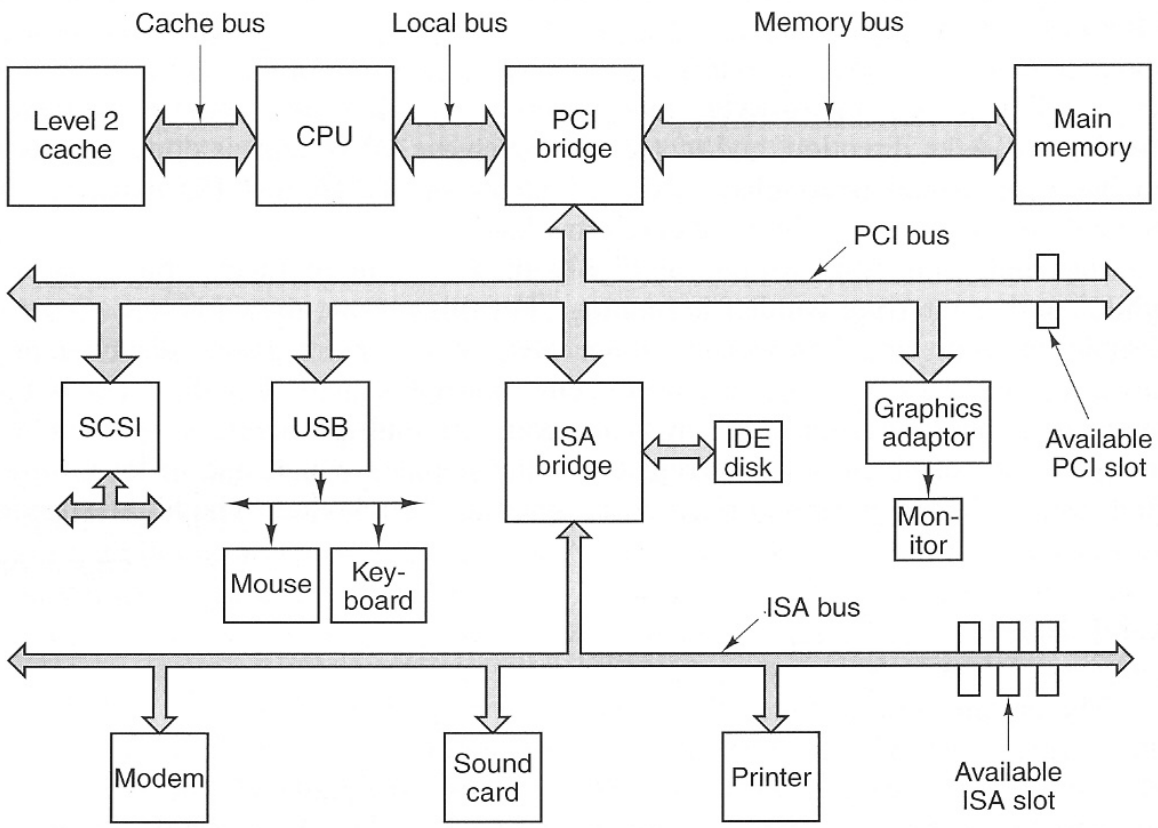
PCI Evolution

- PCI 1.0 (1992)
  - 32-bit, 33.3 MHz
- PCI 2.0 (1993)
  - 64-bit extensions
- PCI 2.1 (1995)
  - 66 MHz clock
- PCI 2.2 (1998)
- PCI 2.3 (2002)
- PCI 3.0

- PCI-X 1.0 (2002)
  - PCI-X 66
  - PCI-X 133
- PCI-X 2.0
  - PCI-X 66
  - PCI-X 133
  - PCI-X 266
  - PCI-X 533
    - 4.2 GB/s
      - (533 MHz x 64 bits/8)

