System Buses

Chapter 5

S. Dandamudi

Outline

- Introduction
- Bus design issues
  - ∗ Bus width
  - ∗ Bus type
  - ∗ Bus operations
- Synchronous bus
  - ∗ Bus operation
  - ∗ Wait states
  - ∗ Block transfer
- Asynchronous bus
- Bus arbitration
  - ∗ Dynamic bus arbitration
  - ∗ Implementation
    - » Centralized
    - » Distributed
- Example buses
  - ∗ ISA
  - ∗ PCI
  - ∗ AGP
  - ∗ PCI-X
  - ∗ PCMCIA

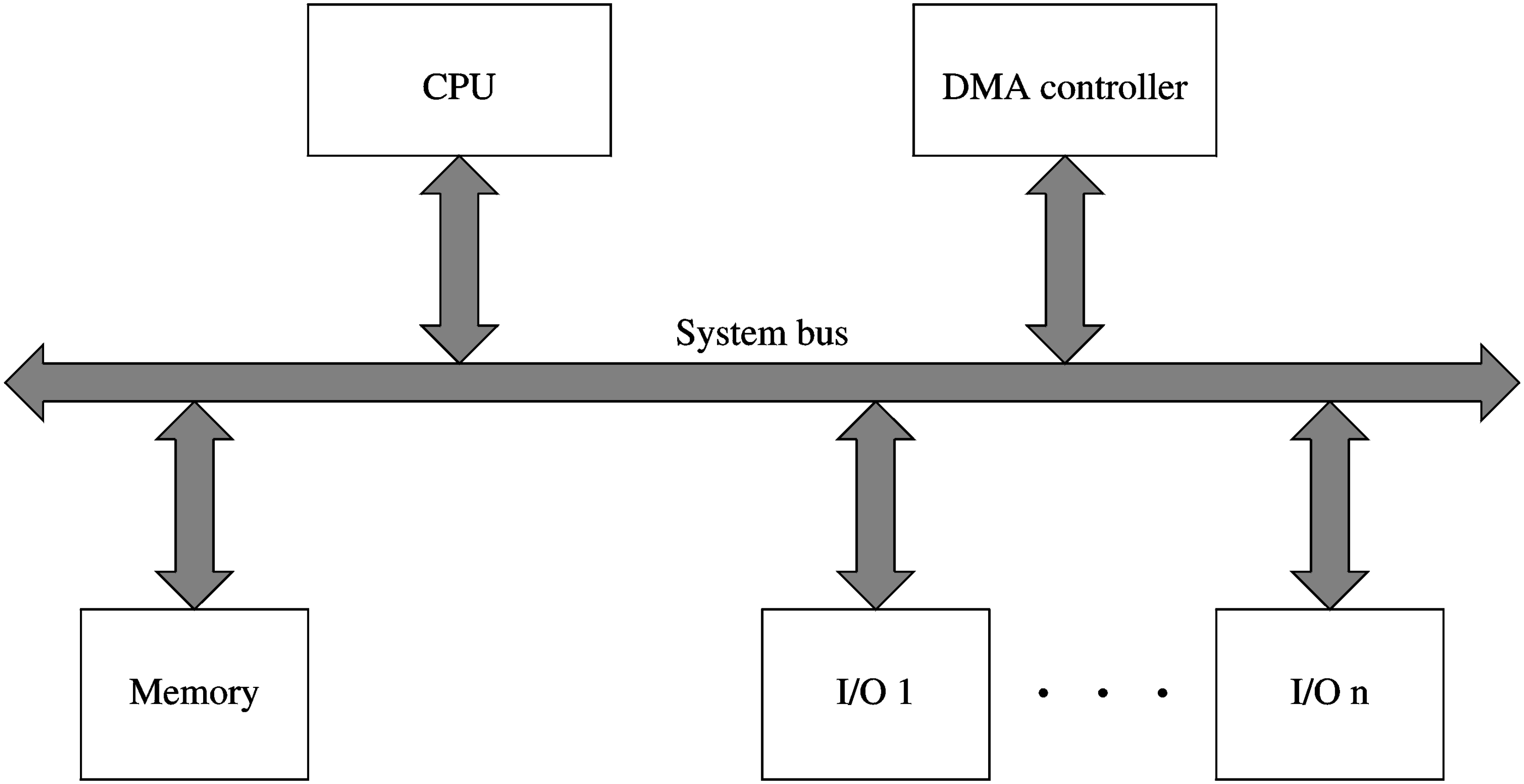
Introduction

- System buses
  - » Internal
    - – PCI
    - – AGP
    - – PCMCIA, …
    - – Focus of this chapter
  - » External
    - – USB
    - – FireWire, …
    - – Discussed in Chapter 19

Introduction (cont'd)

- Bus transactions
  - ∗ Sequence of actions to complete a well-defined activity
    - » Memory read, memory write, I/O read, burst read
      - – Master initiates the transaction
        - ✈A slave responds
- Bus operations
  - ∗ A bus transaction may perform one or more bus operations
- Bus cycle
  - ∗ Each operation may take several bus cycles
    - » Each is a bus clock cycle

Introduction (cont'd)



Introduction (cont'd)

- System bus consists of
  - ∗ Address bus
  - ∗ Data bus
  - ∗ Control bus
- Buses can be
  - ∗ Dedicated
  - ∗ Multiplexed
  - ∗ Synchronous
  - ∗ Asynchronous

Introduction (cont'd)

- Control bus
  - ∗ Memory read and Memory write
  - ∗ I/O read and I/O write
  - ∗ Ready
  - ∗ Bus request and Bus grant
  - ∗ Interrupt and Interrupt acknowledgement
  - ∗ DMA request and DMA acknowledgement
  - ∗ Clock
  - ∗ Reset

Bus Design Issues

- Need to consider several design issues
  - ∗ Bus width
    - » Data and address buses
  - ∗ Bus type
    - » Dedicated or multiplexed
  - ∗ Bus operations
    - » Read, write, block transfer, interrupt, …
  - ∗ Bus arbitration
    - » Centralized or distributed
  - ∗ Bus timing
    - » Synchronous or asynchronous

Bus Width

- Data bus width
  - ∗ A critical parameter in determining system performance
  - ∗ Need not correspond to the ISA-specific value
    - » Pentium is a 32-bit processor
      - – But has 64-bit data bus
    - » Itanium is a 64-bit processor
      - – But has 128-bit data bus
  - ∗ The wider the data bus, the better
    - » Wider buses are expensive

Bus Width (cont'd)

- Address bus width
  - ∗ Determines the system addressing capacity
  - ∗ *N* address lines directly address 2^N memory locations
    - » 8086: 20 address lines
      - – Could address 1 MB of memory
    - » Pentium: 32 address lines
      - – Could address 4 GB of memory
    - » Itanium: 64 address lines
      - – Could address 2^{64} bytes of memory

Bus Type

- Dedicated buses
  - ∗ Separate buses dedicated to carry data and address information
  - ∗ Good for performance
    - » But increases cost
- Multiplexed buses
  - ∗ Data and address information is time multiplexed on a shared bus
  - ∗ Better utilization of buses
  - ∗ Reduces cost

Bus Operations

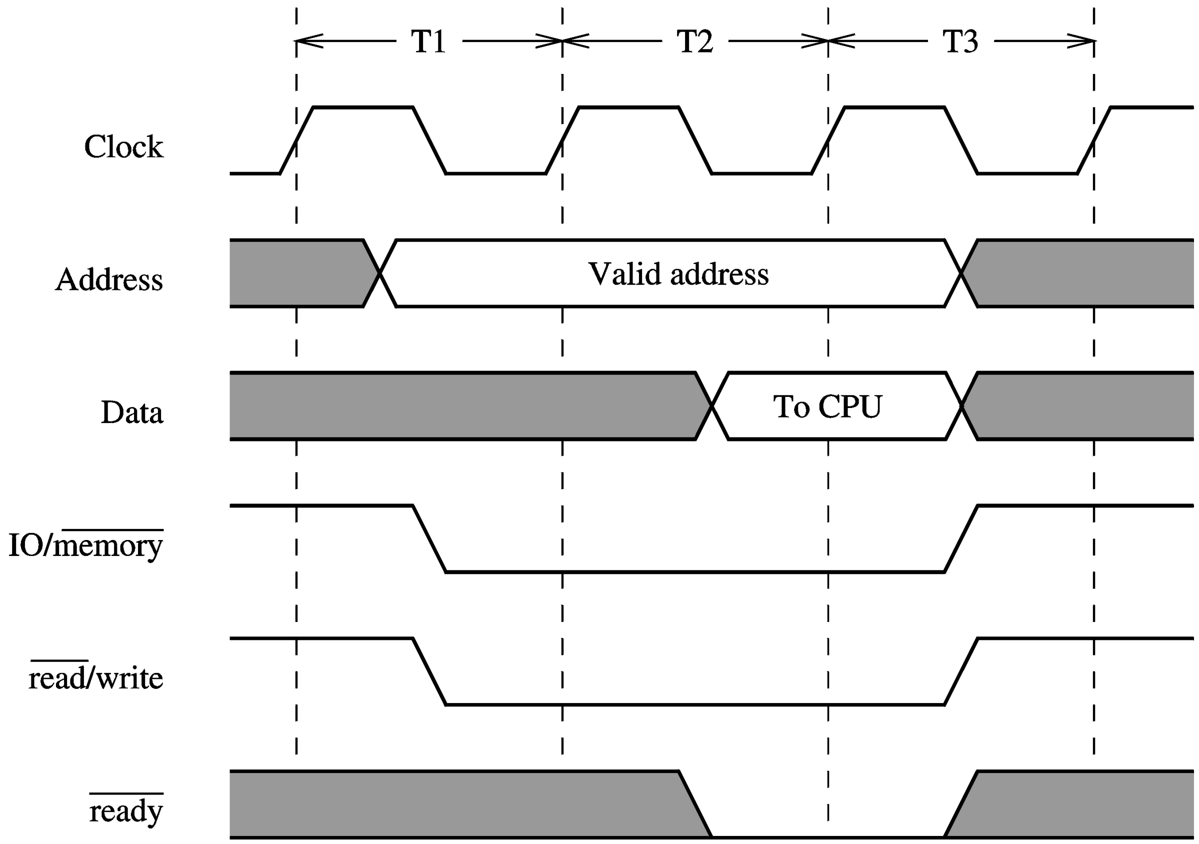
- Basic operations
  - ∗ Read and write
- Block transfer operations
  - ∗ Read or write several contiguous memory locations
    - » Example: cache line fill
- Read-modify-write operation
  - ∗ Useful for critical sections
- Interrupt operation
- Several other types…

Synchronous Bus

- A bus clock signal provides timing information for all actions
  - ∗ Changes occur relative to the falling or rising edge of the clock
  - ∗ Choosing appropriate clock is important
  - ∗ Easier to implement
  - ∗ Most buses are synchronous
- Bus operations can be
  - ∗ With no wait states, or
  - ∗ With wait states

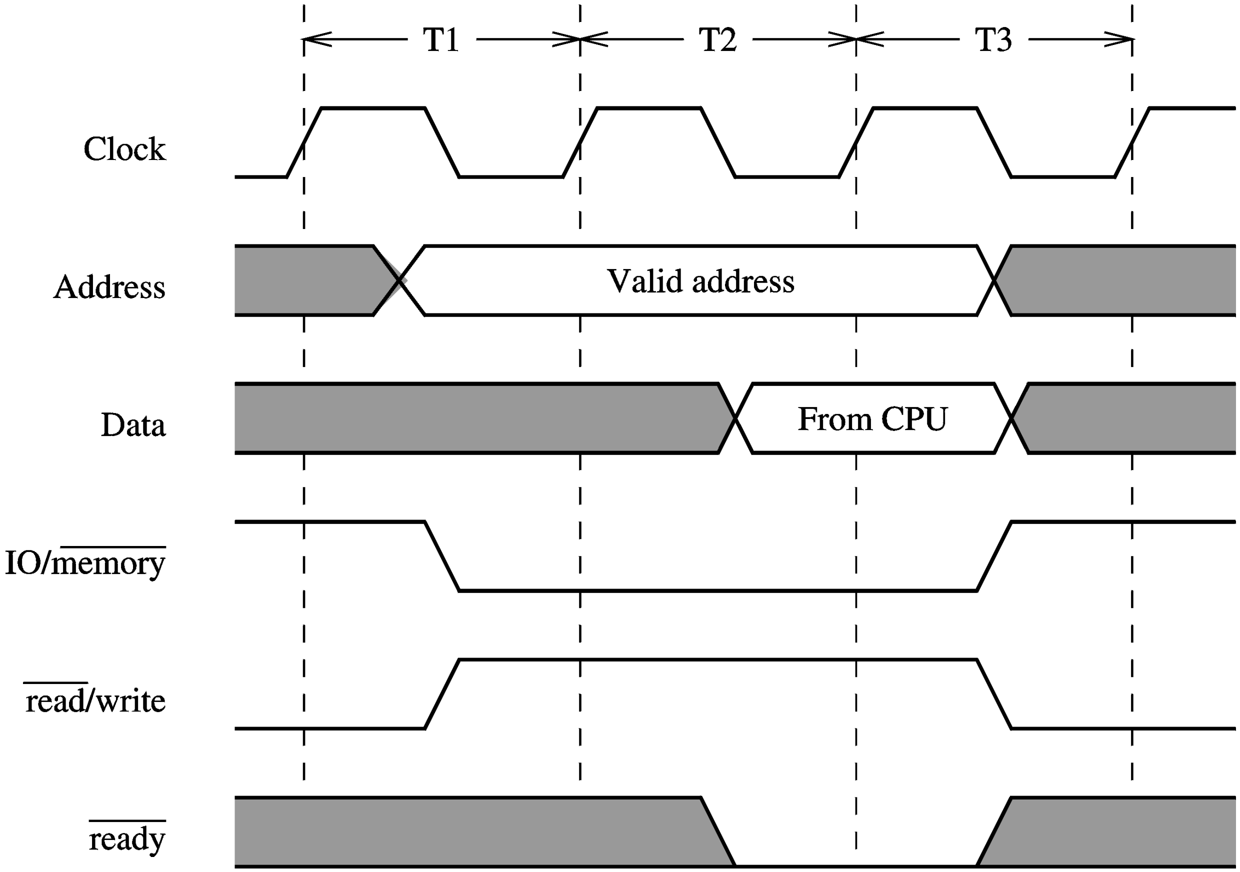
Synchronous Bus (cont’d)

