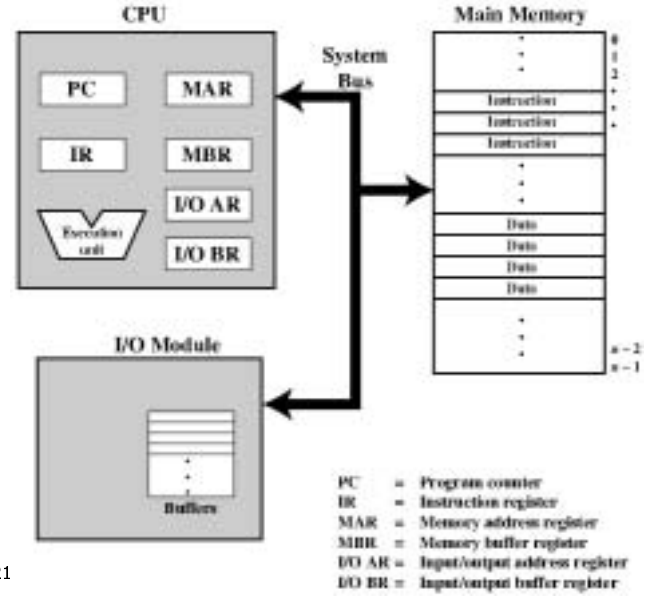
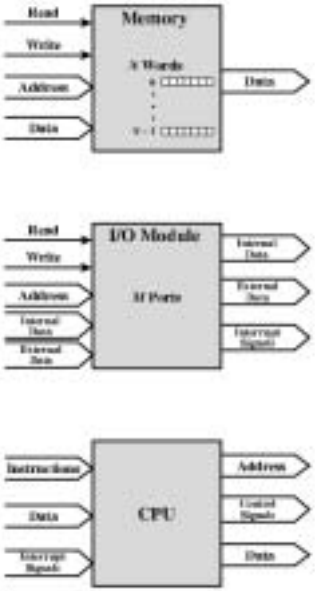
L8: Buses & Bus Design

Computer Components: Top Level View



Computer Modules



Memory Connection

- Receives and sends data
- Receives addresses (of locations)
- Receives control signals
  - Read
  - Write
  - Timing

Input/Output Connection(1)

- Similar to memory from computer's viewpoint
- Output
  - Receive data from computer
  - Send data to peripheral
- Input
  - Receive data from peripheral
  - Send data to computer

Input/Output Connection(2)

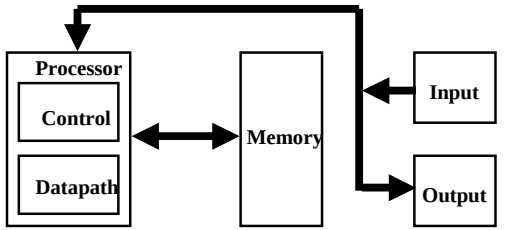
- Receive control signals from computer
- Send control signals to peripherals
  - e.g. spin disk
- Receive addresses from computer
  - e.g. port number to identify peripheral
- Send interrupt signals (control)

CPU Connection

- Reads instruction and data
- Writes out data (after processing)
- Sends control signals to other units
- Receives (& acts on) interrupts

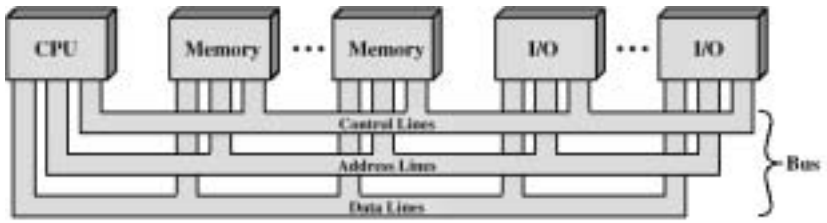
A Bus is:

- 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

Bus Interconnection Scheme



Data Bus

- Carries data
  - Remember that there is no difference between "data" and "instruction" at this level
- Width is a key determinant of performance
  - 8, 16, 32, 64 bit