PCI-X is backwards compatible with PCI

Early PCI to Host Interfaces



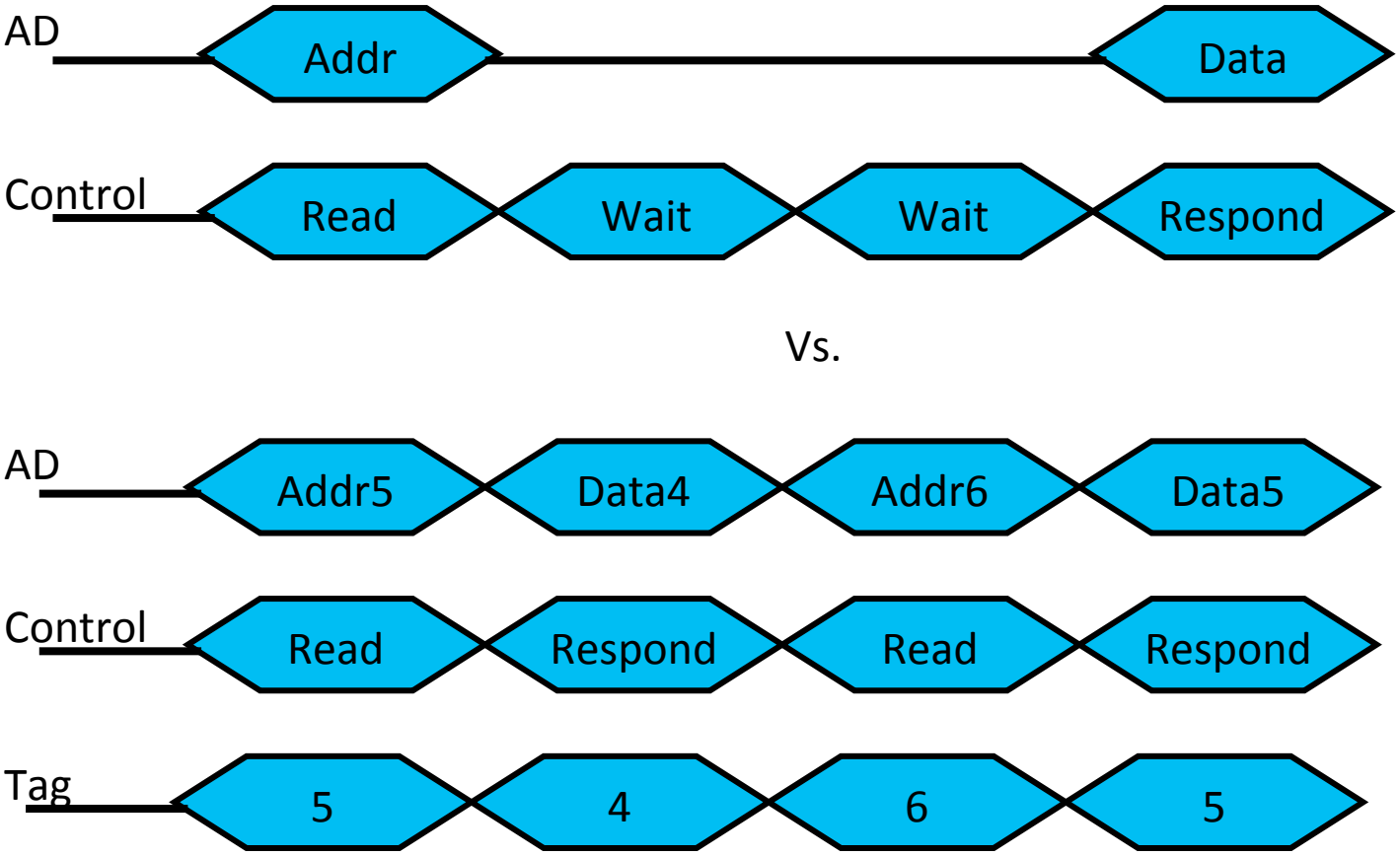
Increasing the Bus Bandwidth

- 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

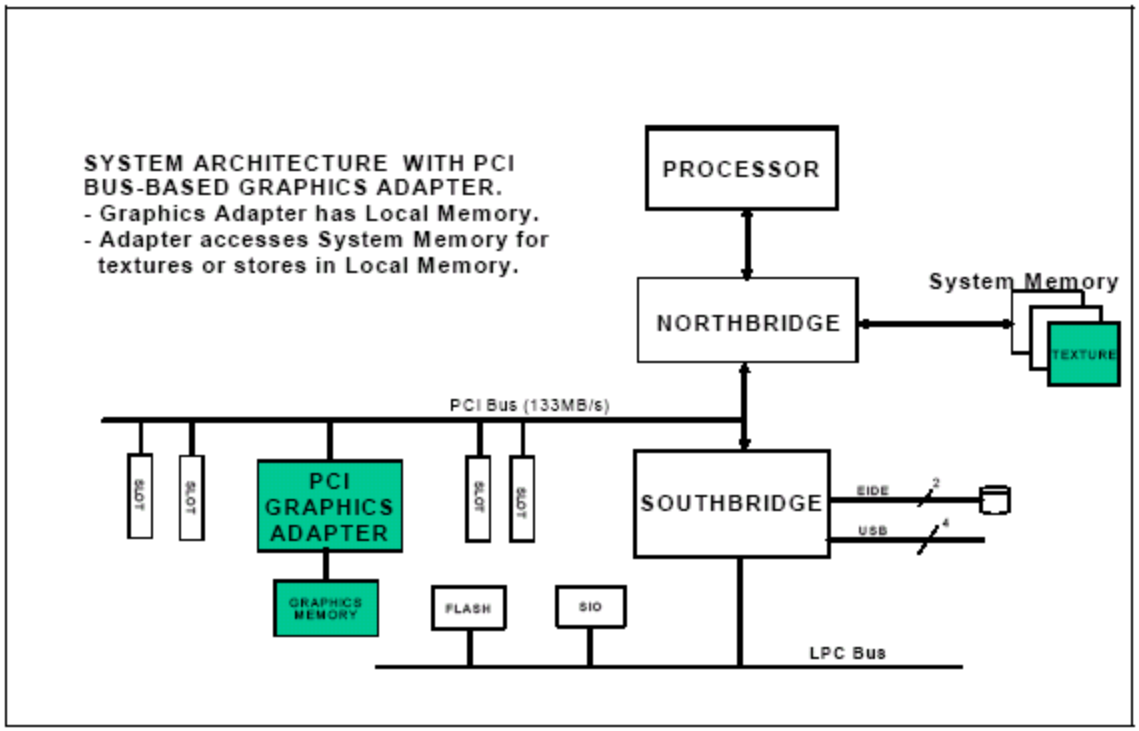
Increasing Transaction Rate on Multimaster Bus

- Overlapped arbitration
  - perform arbitration for next transaction during current transaction
- Bus parking
  - master holds onto bus and performs multiple transactions as long as no other master makes request
- Overlapped address/data phases
  - 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 most modern memory buses

Benefits of Split Transactions



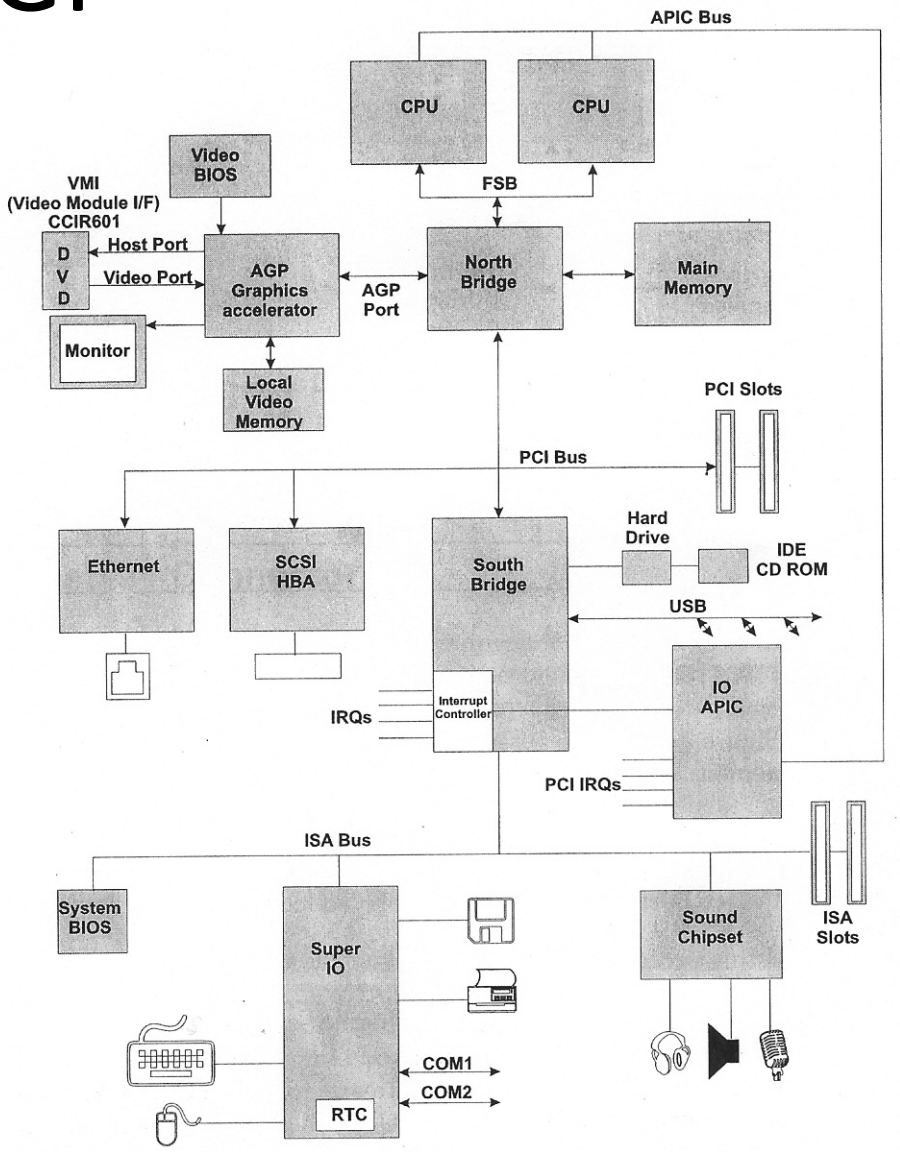
Old PCI-Based Graphics Adapters



- 3D graphics cards
  - Voracious appetite for memory
    - Textures: Gaming, 3D modeling
- Limitations to PCI