- Memory read operation with no wait states


T1
T2
T3
Clock
Address
Valid address
Data
To CPU
IO/$\overline{\text{memory}}$
$\overline{\text{read}}$/write
$\overline{\text{ready}}$

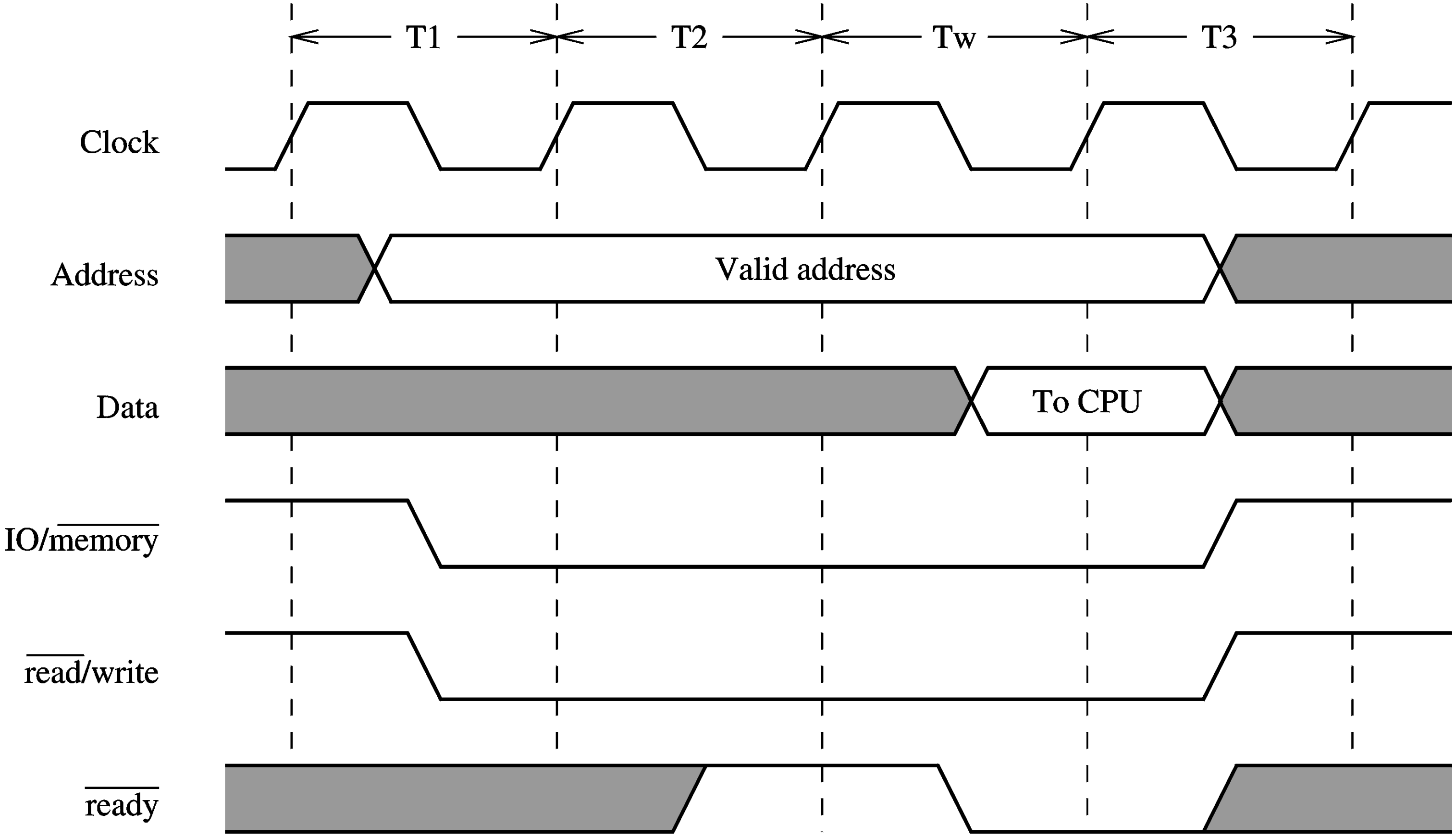
Synchronous Bus (cont'd)

- Memory write operation with no wait states


T1
T2
T3
Clock
Address
Valid address
Data
From CPU
IO/$\overline{\text{memory}}$
$\overline{\text{read}}$/write
$\overline{\text{ready}}$

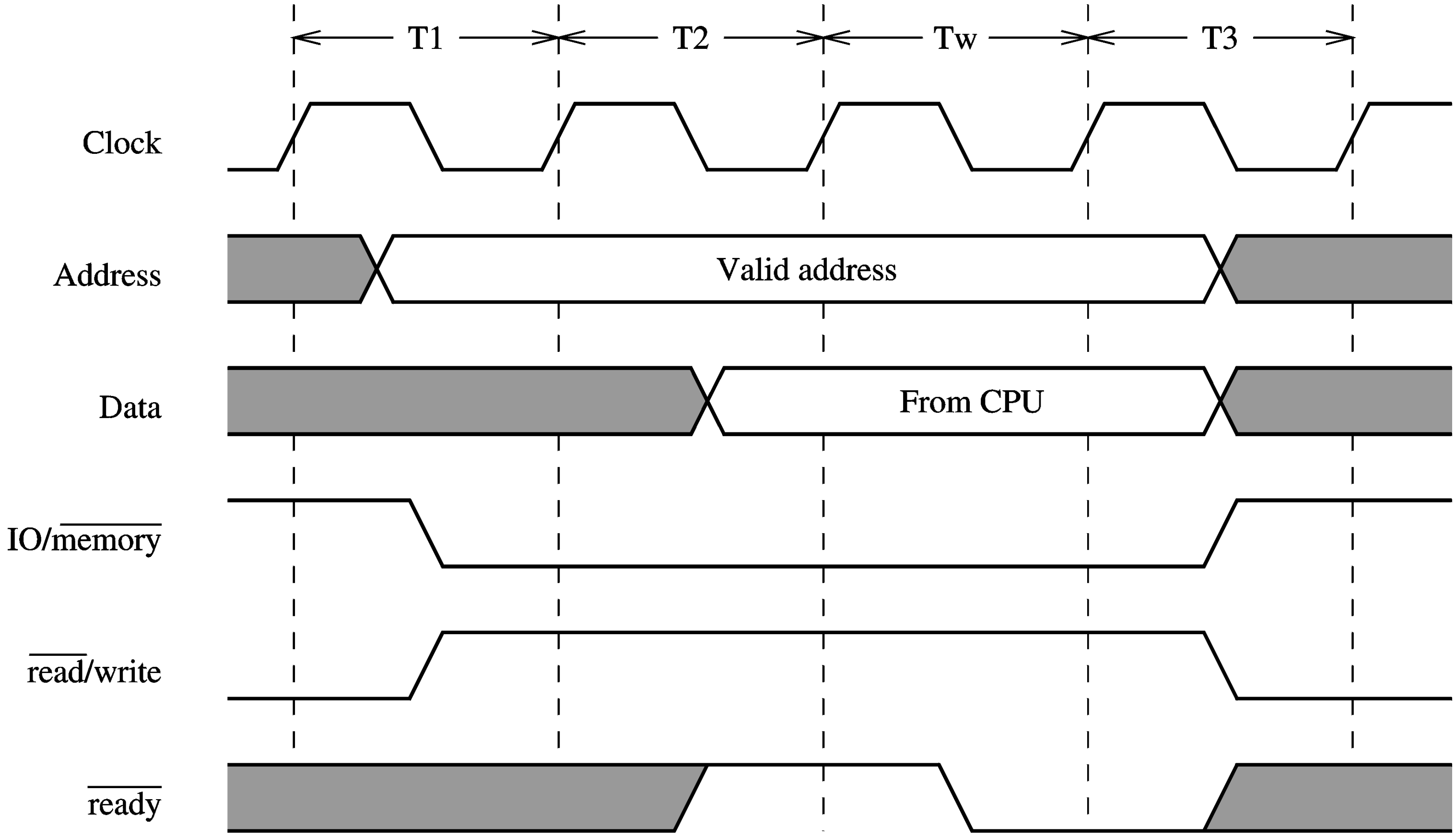
Synchronous Bus (cont'd)

- Memory read operation with a wait state


T1
T2
Tw
T3
Clock
Address
Valid address
Data
To CPU
IO/$\overline{\text{memory}}$
$\overline{\text{read}}$/write
$\overline{\text{ready}}$

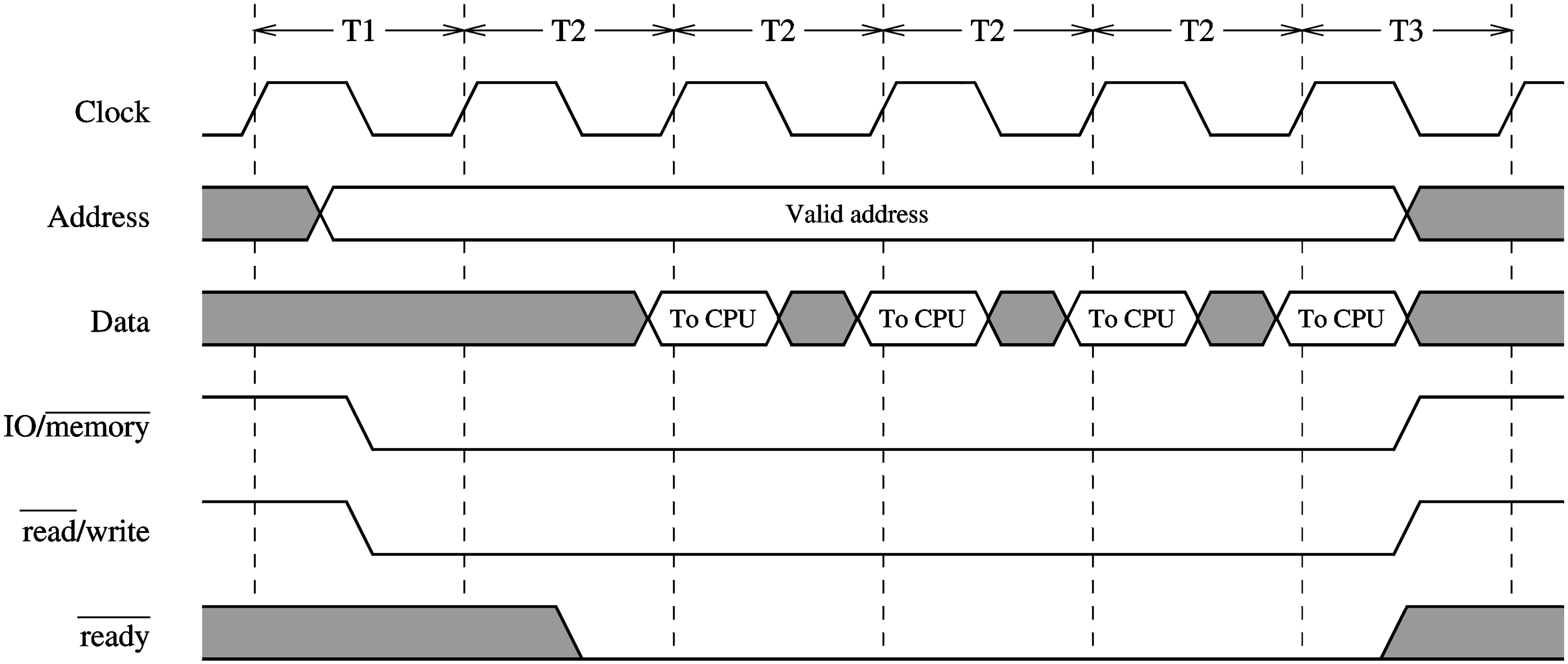
Synchronous Bus (cont'd)

- Memory write operation with a wait state


T1
T2
Tw
T3
Clock
Address
Valid address
Data
From CPU
IO/$\overline{\text{memory}}$
$\overline{\text{read}}$/write
$\overline{\text{ready}}$

Synchronous Bus (cont’d)