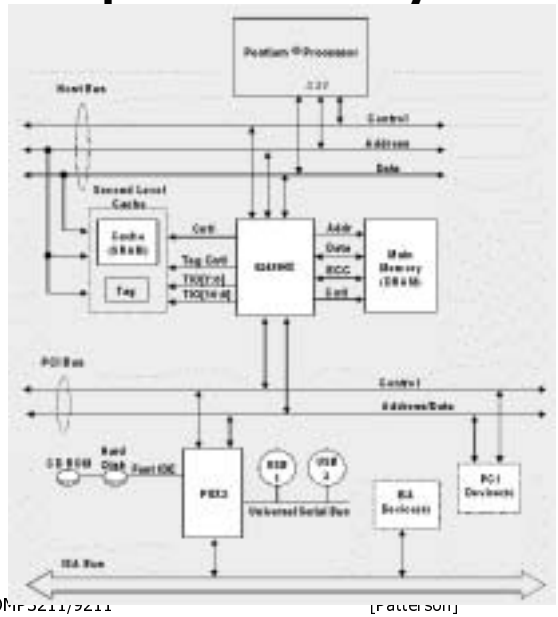
Address bus

- Identify the source or destination of data
- e.g. CPU needs to read an instruction (data) from a given location in memory
- Bus width determines maximum memory capacity of system
  - e.g. 8080 has 16 bit address bus giving 64k address space

Control Bus

- Control and timing information
  - Memory read/write signal
  - Interrupt request
  - Clock signals

Example: Pentium System Organization

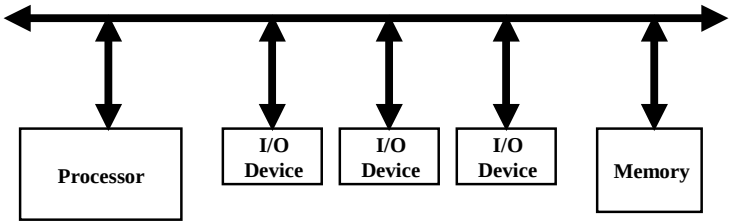


Processor/Memory Bus

PCI Bus

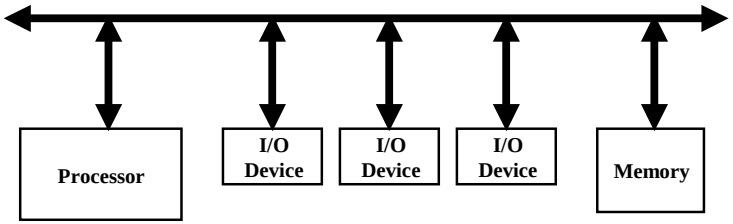
I/O Busses

Advantages of Buses



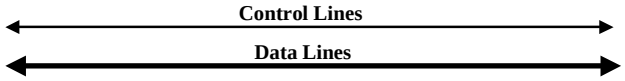
- Versatility:
  - New devices can be added easily
  - Peripherals can be moved between computer systems that use the same bus standard
- Low Cost:
  - A single set of wires is shared in multiple ways
- Manage complexity by partitioning the design

Disadvantage of Buses



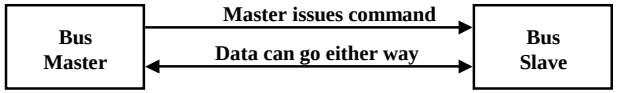
- It creates a communication bottleneck
  - The bandwidth of the bus can limit the maximum I/O throughput
- The maximum bus speed is largely limited by:
  - The length of the bus
  - The number of devices on the bus
  - The need to support a range of devices with:
    - Widely varying latencies
    -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

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

Master versus Slave

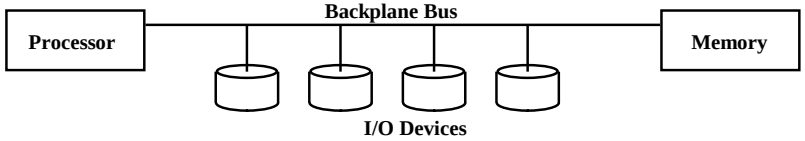


- A bus transaction includes two parts:
  - Issuing the command (and address) – request
  - Transferring the data – action
- Master is the one who starts the bus transaction by:
  - issuing the command (and address)
- Slave is the one who responds to the address by:
  - Sending data to the master if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