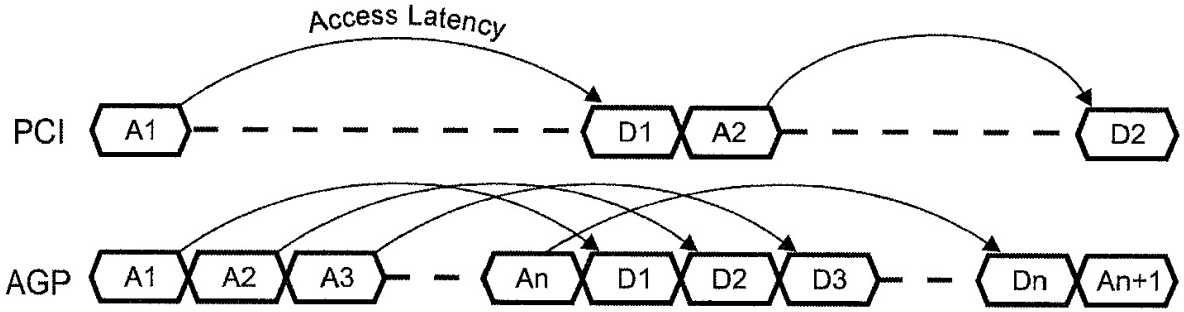
AGP

- Borrowed heavily from PCI 2.1
- Not Intel-specific
  - AMD, Apple!
- Point-to-point
  - Graphics card to memory
  - Through MCH (NorthBridge)
- Pipelined memory access
- 32-bits wide
- 66 MHz, 1x, 2x, 4x, 8x speeds
- Made obsolete by PCI Express



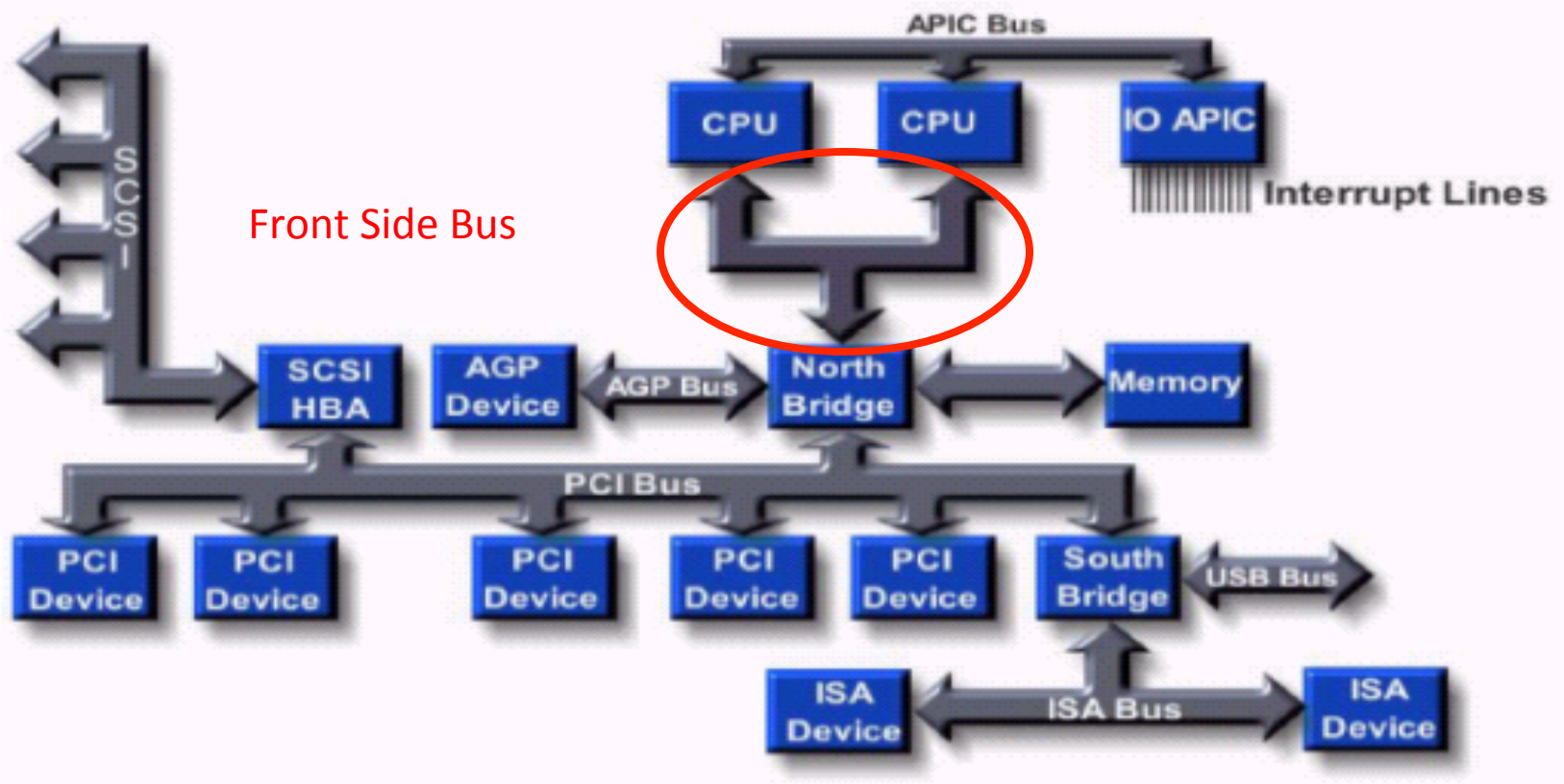
Pipelined Bus Transactions

- Pipelined Bus Transactions
  - Similar to microprocessor instruction pipeline
  - Improves throughput/bandwidth
  - Doesn't affect latency
- Multiple outstanding transactions
- Sideband Addressing
  - De-multiplex Address/Data
  - Transmit address byte-serially on 8-bit path
    - Can be used even when data bus is busy
- Constrains addresses to 8-byte boundaries



Intel Front Side Bus (FSB)

Typical Intel Multiprocessor System: Pentium, Core 2

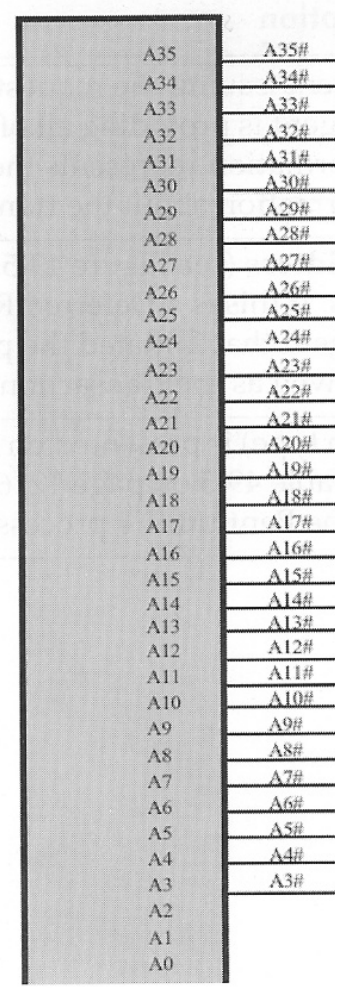


IA32 Memory Access

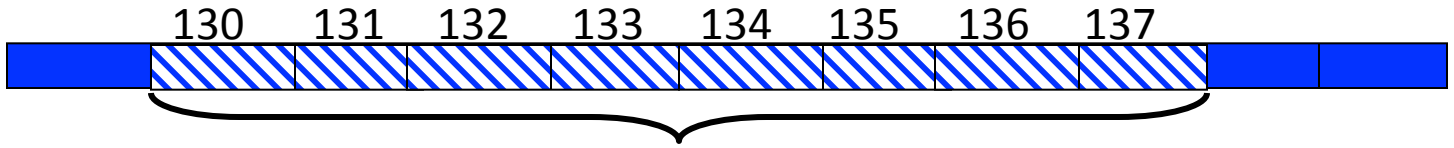
- Evolution towards larger physical address spaces
  - 16→32→36
- Wider data buses (64 bits)
- Quad-word aligned memory accesses
- Burst mode memory reads
  - Faster cache line fill, less overhead