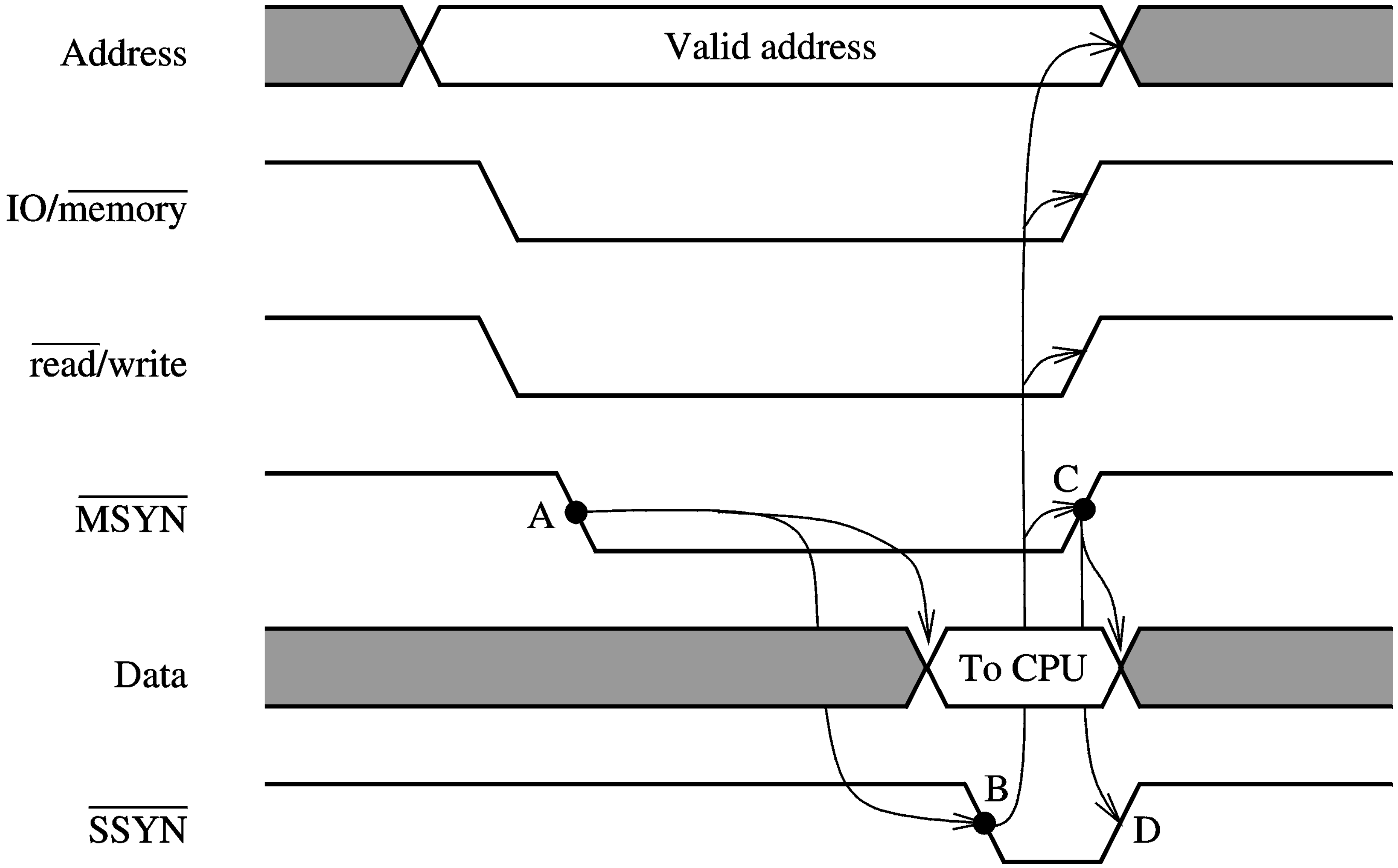
- Block transfer of data


T1
T2
T2
T2
T2
T3
Clock
Address
Valid address
Data
To CPU
To CPU
To CPU
To CPU
IO/$\overline{\text{memory}}$
$\overline{\text{read}}$/write
$\overline{\text{ready}}$

Asynchronous Bus

- No clock signal to synchronize actions
- Operates in master-slave mode
- Uses handshaking to perform a bus transaction
  - ∗ Two synchronization signals facilitate this
    - » Master synchronization (MSYN)
    - » Slave synchronization (SSYN)
- Advantage of asynchronous buses
    - » No need for bus clock
- Synchronous buses
    - » Easier to implement

Asynchronous Bus (cont'd)



Bus Arbitration

- More than one bus master can request the bus
  - ∗ Need an arbitration mechanism to allocate the bus
- Bus arbitration can be done either
  - ∗ Statically
  - ∗ Dynamically
- Static arbitration
  - ∗ Done in a predetermined way
    - » Easy to implement
    - » Does not take needs into account
    - » Poor utilization
      - – Bus could be assigned even when not needed

Dynamic Bus Arbitration

- Bus allocated only in response to a request
- Each master is equipped with
  - Bus request line
  - Bus grant line

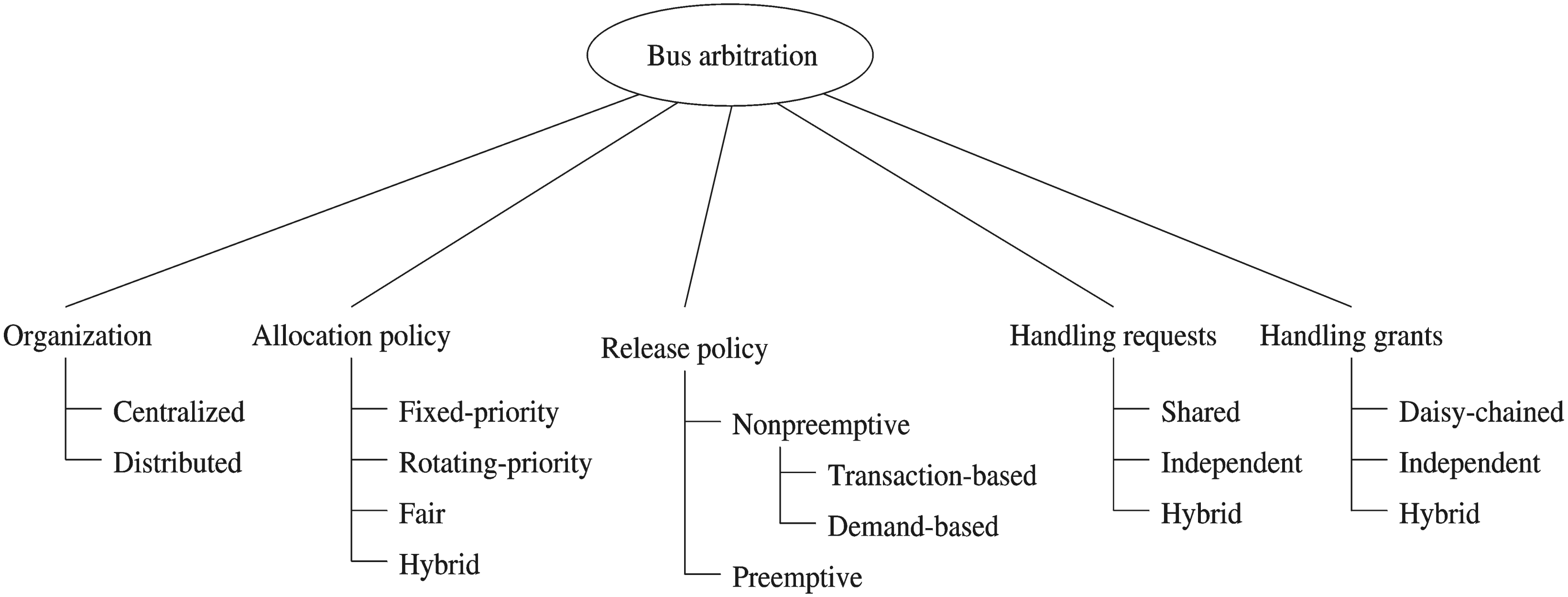
- A master uses the bus request line to let others know that it needs the bus
- Before a master can use the bus, it must receive permission to use the bus via the bus grant line

Dynamic Bus Arbitration (cont'd)

- Bus arbitration can be implemented
  - ∗ Centralized
  - ∗ Distributed
- Centralized arbitration
    - » A central arbiter receives all bus requests
    - » Uses an allocation policy to determine which request should be granted
    - » This decision is conveyed through the bus grant lines
  - ∗ Once the transaction is over, bus is released
    - » A bus release policy determines the actual release mechanism

Dynamic Bus Arbitration (cont'd)

- Distributed arbitration
  - ∗ Arbitration hardware is distributed among the masters
  - ∗ A distributed arbitration algorithm is used to determine who should get the bus



Dynamic Bus Arbitration (cont'd)

- Bus Allocation Policies
  - Fixed priority
    - Each master is assigned a fixed priority
    - Highest priority master always gets the bus
      - Could hog the bus
    - Priorities can be assigned based on the importance of service
  - Rotating priority
    - Priority is not fixed
    - Several ways of changing priority
      - Increase the priority as a function of waiting time
      - Lowest priority for the master that just received the bus

Dynamic Bus Arbitration (cont'd)

- Bus Allocation Policies (cont'd)
  - ∗ Fair policies
    - » A fair policy will not allow starvation
      - – Rotating priority policies are fair
    - » Fair policies need not use priorities
    - » Fairness can be defined in several ways
      - – A window-based request satisfaction
      - – Within a specified time period
        - ✈In PCI, we can specify the maximum delay to grant a request
  - ∗ Hybrid policies
    - » Both priority and fairness can be incorporated into a single policy

Dynamic Bus Arbitration (cont'd)

- Bus Release Policies
  - ∗ Governs the conditions under which the current master releases the bus
  - ∗ Two types
    - » Non-preemptive
      - – Current master voluntarily releases the bus
      - – Disadvantage
        - ✈May hold bus for long time
    - » Preemptive
      - – Forces the current master to release the bus without completing its bus transaction

Dynamic Bus Arbitration (cont’d)

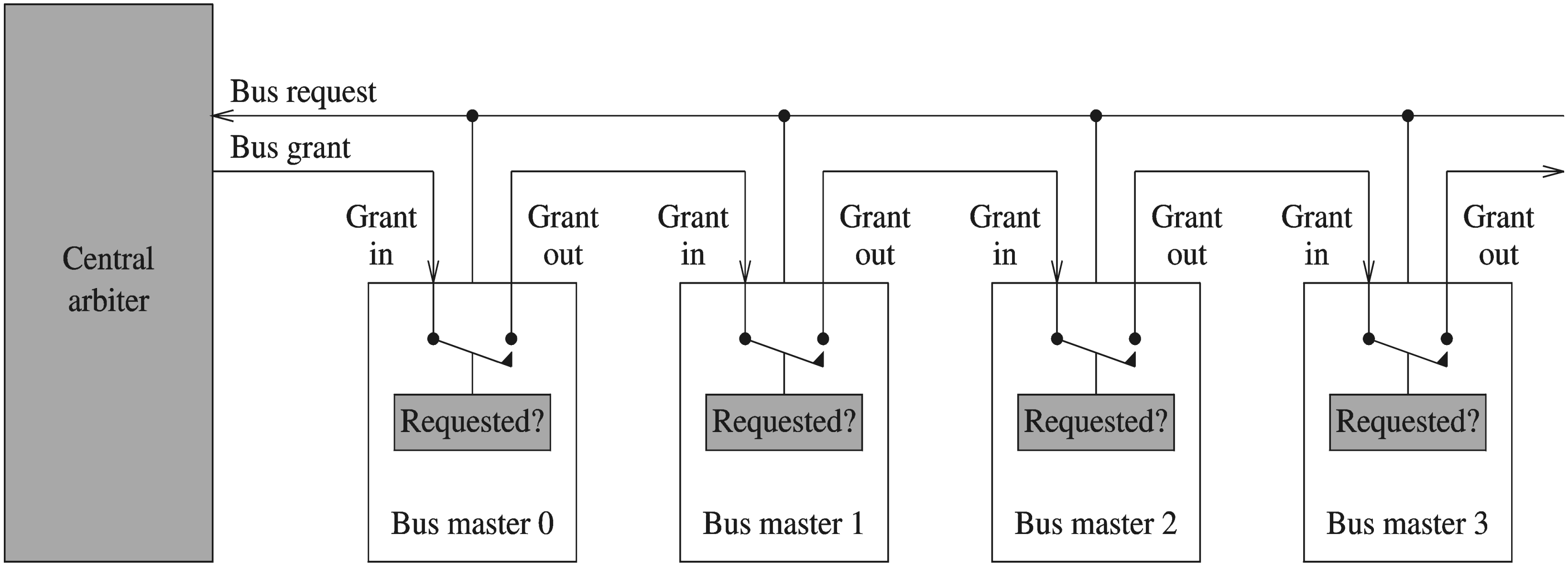
- Non-Preemptive Bus Release Policies
  - ∗ Transaction-based release
    - » Releases bus after completing the current transaction
    - » Requests bus again if it has more transactions
      - – By releasing the bus after every transactions, fairness can be ensured
    - » Easy to implement
    - » Unnecessary overhead if only one master needs the bus
  - ∗ Demand-driven release
    - » Avoids unnecessary bus requests of the previous policy
    - » Releases the bus only if another master requests the bus
    - » More efficient