Types of Buses

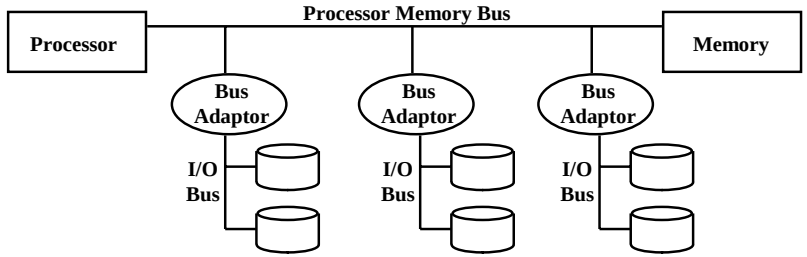
- Processor-Memory Bus (design specific)
  - Short and high speed
  - Only need to match the memory system
    - Maximize memory-to-processor bandwidth
  - Connects directly to the processor
  - Optimized for cache block transfers
- I/O Bus (industry standard)
  - Usually is lengthy and slower
  - Need 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

A Computer System with One Bus: Backplane Bus



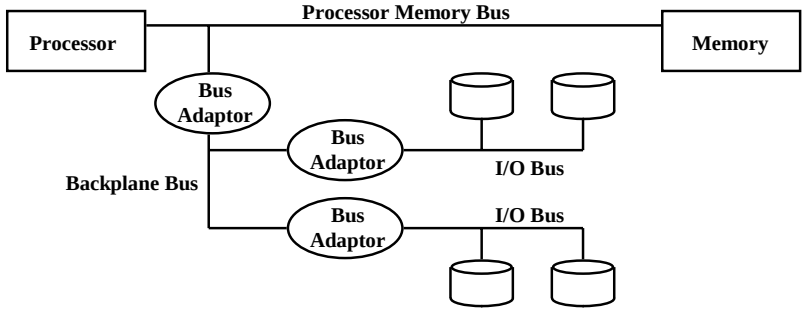
- A single bus (the backplane bus) is used for:
  - Processor to memory communication
  - Communication between I/O devices and memory
- Advantages: Simple and low cost
- Disadvantages: slow and the bus can become a major bottleneck
- Example: IBM PC - AT

A Two-Bus System



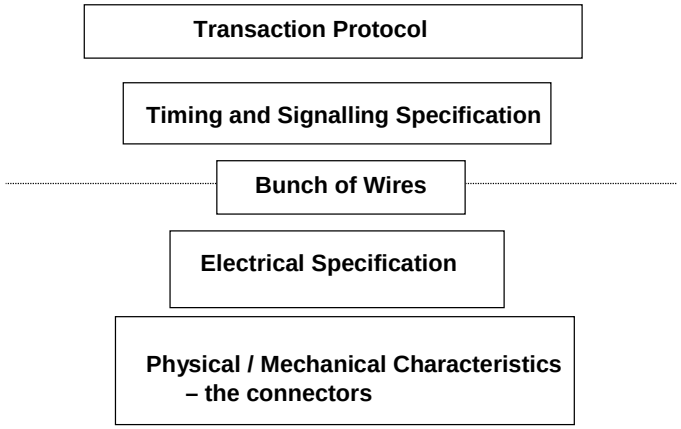
- I/O buses tap into the processor-memory bus via bus adaptors:
  - Processor-memory bus: mainly for processor-memory traffic
  - I/O buses: provide expansion slots for I/O devices
- Apple Macintosh-II
  - NuBus: Processor, memory, and a few selected I/O devices
  - SCCI Bus: the rest of the I/O devices

A 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

What defines a bus?