Processor	Line Size
486	16 bytes
Pentium P6 Family	32 bytes
Pentium 4, Xeon	128 bytes
Pentium M	64 bytes

Quad word aligned access



- Pentium: 2^{36} (64 GB) physical address space
- Address quadword at a time (8 bytes)
- Don't need lower 3 address bits
- Need to specify which byte(s) interested in
- BE[7:0] (Byte Enables)



- Consider a reference to a mis-aligned quadword
  - e.g. 8 bytes beginning at location 134
  - Two quadword memory references
- Consider a reference to a mis-aligned word
  - e.g. 2 bytes beginning at location 137
  - Two quadword memory references

FSB Agent Types

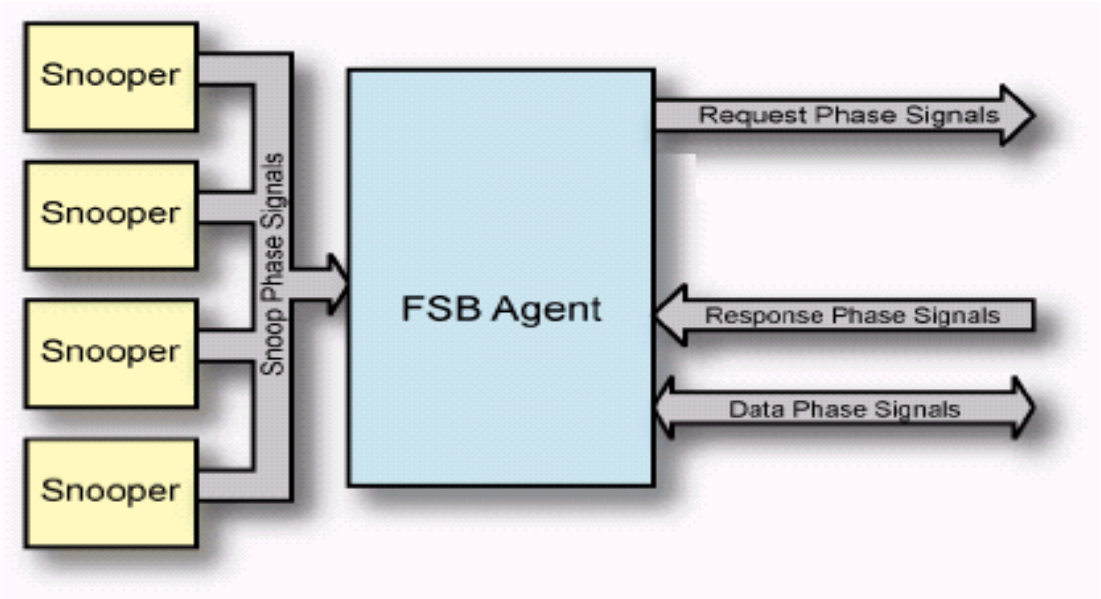
- Request
  - Transaction initiator (memory or I/O read/write)
  - Example: CPU
  - Example: MCH (North Bridge) on behalf of I/O device
- Response
  - Target of transaction (I/O or memory)
  - Example: Memory controller is “response only” agent
- Snoop
  - Any device with a cache (typically processor)

FSB Overview

- Synchronous
  - BCLK (differential)
- Split Transactions
- Separate Signal Groups
- Separate Address, Data Lines
- Pipelined Transactions

FSB Transaction Phases

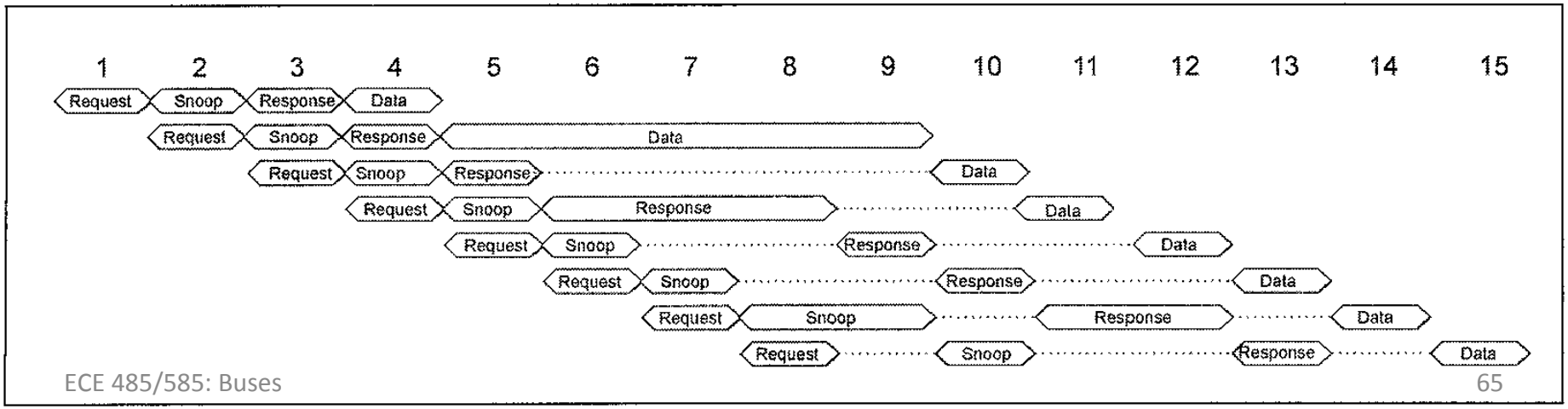
- Arbitration
- Error (eliminated in Pentium 4)
- Request
- Snoop
- Response
- Data