Dynamic Bus Arbitration (cont'd)

- Implementation
  - ∗ Centralized bus arbitration
    - » Daisy-chaining
      - – Uses a single, shared bus request signal
      - – Central arbiter sends the grant signal to the first master in the chain
        - ✈Each master passes the grant signal to its neighbor if it does need the bus
        - ✈Grabs the grant signal if it wants the bus
      - – Easy to implement
        - ✈Needs three control lines independent of the number of hosts

Dynamic Bus Arbitration (cont'd)

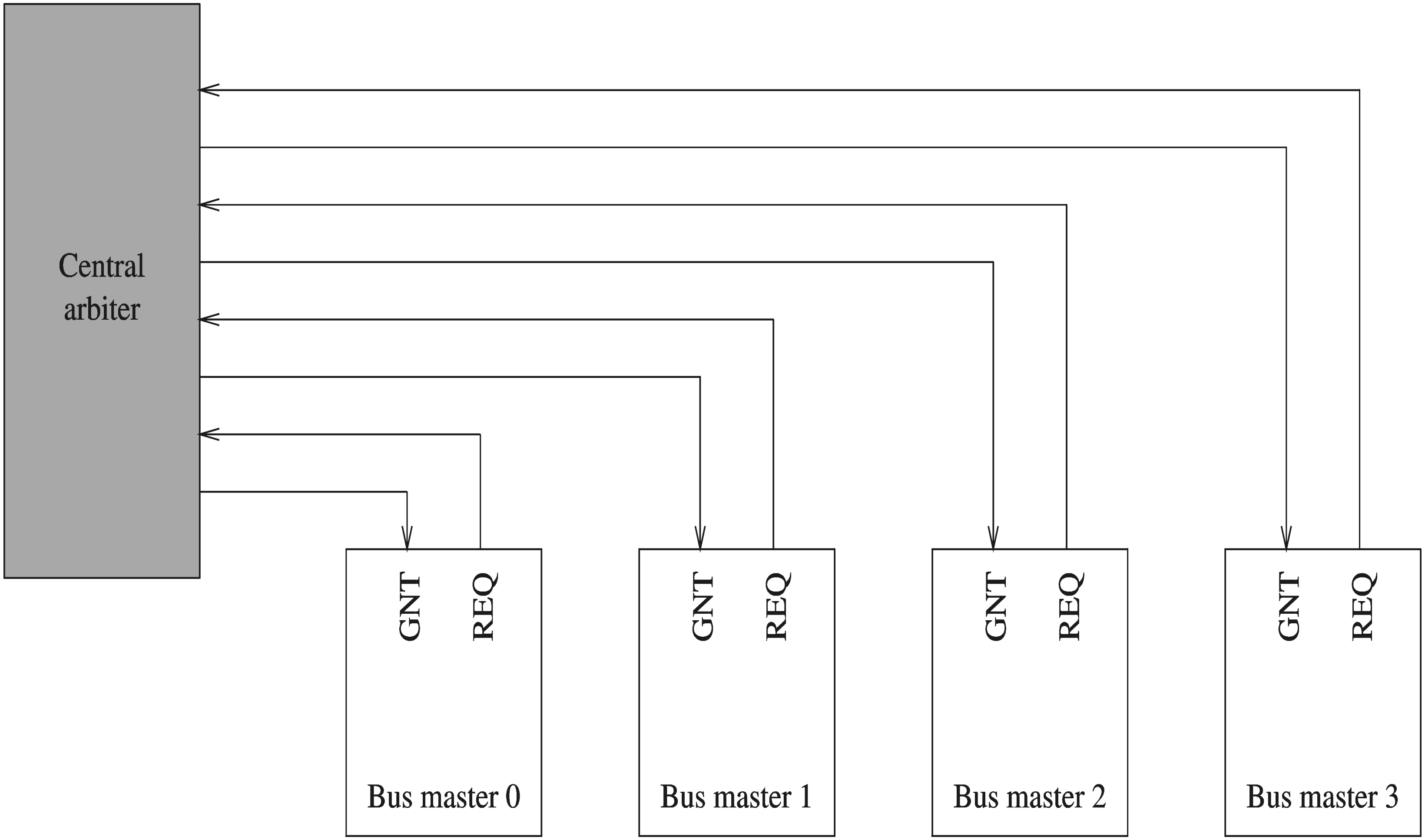
- Three problems
  - ∗ Implements fixed priority
  - ∗ Arbitration time proportional to number of hosts
  - ∗ Not fault-tolerant



Dynamic Bus Arbitration (cont'd)

- Independent requests
  - ∗ Arbiter is connected to each master
  - ∗ Variety of bus allocation policies can be implemented
  - ∗ Avoids the pitfalls of daisy-chaining
    - » Fault-tolerant
    - » Short, constant arbitration time
  - ∗ Disadvantages
    - » Complex to implement
    - » Number of control signals is proportional to the number of hosts
  - ∗ PCI uses this implementation technique

Dynamic Bus Arbitration (cont’d)

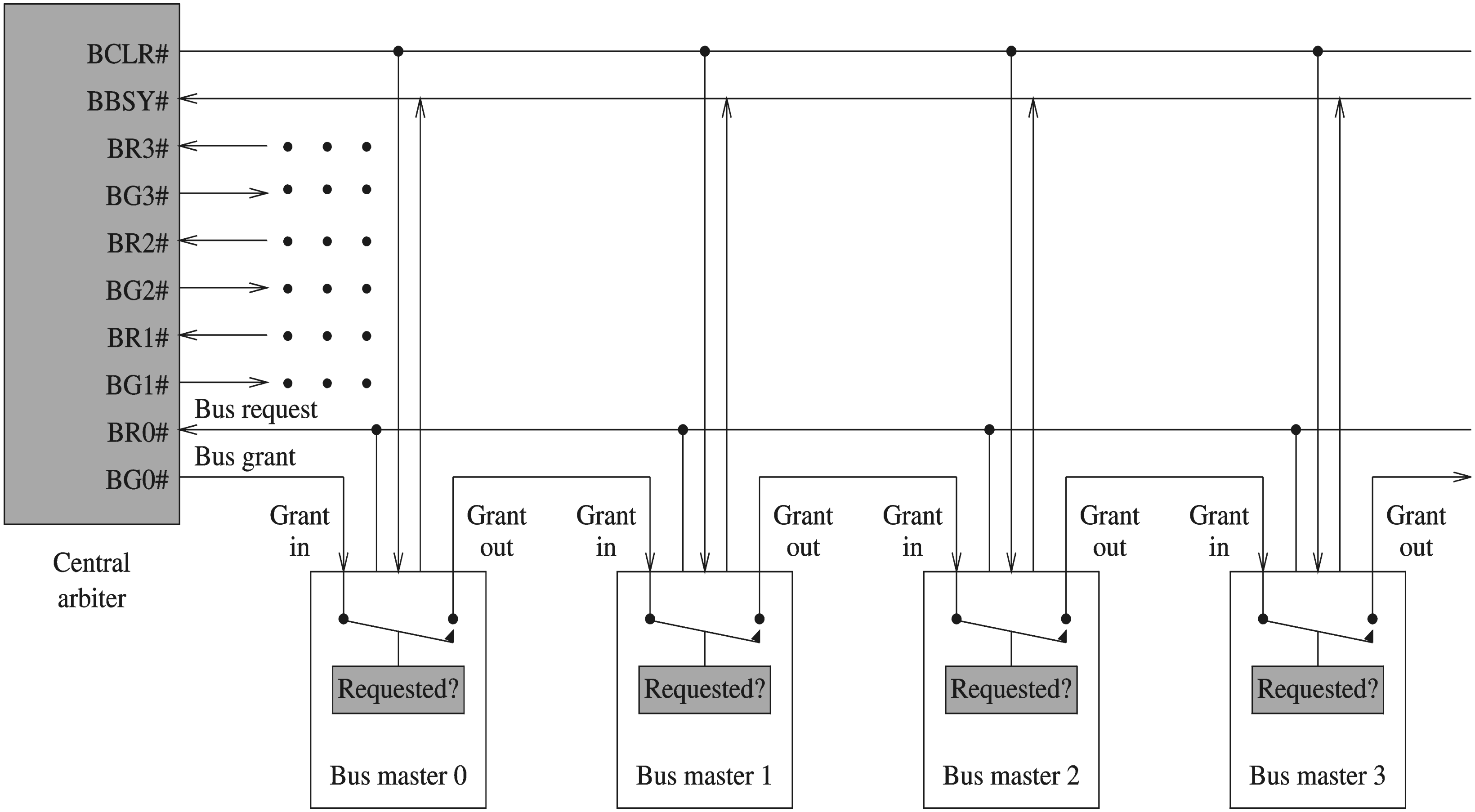


Dynamic Bus Arbitration (cont'd)

- Hybrid scheme
  - ∗ Daisy-chaining and independent requests represent two extremes
    - » Daisy-chaining
      - – Cheaper but several problems
    - » Independent requests
      - – Expensive but avoids the problems of daisy-chaining
  - ∗ Hybrid scheme combines good features of these two
    - » Bus masters are divided into N classes
    - » Independent strategy at the class-level
    - » Daisy-chaining within each class
  - ∗ VME bus uses a hybrid policy

Dynamic Bus Arbitration (cont'd)

VME bus arbitration



Dynamic Bus Arbitration (cont'd)

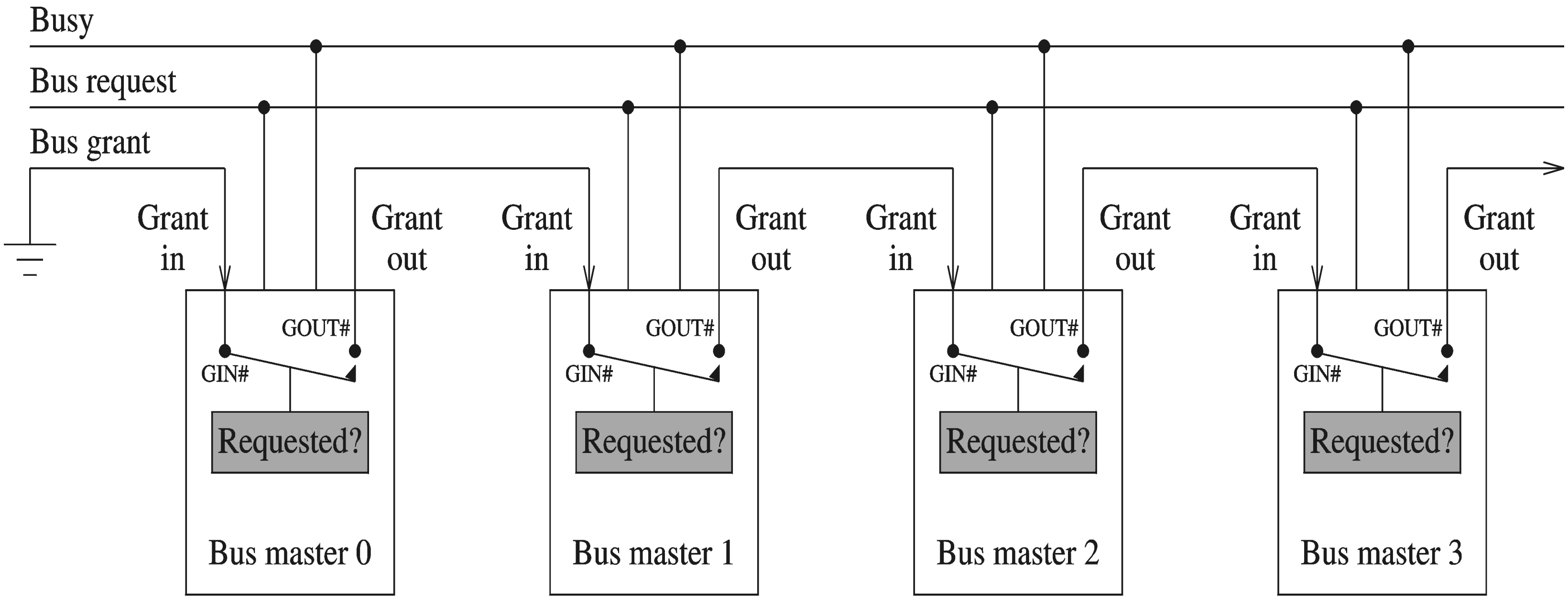
- VME bus arbitration
  - ∗ Allocation policies supported
    - » Fixed-priority
      - – GR0: Lowest priority
      - – GR3: Highest priority
    - » Rotating-priority
      - – Round-robin based policy
        - ✈Assigns the lowest priority to the bus that just received the bus
    - » Daisy-chaining
      - – Implemented by connecting all masters to BR3 grant request line

Dynamic Bus Arbitration (cont'd)

- VME bus arbitration (cont'd)
  - ∗ Release policies supported
    - » Default release policy
      - – Transaction-based
      - – Non-preemptive
    - » When fixed priority is used
      - – Preemptive release is possible when a higher priority request comes in
      - – To effect preemption
        - ✈Bus clear (BCLR) line is asserted
        - ✈The current bus master releases the bus when BCLR is low