Principle Bus Design Issues

- The speed and bandwidth of a bus are influenced by four main factors:
  - the bus width
  - the bus clocking scheme
  - the bus arbitration method, and
  - the bus operation

Bus Width

- The size of the memory that can be addressed is exponential in the number of bus address lines
- However, wide buses need more wires, take up more space, and need bigger connectors. Hence they are more expensive. As a consequence system designers tend to be short-sighted about the size needed (e.g. 8088 had a 20-bit address bus, the 80286 added an additional 4-bit address bus with control, and 80386 added another 8-bit address bus with additional control)
- Data buses also tend to grow over time to provide additional bandwidth. The only other means by which bandwidth can be increased, speed, introduces the problem of bus skew as the signals on the many wires travel at slightly different speeds
- Designers often multiplex data and address signals in different phases of a transaction onto the one set of wires, thereby reducing width but also performance.

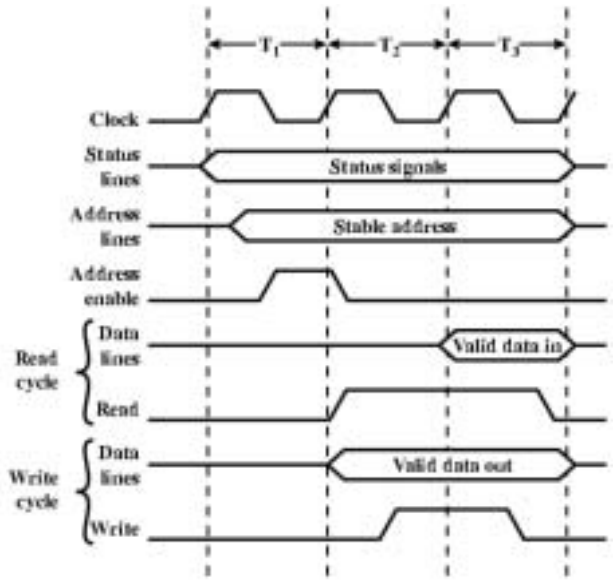
Increasing the Bus Bandwidth

- 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

Bus Clocking: 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

Synchronous Timing Diagram



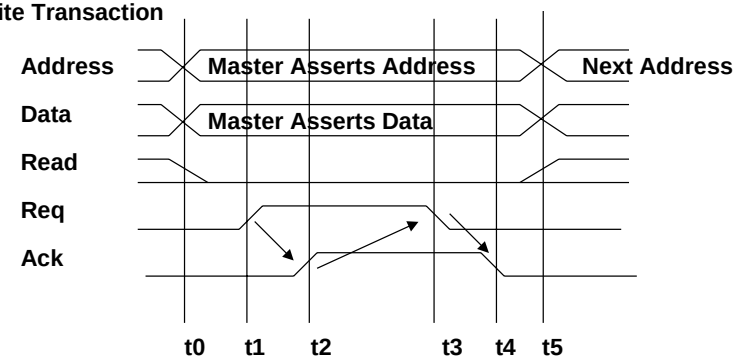
Synchronous bus operation

- Depending upon the protocol, signals undergo transitions and are latched on either rising or falling clock edges – in this example, values are sampled on rising clock edges

1. Status signals may be asserted to indicate the bus is in use – that a transaction is about to take place
2. The bus master drives an address onto the address lines and asserts an address enable line to indicate to the other devices that a transaction address is available
3. For read transactions the master indicates data is to be read, and in this case, the slave must respond within the third cycle so that the master may latch the result
4. For write transactions, the master places the data onto the bus and asserts a write control signal when the slave is to latch the data