Signals required for each phase are separate, permitting pipelining

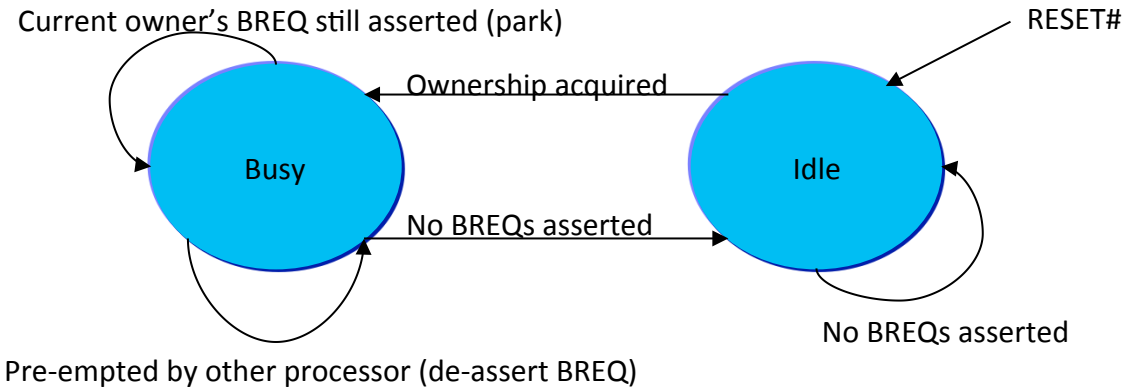
FSB Pipelining

- In-Order Queue (IOQ)
  - All agents must maintain a record of outstanding transactions
  - Removed when request response occurs
  - Depth of IOQ varies with device (Pentium 4: 12)
  - Assert BNR# (Block Next Request when IOQ full

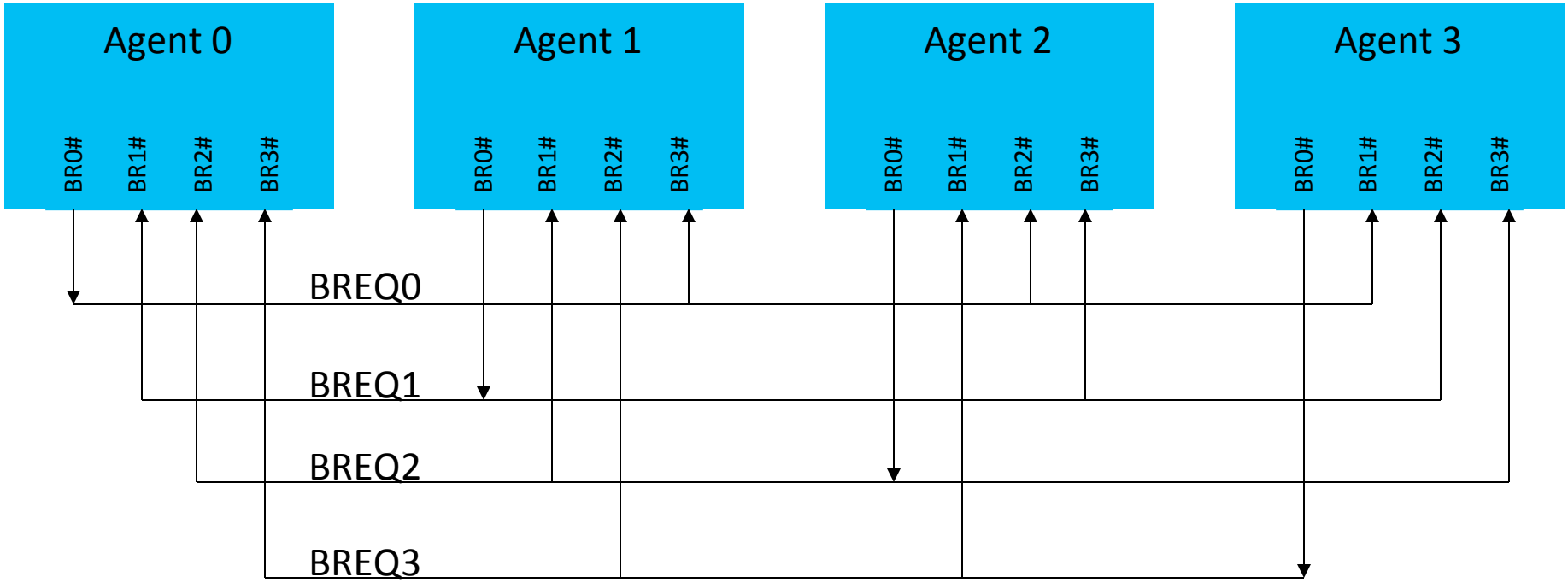


FSB Arbitration

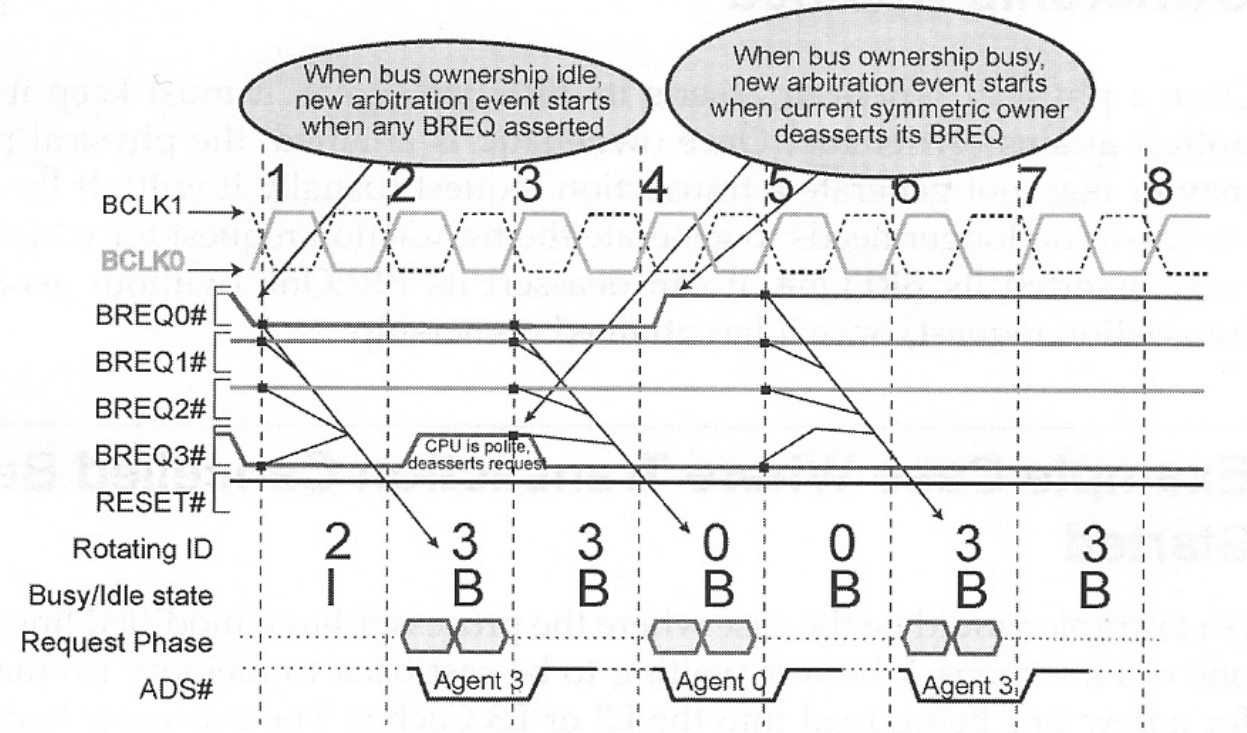
- Decentralized arbitration
  - Rotational priority {0,1,2,3,0,…}
- Each processor keeps track of
  - Whether it owns the Request Phase signal group
  - Which processor owned it last (or still does)
  - Which of them gets it next (assuming a request)
- Rotating ID
  - Reset to 3 to indicate last owner (next owner: 0)
  - Updated upon arbitration events (
- Busy/Idle Ownership States
  - *Busy:* Last owner still owns request phase signal group
  - *Idle:* Last owner has relinquished request phase signal group