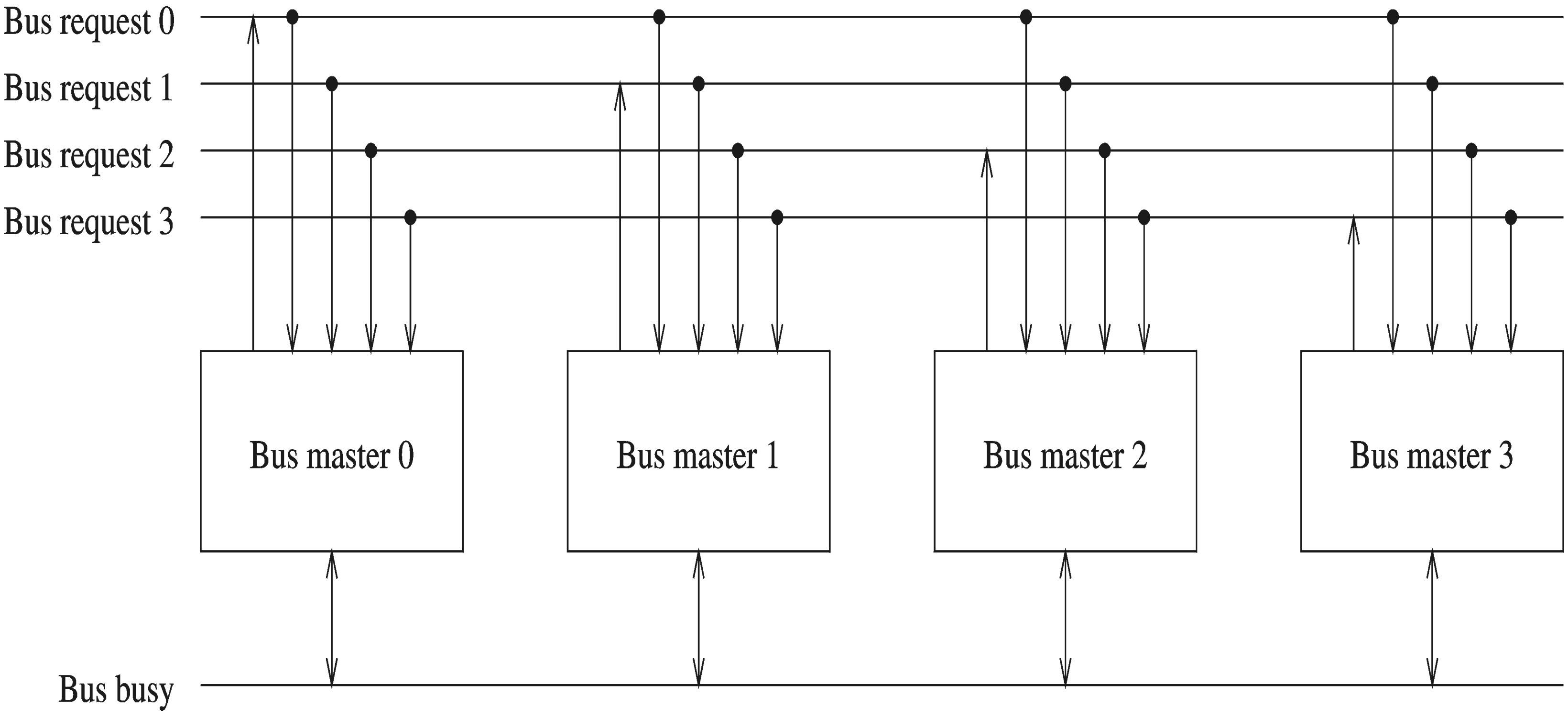
Dynamic Bus Arbitration (cont'd)

- Distributed bus arbitration
  - ∗ Daisy-chaining
    - » Needs three control lines as before



Dynamic Bus Arbitration (cont'd)

- Separate bus request lines
  - All bus masters can read all bus request lines
  - Highest priority master gets the bus



Dynamic Bus Arbitration (cont’d)

- Separate bus request lines (cont’d)
  - Implements fixed-priority scheme
  - Starvation is a potential problem
  - To avoid starvation, a kind of honor system can be used
    - The highest-priority master that just used the bus will not raise its bus request until all the other lower priority masters have been allocated the bus

Example Buses

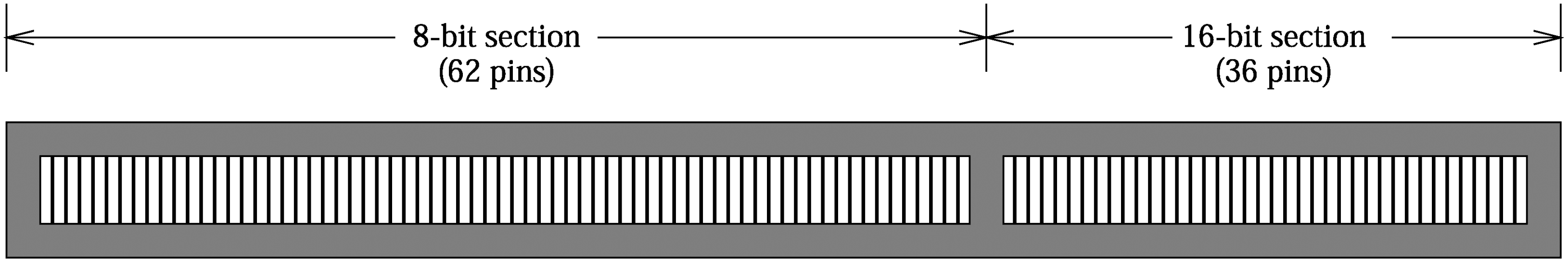
- We look at five buses
  - ∗ ISA
    - » Supports older text-based applications
  - ∗ PCI
    - » Supports modern window-based systems
  - ∗ AGP
    - » Supports high-performance graphics and full-motion video
  - ∗ PCI-X
    - » Improved and faster PCI
  - ∗ PCMCIA
    - » Useful for laptops

ISA Bus

- Closely associated with the PC system bus
  - ∗ ISA = Industry Standard Architecture
- First ISA bus (8-bit wide data path)
  - ∗ Based on 8088 processor
    - » It had 82 pins including
      - 20 address lines
      - 8 data lines
      - 6 interrupt signals
      - Memory read, memory write
      - I/O read, and I/O write
      - 4 DMA requests and 4 DMA acks

ISA Bus (cont'd)

- 16-bit ISA
  - ∗ Added 36 pins to the 8-bit ISA bus
  - ∗ 24 address lines
  - ∗ 16 data lines
  - ∗ Backward compatible with 8-bit ISA



ISA Bus (cont'd)

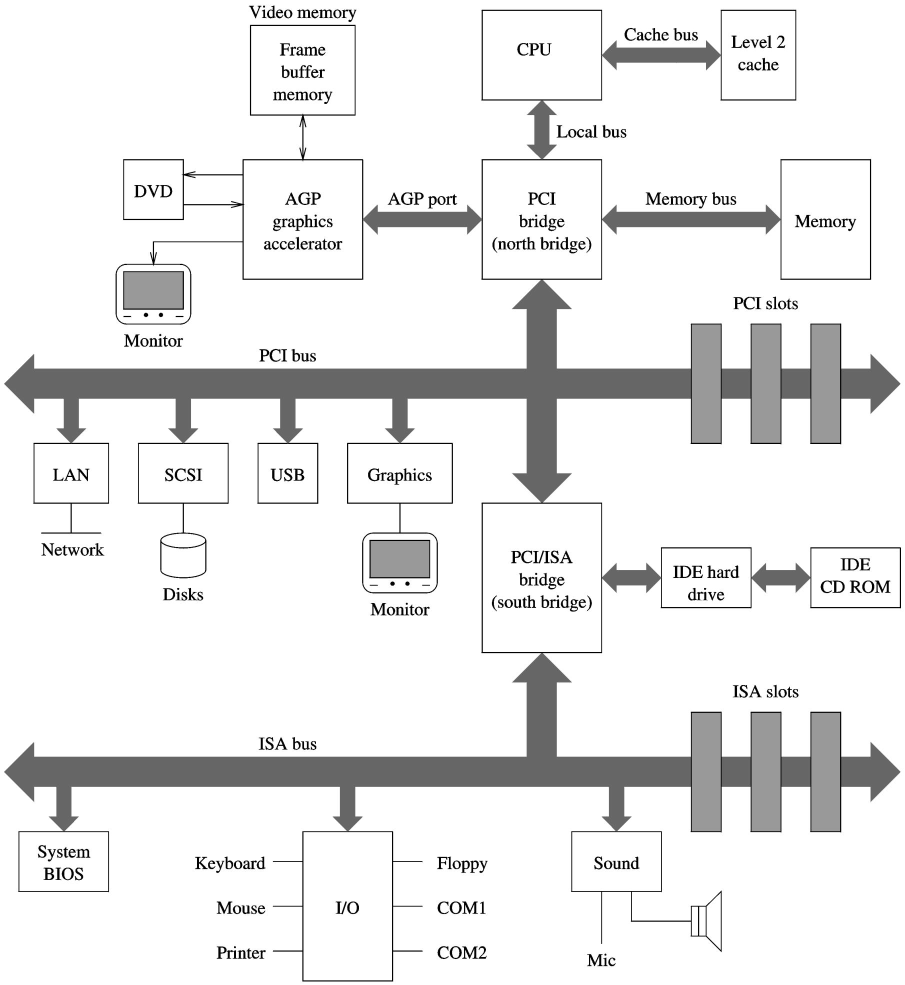
- Operates at 8.33 MHz
- Bandwidth of about 8 MB/s
- 32-bit processors need more support
  - * Several attempts were made to accommodate
    - » EISA (Extended ISA)
      - – Bus mastering signals
    - » MCA (Micro Channel Architecture)
      - – IBM proprietary
      - – Never really took off
- ISA is used for older, slower I/O devices

PC System Buses

ISA

PCI

AGP



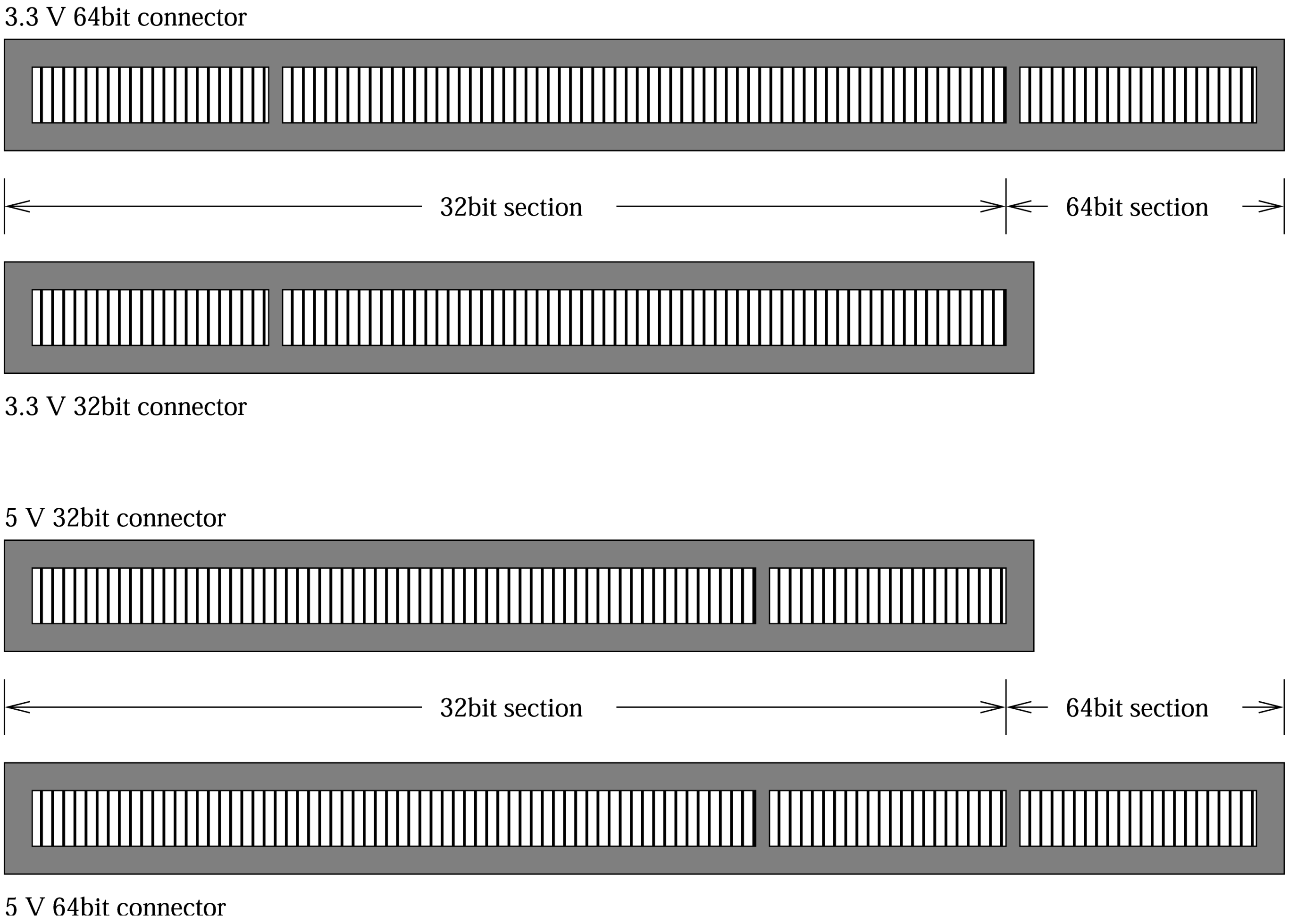
PCI Bus

- Work began in 1990
  - ∗ Intel began work on a new bus for their Pentium systems
    - » Processor independent
  - ∗ To support higher bandwidth requirements of window-based systems
    - » Original version (1.0) developed by Intel in 1990
    - » Version 2 in 1993
    - » Version 2.1 in 1995
    - » Version 2.2 introduced power management for mobile systems