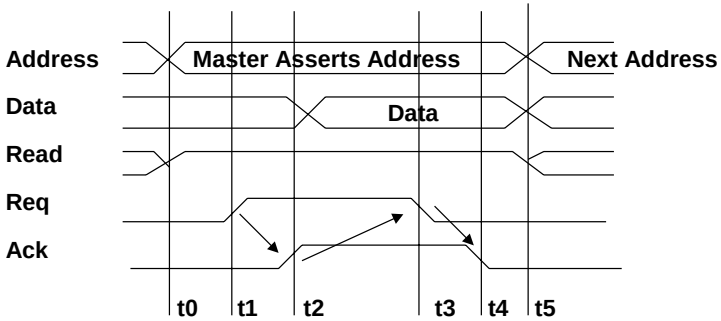
Asynchronous Handshake

Write Transaction



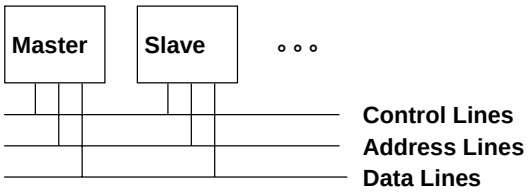
- t0 : Master has obtained control and asserts address, direction, data
  Waits a specified amount of time for slaves to decode target
- t1: Master asserts request line
- t2: Slave asserts ack, indicating data received
- t3: Master releases req
- t4: Slave releases ack

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

Buses 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.

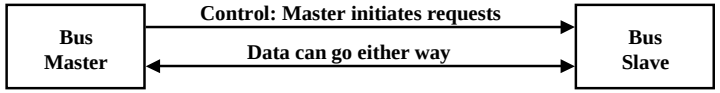
Asynchronous Bus Transfers: control lines (req, ack) serve to orchestrate sequencing.

Synchronous Bus Transfers: sequence relative to common clock.

Bus Transaction

- Arbitration
- Request
- Action

Arbitration: Obtaining Access to the Bus

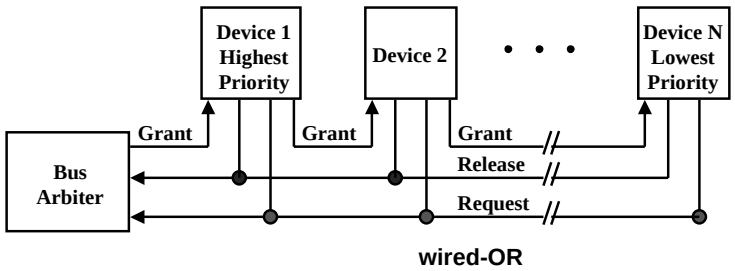


- One of the most important issues in bus design:
  - How is the bus reserved by a devices that wishes to use it?
- Chaos is avoided by a master-slave arrangement:
  - Only the bus master can control access to the bus:
    - It initiates and controls all bus requests
  - A slave responds to read and 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

Multiple Potential Bus Masters: the need for 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 the slide after next
  - 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.

The Daisy Chain Bus Arbitration Scheme

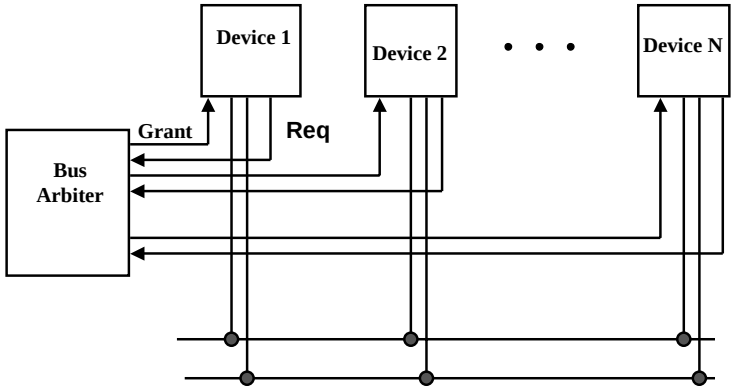


- Advantage: simple
- Disadvantages:
  - Cannot assure fairness:
    A low-priority device may be locked out indefinitely
  - The use of the daisy chain grant signal also limits the bus speed