BREQs for Symmetric Agents

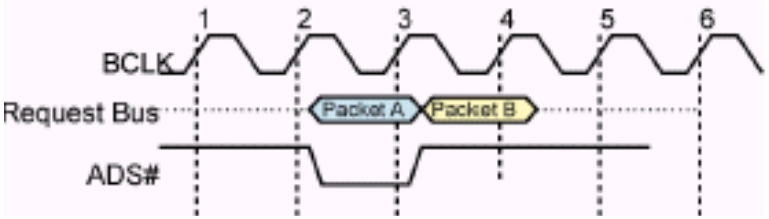


Arbitration Events: BREQ# Transitions



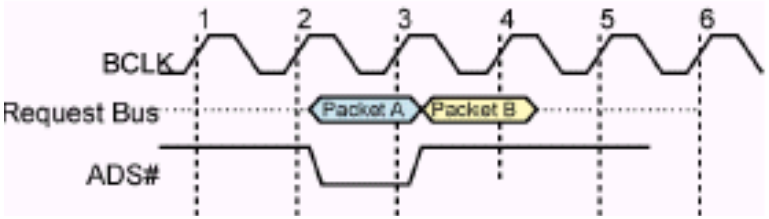
Arbitration events at clocks 1, 3, and 5
FSB agents sample BREQs on rising clock edge
Update their Rotating IDs during following cycle

Request Phase



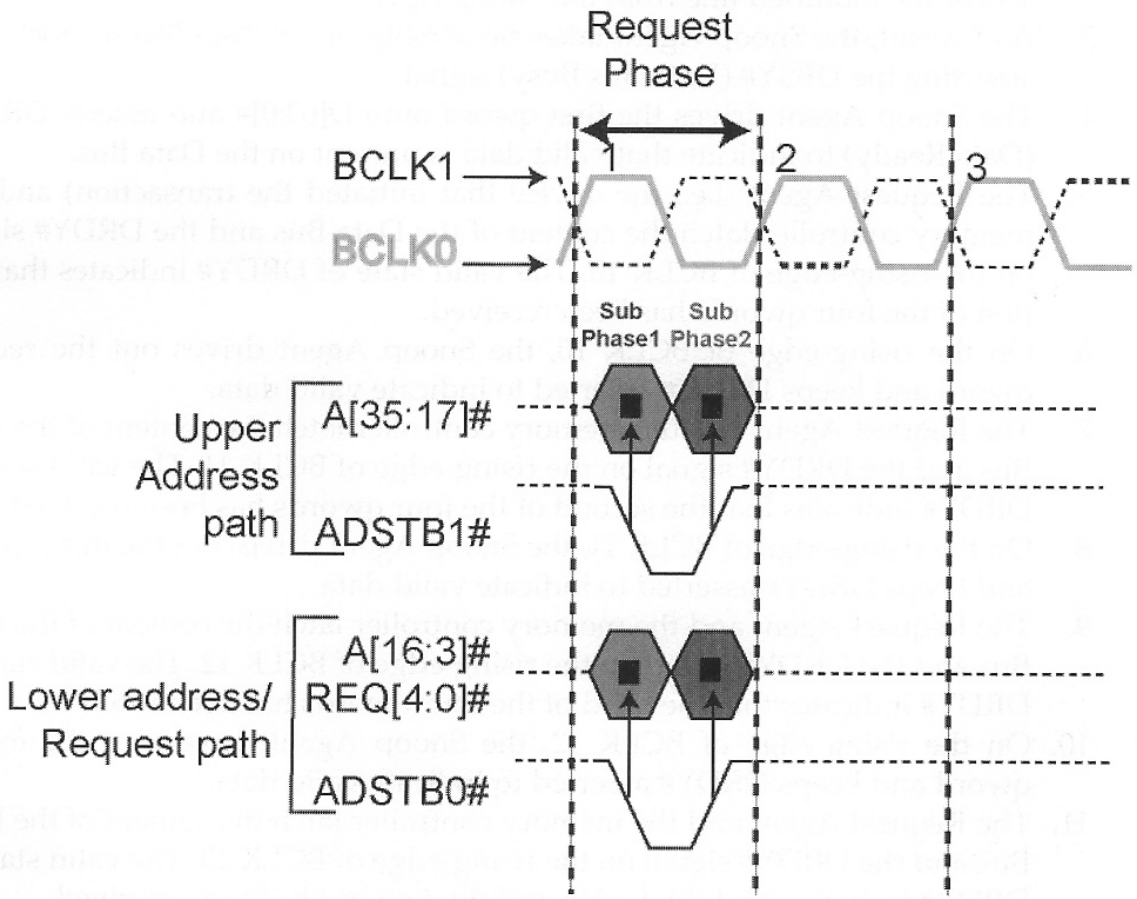
- Signals multiplexed
- Two Packets Sent Consecutively
  - Packet A
    - Sufficient information for both responding agent and snoopers to decode address and intent
      - Address type
      - Address
      - Request (e.g. Memory Read, Cache Line Fill)
  - Packet B
    - Additional information
      - Byte Enables
      - Attributes (from MTRRs: determine cache behavior)
      - Agent ID
      - Transaction ID
- Snooping takes place during Request Phase

Request Phase

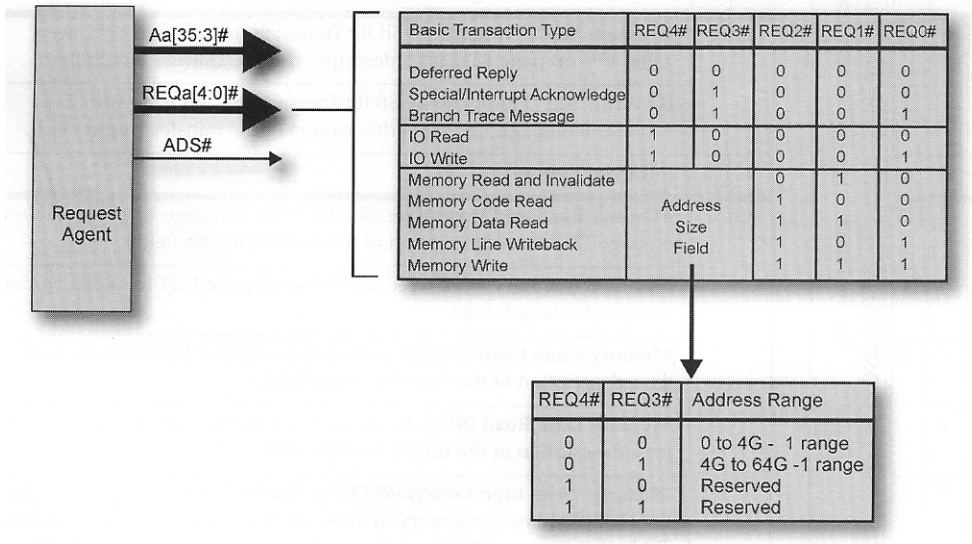


- ADS# (Address Strobe) signals new transaction
- ADSTB[1:0]# Source synchronous strobes
  - Latch packets (all Agents, not just Response Agents)
  - Each strobe responsible for subset of lines
    - ADSTB[0] – A[16:3]# and REQ[4:0]#
    - ADSTB[1] – A[35:17]#
  - Allows request to occur in single BCLK cycle
    - Falling edges latch Packet A
    - Rising edges latch Packet B
  - Provides all info for snoop to initiate in first packet