PCI Bus (cont'd)

- Implemented either
  - ∗ 32-bit or 64-bit
- Operate at
  - ∗ 33 MHz or 66 MHz
  - ∗ 5 V (older cards) or 3.3 V (newer ones)
- 32-bit PCI operating at 33 MHz
  - ∗ Provides peak bandwidth of 133 MB/s
- 64-bit PCI operating at 66 MHz
  - ∗ Provides peak bandwidth of 528 MB/s

PCI Bus (cont’d)



PCI Bus (cont'd)

- Bus signals
  - ∗ Mandatory signals
    - » System signals
    - » Transaction control signals
    - » Bus arbitration and error reporting signals
  - ∗ Optional signals
    - » 64 bit extension signals and Interrupt request signals
- System signals
  - ∗ Clock (CLK)
  - ∗ Reset (RST#)
  - ∗ Address/Data bus (AD[0-31])
    - » 32 lines multiplexed to serve both address and data buses

PCI Bus (cont'd)

- ∗ Command Bus (C/BE#[0-3])
  - » Multiplexed command and byte enable (BE#) signals
    - – Command signals identify transaction type
      - ✈Memory read, memory write, …
      - ✈Asserted during address phase
    - – BE signals select the bytes to be transferred
      - ✈Asserted during data phase
      - ✈BE0# identifies byte 0, BE#1 byte 1, …
      - ✈0000 = all four bytes
      - ✈1111 = none of the bytes (null data phase)
- ∗ Parity Signal (PAR)
  - » Even parity for AD and C/BE lines

PCI Bus (cont'd)

- Transaction Control Signals
  - ∗ Cycle Frame (FRAME#)
    - » Indicates start of a bus transaction
    - » Also indicates the length of the bus transaction cycle
  - ∗ Initiator Ready (IRDY#)
    - » During Write transaction:
      - – Indicates the initiator has placed data on AD lines
    - » During Read transaction:
      - – Indicates the initiator is ready to accept data
  - ∗ Target Ready (TRDY#)
    - » Complements IRDY# signal
    - » IRDY# and TRDY# together implement handshake to transfer data

PCI Bus (cont’d)

- Transaction Control Signals
  - ∗ Stop Transaction (STOP#)
    - » Target asserts this to tell initiator that it wants to terminate the current transaction
  - ∗ Initialization Device Select (IDSEL)
    - » Used as a chip select for configuration read and write transactions
  - ∗ Device Select (DEVSEL#)
    - » Selected target asserts this to tell the initiator that it is present
  - ∗ Bus Lock (LOCK#)
    - » Imitator uses this to lock the target to execute atomic transactions

PCI Bus (cont'd)

- Bus Arbitration Signals
  - ∗ Uses centralized bus arbitration with independent request and grant lines
    - » Bus Request (REQ#)
      - – A device asserts when it needs the bus
    - » Bus Grant (GNT#)
      - – Bus arbiter asserts this signal to indicate allocation of the bus
  - ∗ Bus arbitration can overlap execution of another transaction
    - » Improves PCI performance

PCI Bus (cont'd)

- Error Reporting Signals
  - ∗ Parity Error (PERR#)
    - » All devices are expected to report this error
    - » Exceptions exist
      - – E.g. when transmitting video frames
  - ∗ System Error (SERR#)
    - » Any PCI device can generate this signal
      - – To indicate address and other errors
    - » Typically connected to NMI (non-maskable interrupt)

PCI Bus (cont'd)

- 64-bit Extension Signals
  - ∗ Address/Data Lines (AD[32 - 63])
    - » Extension to 64 bits
  - ∗ Command bus (C/BE#[4 - 7])
    - » Extended by 4 lines to handle 8 bytes
  - ∗ Request 64-Bit Transfer (REQ64#)
    - » Indicates to target that initiator likes 64-bit transfers
  - ∗ Acknowledge 64-Bit Transfer (ACK64#)
    - » Target indicates that it is capable of 64-bit transfers
  - ∗ Parity Bit for Upper Data (PAR64)
    - » Even parity for upper 32 AD bits and four command lines

PCI Bus (cont'd)

- Interrupt Request Lines
  - ∗ Four interrupt lines
    - » INTA#, INTB#, INTC#, INTD#
    - » Not shared
- Additional signals
  - ∗ To support snoopy cache protocol
  - ∗ IEEE 1149.1 boundary scan signals
    - » Allows in-circuit testing of PCI devices
  - ∗ M66En signal to indicate bus frequency
    - » Low: 33 MHz
    - » High: 66 MHz

PCI Commands

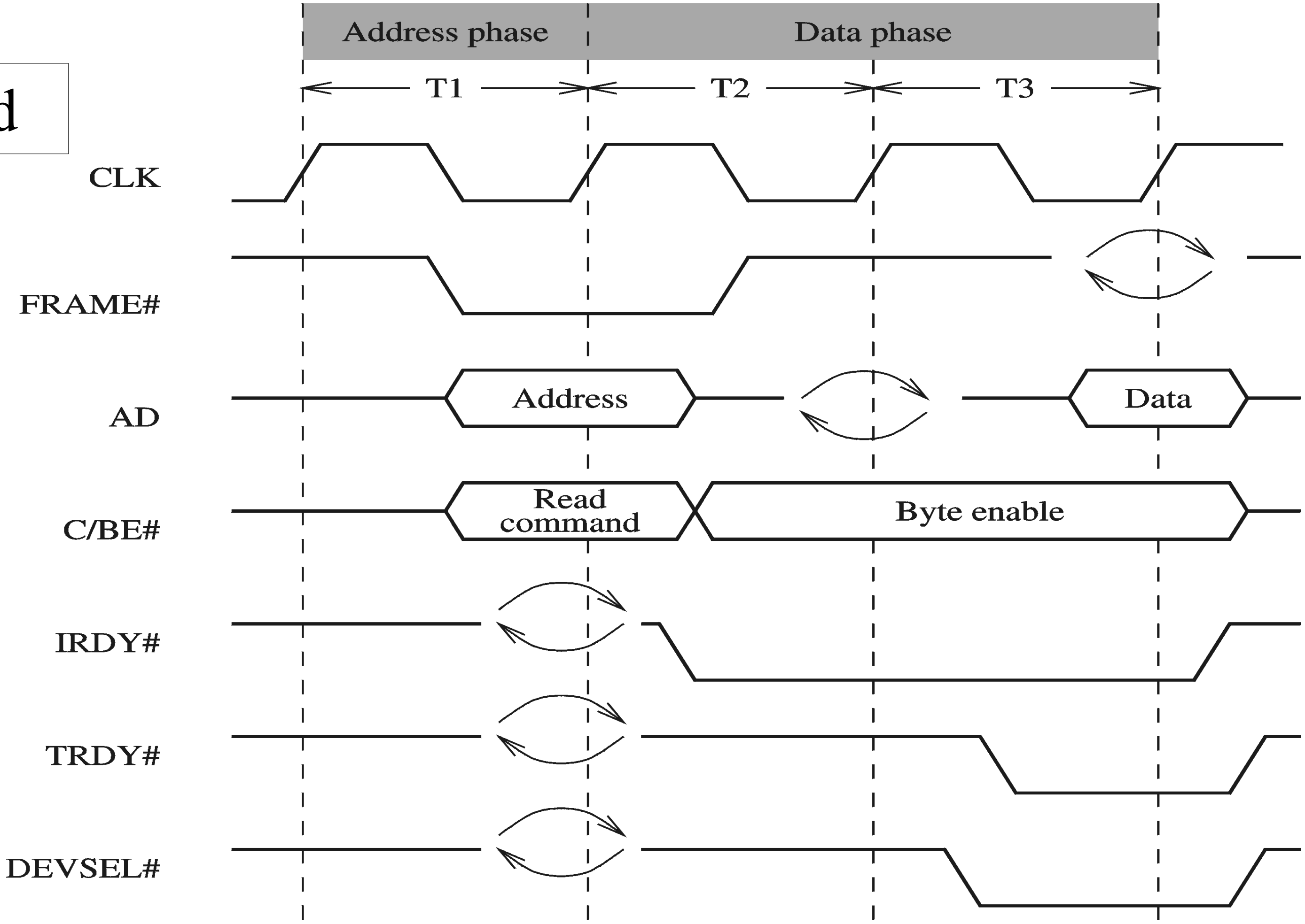
- A command value is placed on C/BE lines during the address phase
  - I/O operations
    - I/O Read and I/O Write
  - Memory operations
    - Standard memory operations
      - Memory Read and Memory Write
    - Bulk memory operations
      - Memory Read Line
      - Memory Read Multiple
      - Memory Write-and-Invalidate

PCI Commands (cont'd)

- Configuration operations
  - ∗ Every PCI device has a 256-byte configuration space
  - ∗ Two commands:
    - » Configuration Read and Configuration Write
- Miscellaneous operations
  - ∗ Special Cycle Command
    - » Used to broadcast a message to all PCI targets
      - – Shutdown and Halt
  - ∗ Dual Address Cycle Command
    - » Allows 32-bit initiator to use 64-bit addresses
      - – 64-bit address is passed as two 32-bit values