Daisy Chain Arbitration Protocol

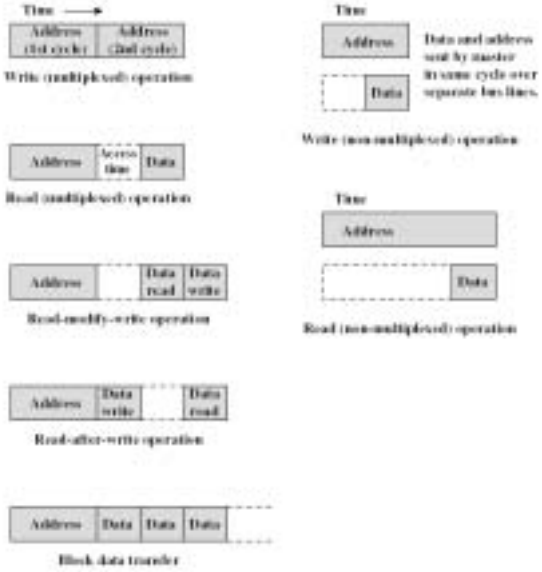
- Signal the request line
- Wait for a L→H transition on the grant line indicating that the bus is being reassigned
- Intercept the grant signal, and do not allow lower-priority devices to see it. Stop asserting the request line
- Use the bus
- Signal that the bus is no longer required by asserting the release line
- Fairness can be improved by not allowing a device to reassert the request line until it sees the bus request line go low.
- Some bus systems use multiple daisy chains with separate request and grant lines for each daisy chain and a priority encoder to select from among the requests on multiple request lines.

Centralized Parallel Arbitration



- Used in essentially all processor-memory busses and in high-speed I/O busses e.g. PCI
- Can use any one of a few arbitration schemes e.g. round robin, priority, etc.

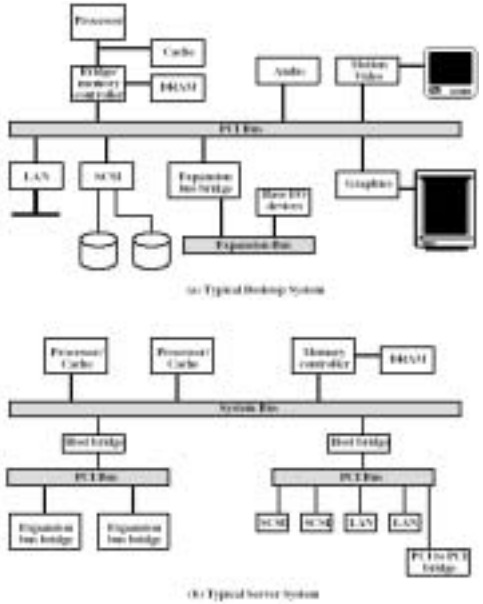
Transaction types



Example: Peripheral Component Interconnect (PCI) Bus

- High-bandwidth, processor-independent bus
- Current standard allows use of up to 64 data lines at 66 MHz for a raw transfer rate of 528 MB/s or 4.224 Gbps
- Initial work done by Intel in 1990 for Pentium-based systems; Intel released patents into public domain and promoted the creation of an industry association to further develop and maintain compatability of PCI specs
- PCI thus widely adopted
- PCI supports a variety of microprocessor-based configurations including single- and multi-processor systems
- PCI thus provides a general-purpose set of functions

Example PCI Configurations



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 BUS CMD and ADDRESS
- Data transfers happen when
  - IRDY# asserted by master when ready to transfer data
  - TRDY# asserted by target when ready to transfer data
- FRAME# deasserted when master intends to complete only one more data transfer

Mandatory PCI Signal lines (1)

Designation	Type	Description
System Pins		
CLK	in	Provides timing for all transactions and is sampled by all inputs on the rising edge. Clock rates up to 33 MHz are supported.
RST#	in	Forces all PCI-specific registers, sequencers, and signals to an initialized state.
Address and Data Pins		
AD[31::0]	t/s	Multiplexed lines used for address and data
C/BE[3::0]#	t/s	Multiplexed bus command and byte enable signals. During the data phase, the lines indicate which of the four byte lanes carry meaningful data.
PAR	t/s	Provides even parity across AD and C/BE lines one clock cycle later. The master drives PAR for address and write data phases; the target drive PAR for read data phases.

Mandatory PCI Signal lines (2)

Interface Control Pins		
FRAME#	s/t/s	Driven by current master to indicate the start and duration of a transaction. It is asserted at the start and deasserted when the initiator is ready to begin the final data phase.
IRDY#	s/t/s	Initiator Ready. Driven by current bus master (initiator of transaction). During a read, indicates that the master is prepared to accept data; during a write, indicates that valid data are present on AD.
TRDY#	s/t/s	Target Ready. Driven by the target (selected device). During a read, indicatest that valid data are present on AD; during a write, indicates that target is ready to accept data.
STOP#	s/t/s	Indicates that current target wishes the initiator to stop the current transaction.
IDSEL	in	Initialization Device Select. Used as a chip select during configuration read and write transactions.
DEVSEL#	in	Device Select. Asserted by target when it has recognized its address. Indicates to current initiator whether any device has been selected.