Request Phase



Pentium 4: Determining Transaction Address and Type Packet A



Basic Transaction Type	REQ4#	REQ3#	REQ2#	REQ1#	REQ0#
Deferred Reply	0	0	0	0	0
Special/Interrupt Acknowledge	0	1	0	0	0
Branch Trace Message	0	1	0	0	1
IO Read	1	0	0	0	0
IO Write	1	0	0	0	1
Memory Read and Invalidate	Address Size Field		0	1	0
Memory Code Read			1	0	0
Memory Data Read			1	1	0
Memory Line Writeback			1	0	1
Memory Write			1	1	1

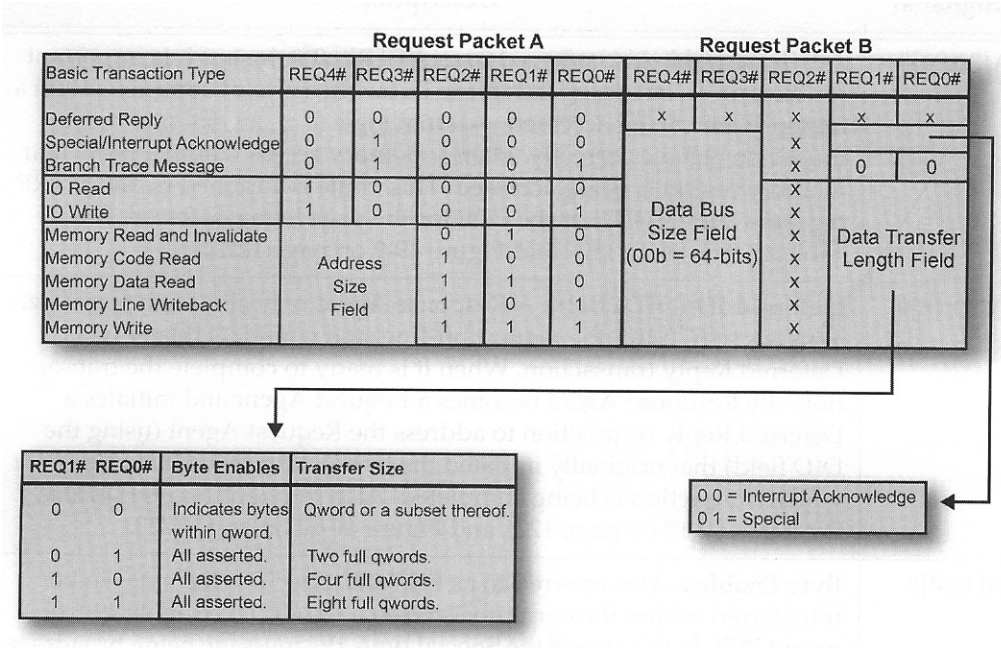
REQ4#	REQ3#	Address Range
0	0	0 to 4G - 1 range
0	1	4G to 64G -1 range
1	0	Reserved
1	1	Reserved

LEN[1:0]#	State of Byte Enables	Size of Transfer
00b	Specify the bytes to be transferred within the addressed qword.	Qword or subset of qword.
01b	All asserted.	16 bytes (two qwords).
10b	All asserted.	32 bytes (four qwords).
11b	Not defined.	64 bytes (eight qwords).

Table 49-7: Address Size Only Applies to Memory Transactions (see Table 49-5 on page 1216)

ASIZ[1:0]#	Description
00b	The memory address is within the 0 through 4GB - 1 range. It should be decoded by any memory targets (i.e., Response Agents) that reside (at least partially) below the 4GB boundary.
01b	The memory address is within the 4GB through 64GB - 1 range. It should be decoded by any memory targets (i.e., Response Agents) that reside (partially or fully) above the 4GB boundary (but below the 64GB boundary).
10b	Reserved.
11b	Reserved.

Pentium 4: Determining Transaction Size and Bytes Packet B



LEN[1:0]#	State of Byte Enables	Size of Transfer
00b	Specify the bytes to be transferred within the addressed qword.	Qword or subset of qword.
01b	All asserted.	16 bytes (two qwords).
10b	All asserted.	32 bytes (four qwords).
11b	Not defined.	64 bytes (eight qwords).

Table 49-7: Address Size Only Applies to Memory Transactions (see Table 49-5 on page 1216)

ASIZ[1:0]#	Description
00b	The memory address is within the 0 through 4GB - 1 range. It should be decoded by any memory targets (i.e., Response Agents) that reside (at least partially) below the 4GB boundary.
01b	The memory address is within the 4GB through 64GB - 1 range. It should be decoded by any memory targets (i.e., Response Agents) that reside (partially or fully) above the 4GB boundary (but below the 64GB boundary).
10b	Reserved.
11b	Reserved.

Snoop Phase

- Snoop Response Occurs During Snoop Phase
- Request Agent Samples Snoop Result Group
- Currently-Addressed Response Agent May Assert DEFER# to Indicate its Intention to Issue “Retry” or “Defer” Response During Response Phase

Response Phase

- Response Agent Delivers its Response
- Request Agent Samples Response Group
- Variety of Responses
  - Retry
  - Hard Failure (no retry – machine check exception occurs)
  - If Write Response Agent will Accept Data in Data Phase
  - If Read Response Agent will Supply Data in the Data Phase
  - If Snooping Agent Indicated Hit on Modified Line (HITM#)
    - Snooper will transfer entire line to memory
    - If Read Transaction, Request Agent will get data then (implicit writeback)
  - Deferred
    1. Response Agent Instructs Request Agent to end transaction without data
    2. Response Agent will obtain requested data or deliver the write data later
    3. Deferred Reply Transaction

Data Phase

- DBSY# (Data Bus Busy)
  - Response Agent must monitor before asserting
  - Indicates taking ownership of Data Phase Signal Group
- DRDY# (Data Ready)
  - Valid data is being driven onto data bus
- DB[63:0]# (Data Bus)
- DSTBN[3:0]#, DSTBP[3:0]# (Data Strobe)
  - Source Synchronous data strobes for data
  - [I] Responsible for Group I (4 groups of 16 lines)
  - N/P negative and positive strobes
- RS[2:0] Response (Type) – Normal, Defer, etc

64-Byte Transfer on Pentium 4 FSB

Why strobes per data group?