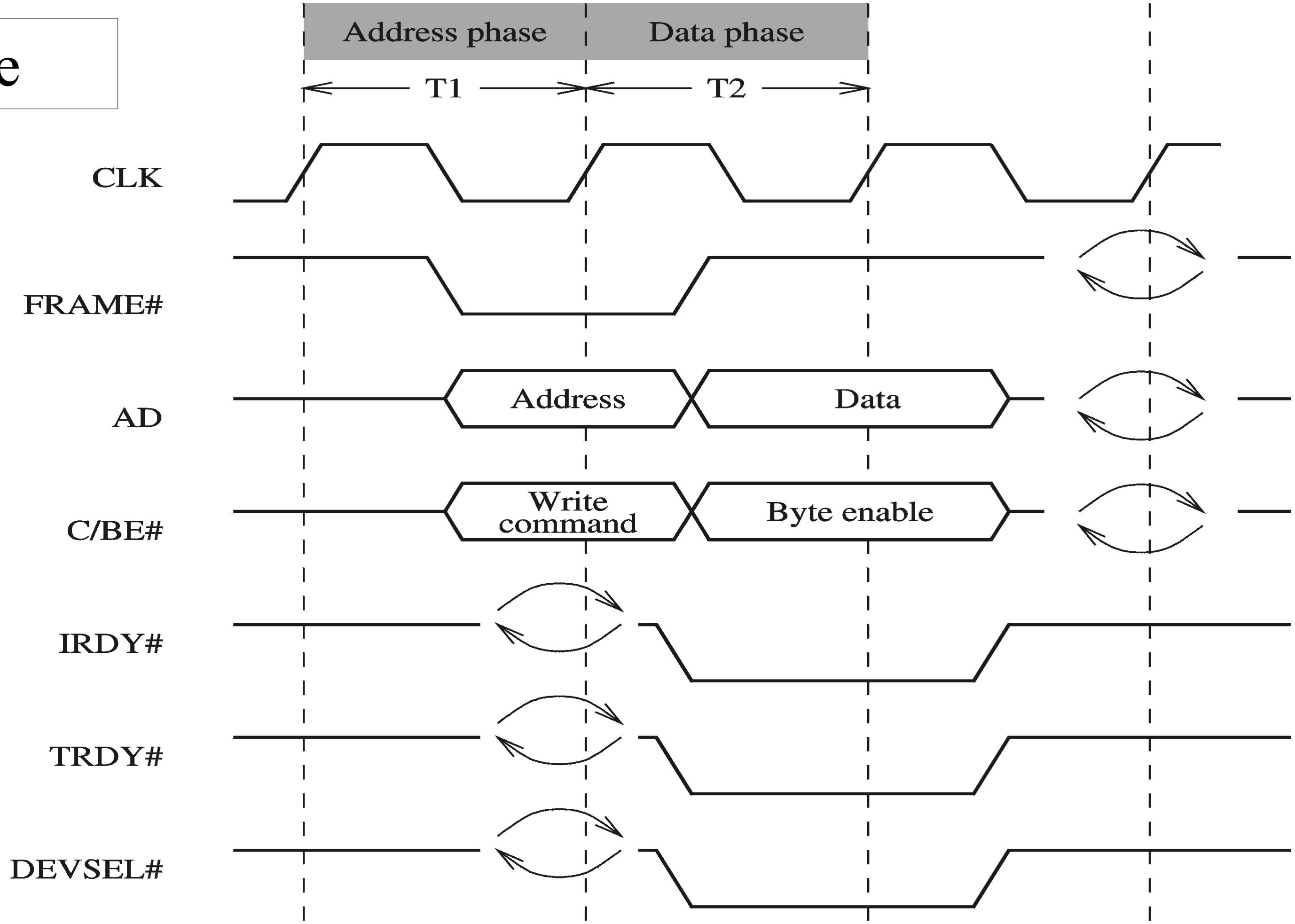
PCI Operations

PCI Read



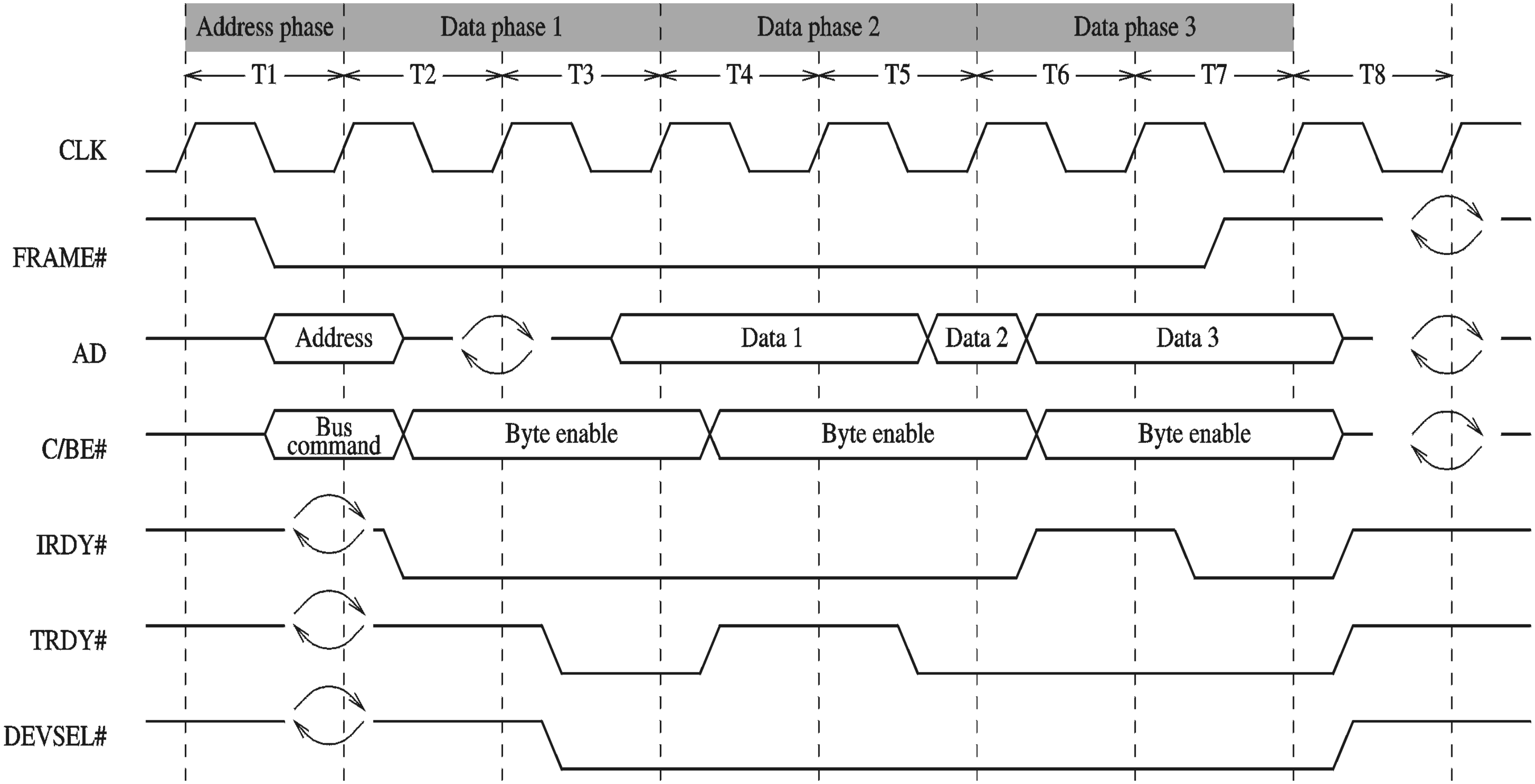
PCI Operations (cont'd)

PCI Write



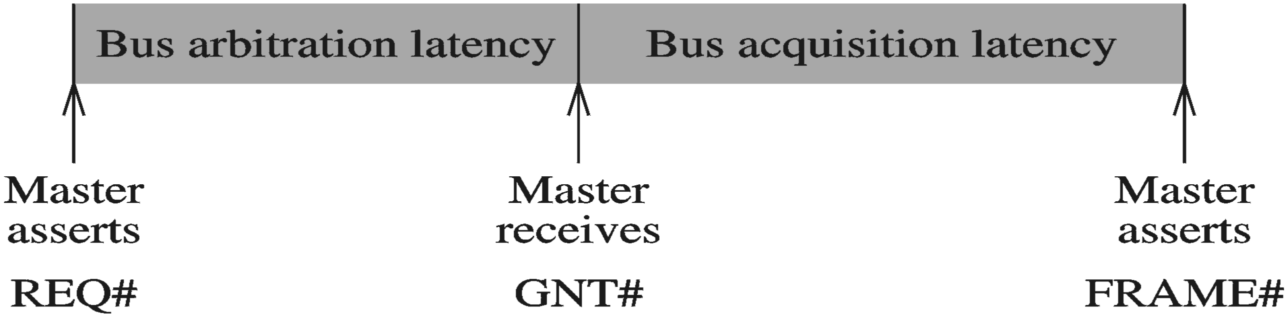
PCI Operations (cont’d)

PCI Burst Read Operation



PCI Bus Arbitration

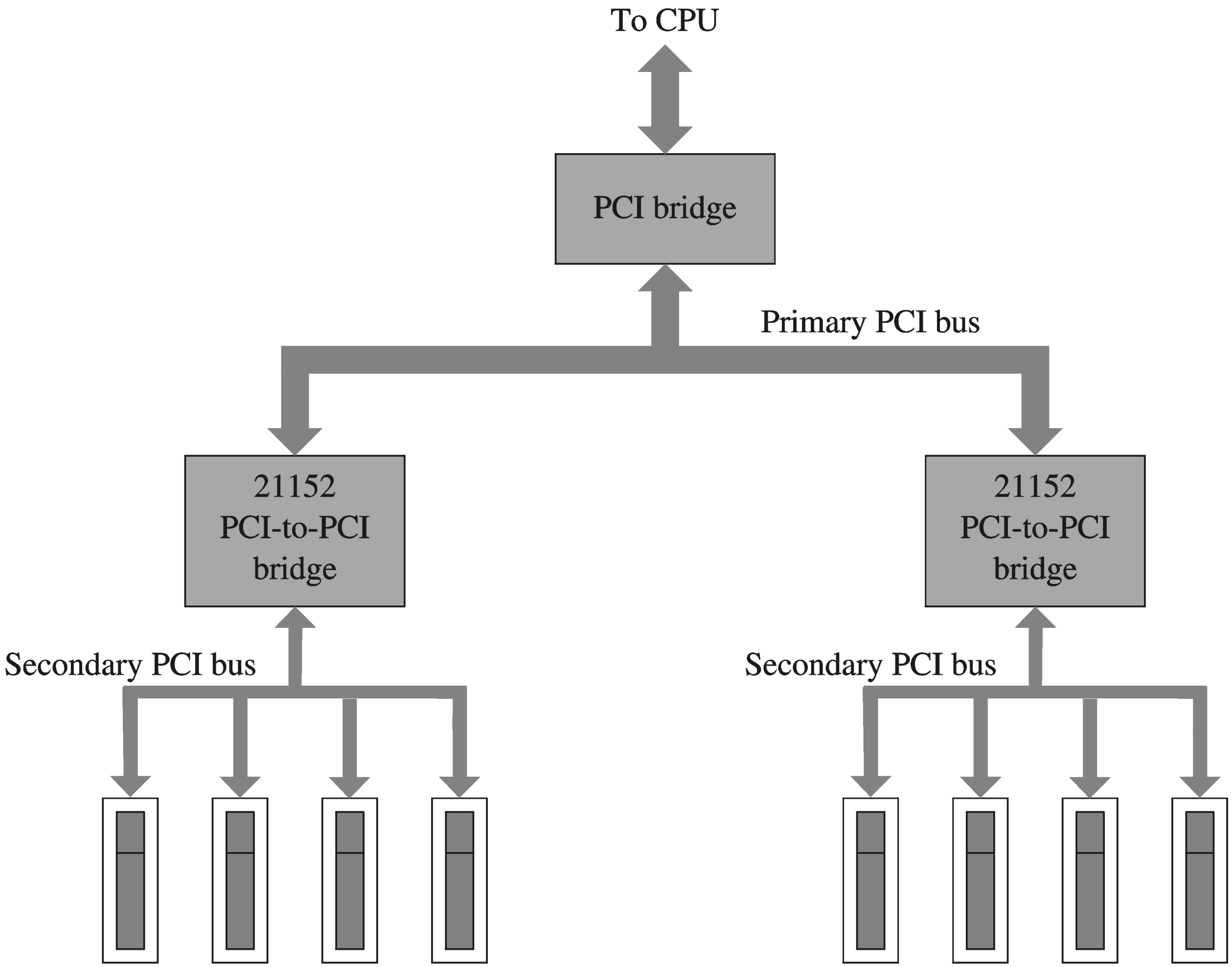
- Uses centralized arbitration
  - ∗ Independent grant and request lines
    - » REQ# and GNT# lines for each device
  - ∗ Does not specify a particular policy
    - » Mandates the use of a fair policy
- Two delay components
  - ∗ Arbitration overlaps with another transaction execution



PCI Bus Hierarchies

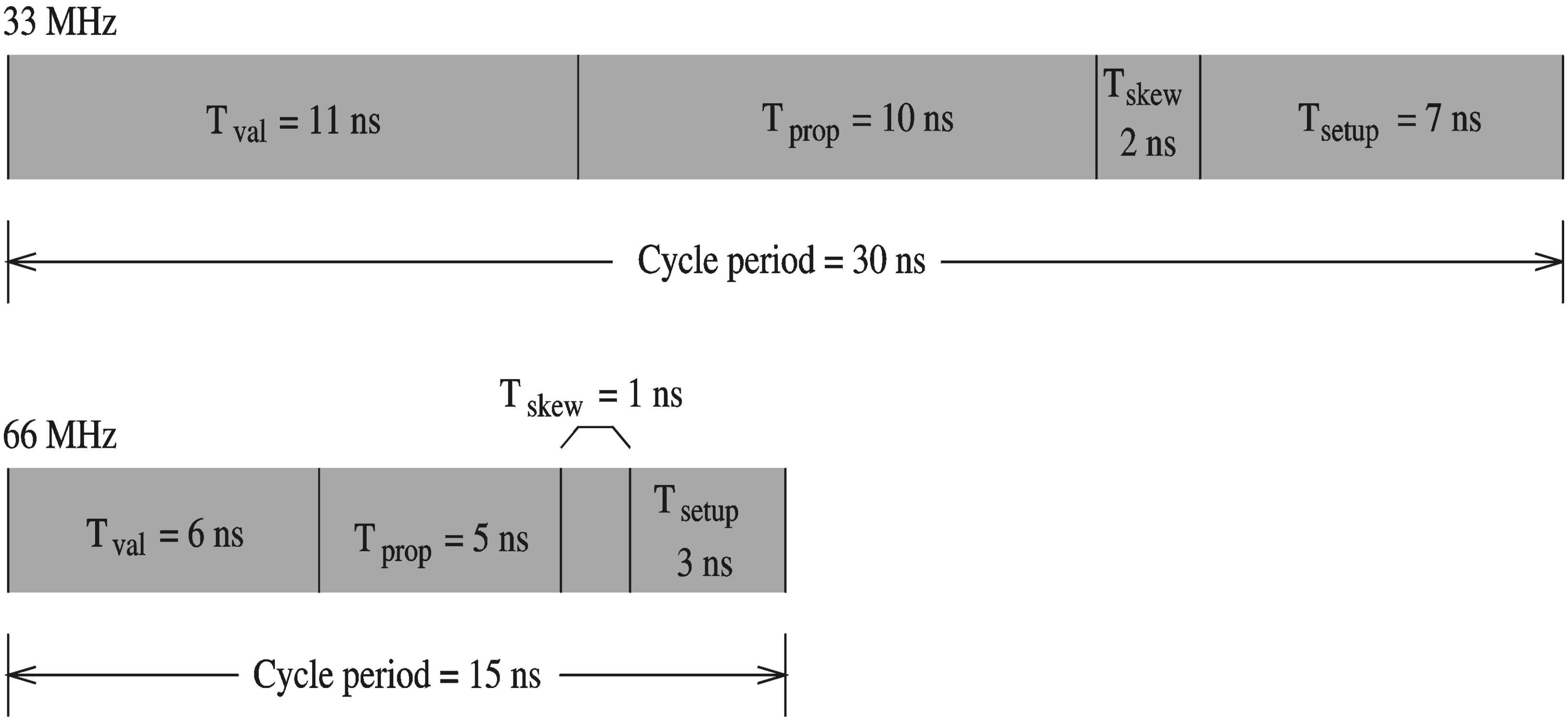
- Allows bus hierarchies
- Built using PCI-to-PCI bridges
  - ∗ Need two bus arbiters
    - » One for the primary bus
    - » One for the secondary bus
- Example: Intel 21152 PCI-to-PCI bridge
  - ∗ Secondary bus can connect up to 4 devices
  - ∗ One internal arbiter available
    - » Can be used on the secondary side
  - ∗ Need external arbiter for the primary side