Mandatory PCI Signal lines (3)

Arbitration Pins		
REQ#	t/s	Indicates to the arbiter that this device requires use of the bus. This is a device-specific point-to-point line.
GNT#	t/s	Indicates to the device that the arbiter has granted bus access. This is a device-specific point-to-point line.
Error Reporting Pins		
PERR#	s/t/s	Parity Error. Indicates a data parity error is detected by a target during a write data phase or by an initiator during a read data phase.
SERR#	o/d	System Error. May be pulsed by any device to report address parity errors and critical errors other than parity.

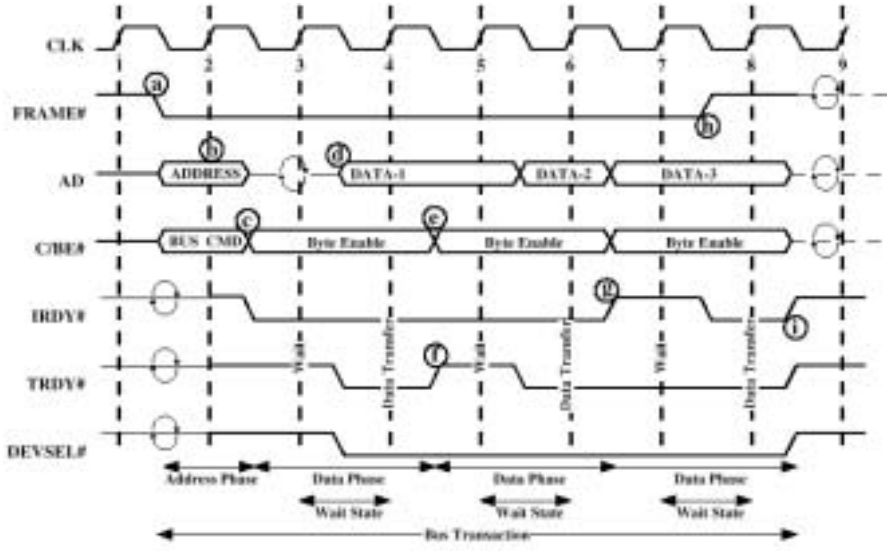
PCI Commands

- Transactions occur between an initiator, or master, and a target
- The initiator determines the type of transaction that is to occur next and signals the type on the C/BE# lines during the address phase
- Transaction types include:
  - Interrupt acknowledge
  - Special Cycle (for broadcasting to multiple targets)
  - I/O Read & Write
  - Memory Read [line/multiple], Write, & Write and Invalidate
  - Configuration Read & Write
  - Dual Address Cycle (indicates 64 bit addressing)

Interpretation of PCI Read Commands

Read Command Type	For Cachable Memory	For Noncachable Memory
Memory Read	Bursting one-half or less of a cache line	Bursting 2 data transfer cycles or less
Memory Read Line	Bursting more than one-half a cache line to three cache lines	Bursting 3 to 12 data transfers
Memory Read Multiple	Bursting more than three cache lines	Bursting more than 12 data transfers