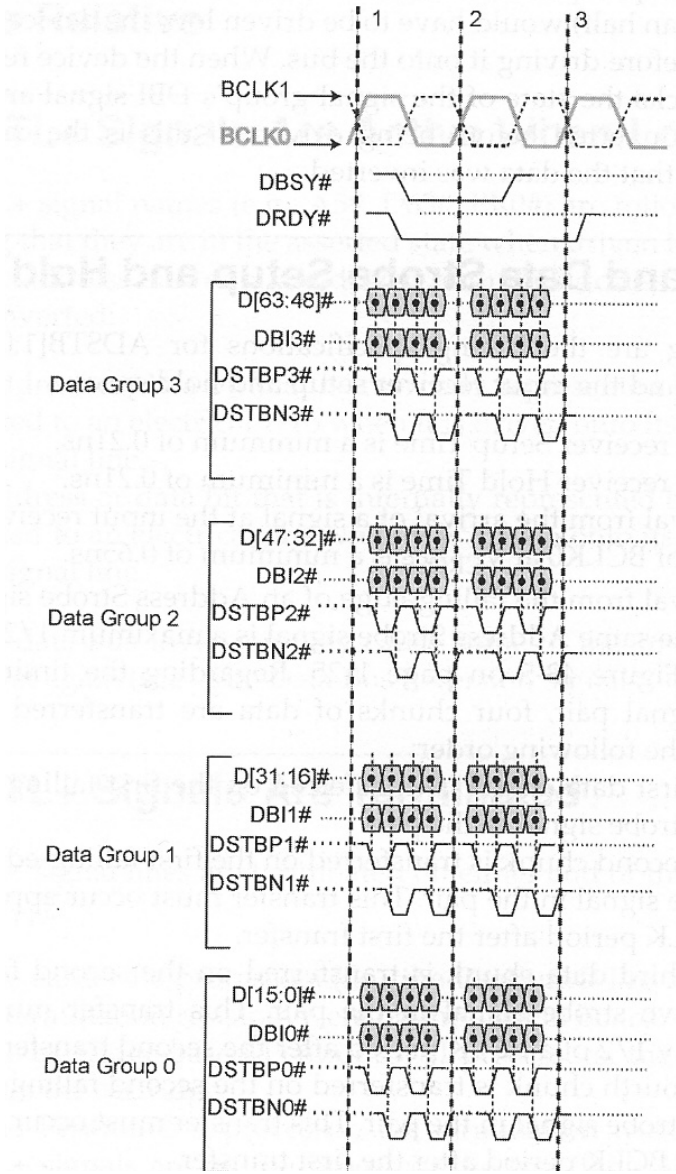
Trace routing: match path length

Why multiple (P/N) strobes?

Avoid using + and – edges

Uses – edges of alternating signals

Less jitter between like edges



Quad Pumped:

4 data transfer periods per clock

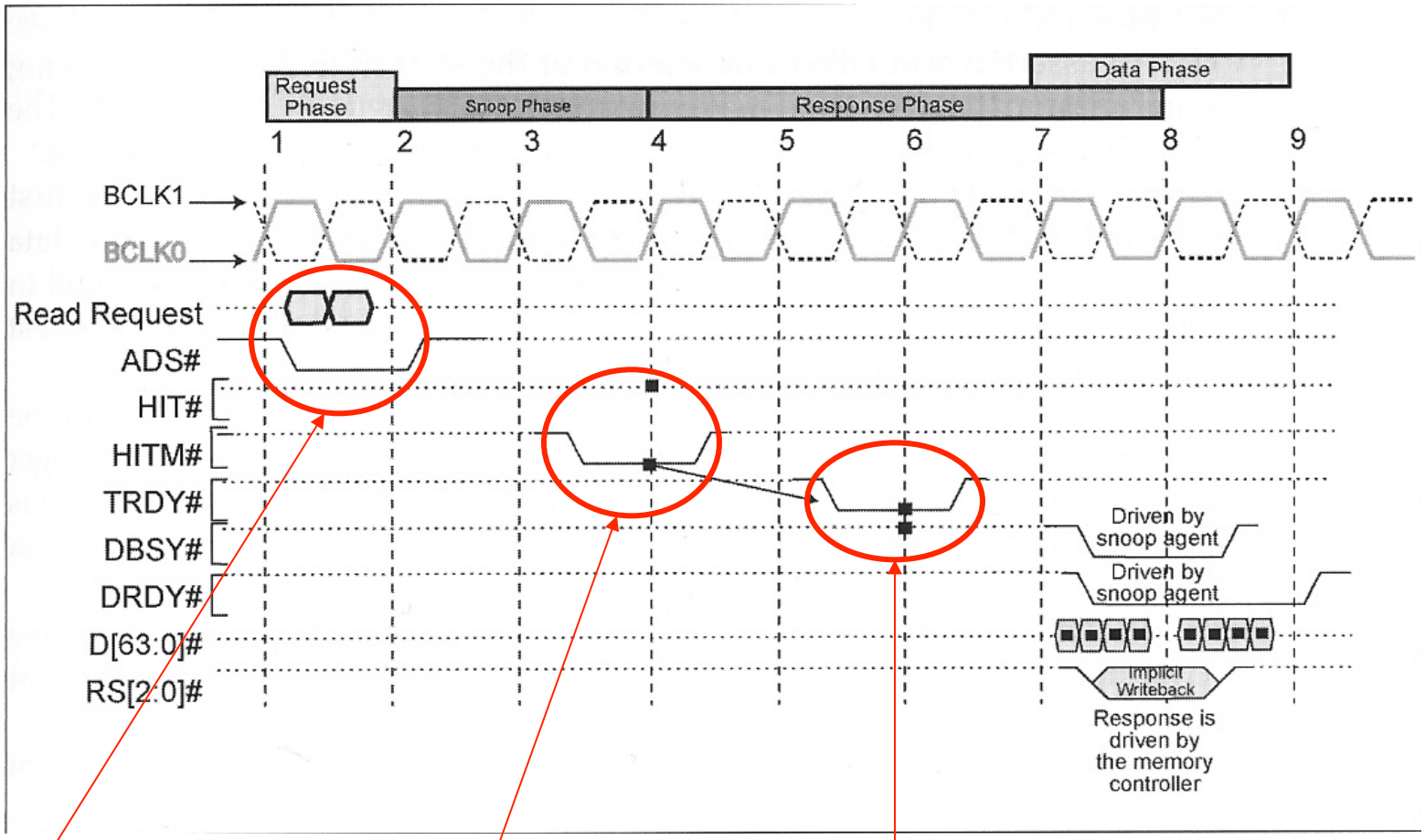
8 bytes/xfer x 4 xfers/clock x 100 MHz

3.2GB/s

[speeds: Intel: 100,133,200,266 MHz]

[PowerPC, AMD Opteron faster]

Read that Hits a Modified Line



Request agent

Snoop agent

Response agent (memory controller)

Not shown: source synchronous data strobes

Concurrency and Replication

"Money can buy bandwidth… latency requires bribing God"
-- attributed to David Clark

- Replication can increase bandwidth and decrease latency
  - Split instruction and data cache
    (parallel access to both during single clock)
- Concurrency can increase bandwidth and latency
  - Pipelining
    (initial latency longer but greater bandwidth)
- Replication can decrease latency without directly affecting bandwidth
  - Increase size of cache
- Replication can increase bandwidth without directly affecting latency
  - 64-bit instead of 32-bit bus

Speeding up buses

- Increase clock rate (synchronous)
- Increase data width
- Use bursts (less protocol overhead)
- De-multiplex (e.g. address/data)
- Partition the bus into segments
  - Balance bandwidth requirements
  - Move slower (longer latency) devices to another bus
- Split transactions (utilize latency for other transactions)
- Exploit Concurrency
  - Pipelining
  - Hidden arbitration