PCI Read Timing Diagram



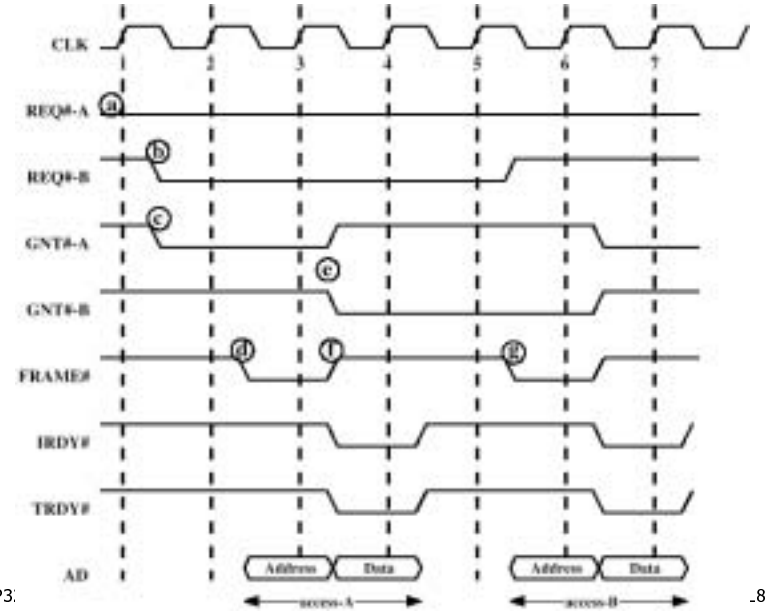
PCI Data Transfers

- Every data transfer is a single transaction consisting of one address phase and one or more data phases
- All events are synchronized to falling clock transitions in the middle of a clock cycle – bus devices sample the bus lines on the rising edge at the beginning of a bus cycle
- In the previous diagram, the significant events are:
  a. FRAME# is asserted once the master gains control of the bus and remains asserted until the initiator is ready to complete the last data phase. The initiator also places the start address onto the AD lines and the read command onto the C/BE# lines.
  b. The target device recognizes its address at the start of clock 2
  c. The initiator ceases driving the AD bus. A turnaround cycle (indicated by the circular arrows) is required on all signal lines that may be driven by more than one device. The initiator changes C/BE# info to designate which AD lines are to be used for the transfer and also asserts IRDY# to indicate that it is ready for the first data item

PCI Data Transfers

d. The selected target asserts DEVSEL# to indicate that it has recognized the address and will respond. It places the requested data onto AD and asserts TRDY# to indicate valid data is present
e. The initiator reads the data at clock 4 and changes the C/BE# lines as needed in preparation for the next read
f. In this example, the target needs time to prepare the 2nd block of data and therefore deasserts TRDY# – the initiator thus does not read the data lines during clock 5 and does not change the byte enable. The read takes place during clock 6
g. The initiator places the 3rd block of data onto the bus but the initiator is not ready this time. It therefore deasserts IRDY# thereby causing the target to maintain the 3rd item on the bus
h. The initiator knows the 3rd data transfer is the last and thus deasserts FRAME# to signal the target that this is the last data transfer and asserts IRDY# to complete the transfer
i. The initiator deasserts IRDY#, returning the bus to the idle state, and the target deasserts TRDY# and DEVSEL#

PCI Bus Arbitration



PCI Bus Arbitration

- PCI uses a centralized, synchronized arbitration scheme
- Each master has a unique request (REQ) and grant (GNT) signal
- The PCI spec does not dictate a particular arbitration policy, i.e., round-robin, FCFS, or some priority scheme could be used
- Arbitration between two devices A and B occurs as follows:
  a. A has asserted its REQ line before clock 1 when the arbiter samples the line
  b. During clock 1, B requests use of the bus by asserting its REQ signal
  c. The arbiter asserts GNT-A to grant bus access to A
  d. Bus master A samples GNT-A at start of clock 2. It asserts FRAME# and places address info onto AD and the command onto C/BE#. It continues to assert REQ since it has a second transaction to perform

PCI Bus Arbitartion

e. The bus arbiter samples all REQ lines at beginning of clock 3 and makes an arbitration decision to grant the bus to B for the next transaction. It then asserts GNT-B and deasserts GNT-A, however, B is not able to use the bus until it returns to an idle state (IRDY# & TRDY# deasserted)
f. A deasserts FRAME# to indicate the last (and only) data transfer is in progress, places data onto AD and signals the target with IRDY#. The target reads the data at the start of the next clock cycle.
g. At the beginning of clock 5, B finds IRDY# and FRAME# deasserted and so is able to take control of the bus by asserting FRAME#. It deasserts its REQ line since it only wishes to perform one transaction.

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

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