PCI Bus Hierarchies (cont’d)



PCI Delays

- 66 MHz implementations pose serious design challenges
- All delays are cut in half compared to 33 MHz clock



AGP

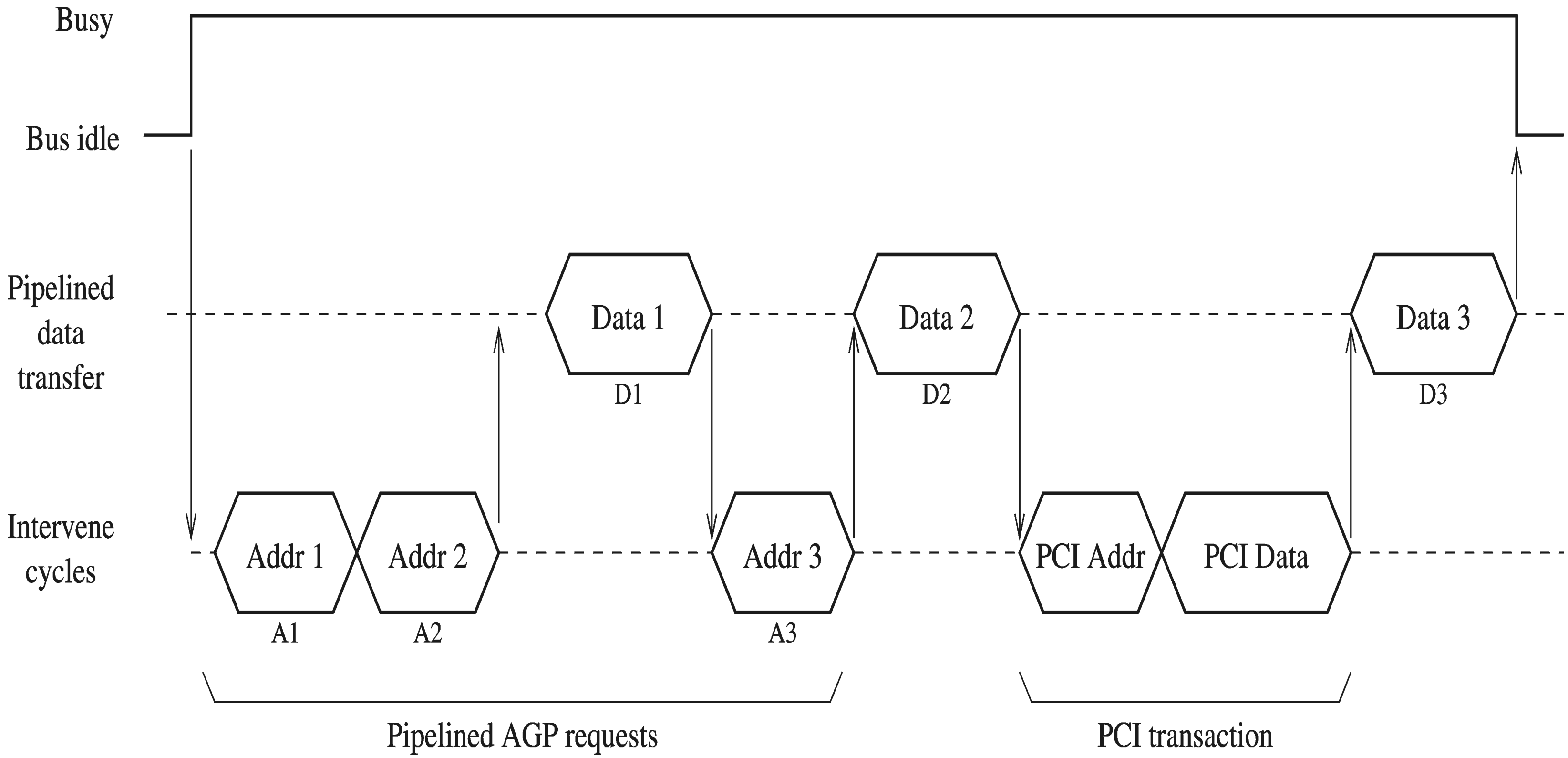
- Supports high-performance video
  - ∗ 3D graphics, full-motion video, …
  - ∗ Takes load off the PCI bus
    - » Full motion video bandwidth requirement
      - – 640 X 480 resolution: 28 MB/s
      - – 1024 X 768 resolution: 71 MB/s
      - – Twice this if displaying from DVD or hard disk
  - ∗ Not a bus, it is a port
    - » Connects just two devices
      - – CPU and video card

AGP (cont'd)

- AGP specification
  - » Based on the 66 MHz PCI 2.1 specification
  - » Retains many of the PCI signals
  - » 2X mode transmits data on the rising as well as falling edge of clock
  - » 4X and 8X speeds are available
- Performance enhancements
  - » Uses pipelining
    - – Used to hide memory latency
  - » Sideband addressing
    - – Used to partially demultiplex the bus

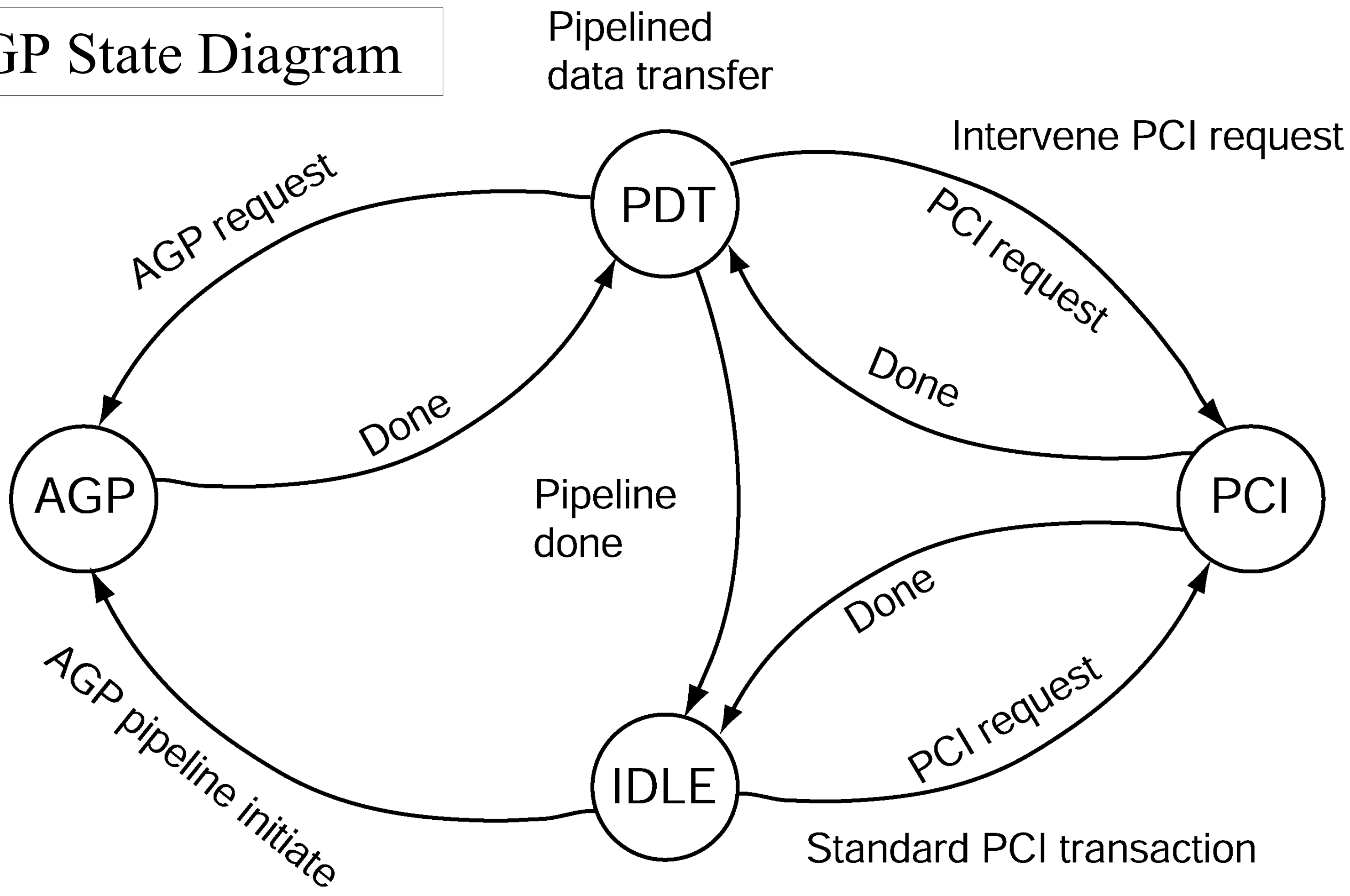
AGP (cont'd)

AGP Pipelined transmission can be interrupted by PCI transactions



AGP (cont'd)

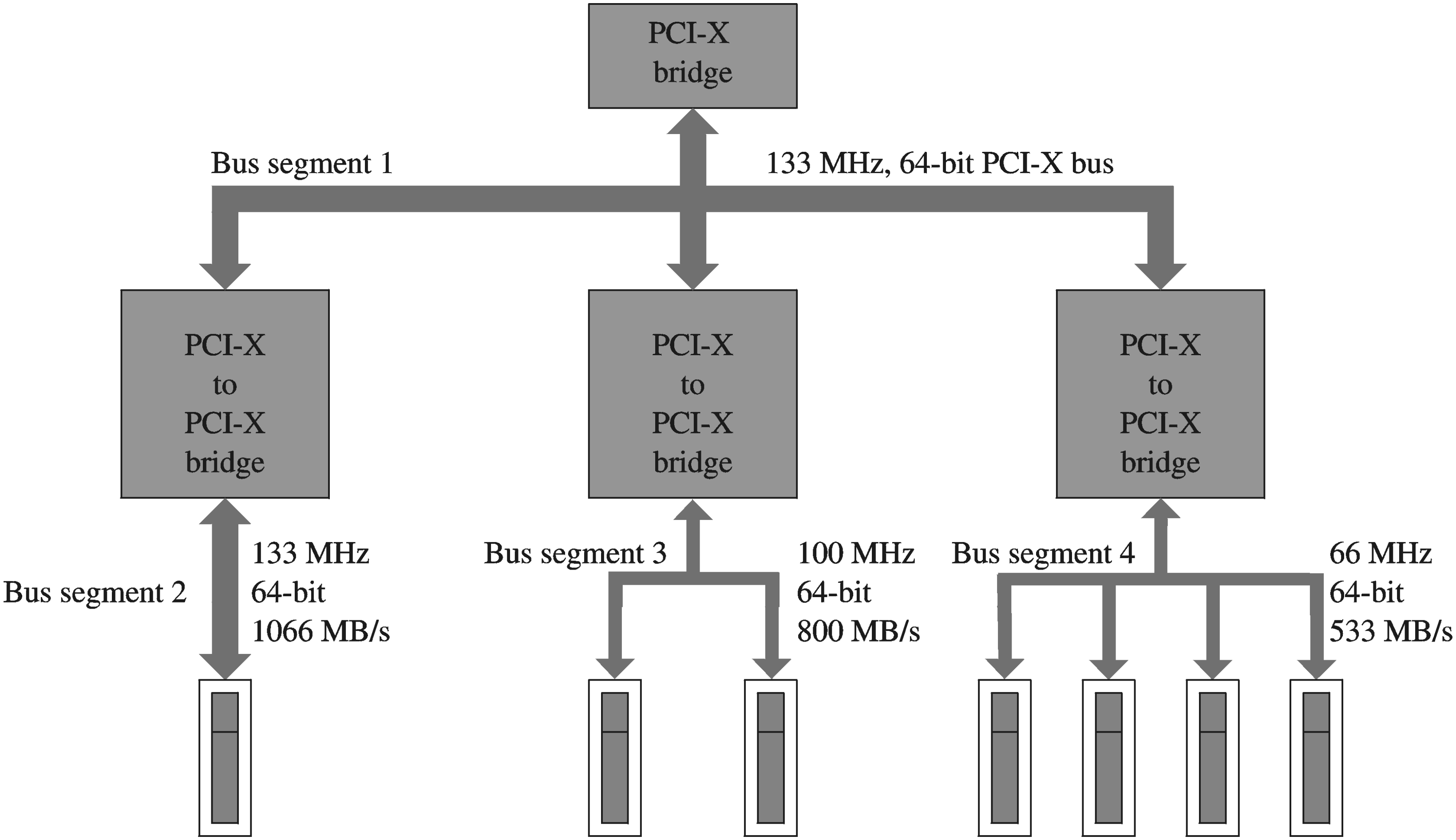
AGP State Diagram



PCI-X Bus

- Addresses the need for higher bandwidth due to
  - ∗ Faster I/O buses
  - ∗ Faster networks
- Can provide greater than 1 GB/s
  - ∗ Achieved by using 64-bit bus operating at 133 MHz
- Uses register-to-register protocol
- Can operate at three different frequencies
  - ∗ 66 MHz
  - ∗ 100 MHz
  - ∗ 133 MHz

PCI-X Bus (cont'd)



PCI-X Bus (cont'd)

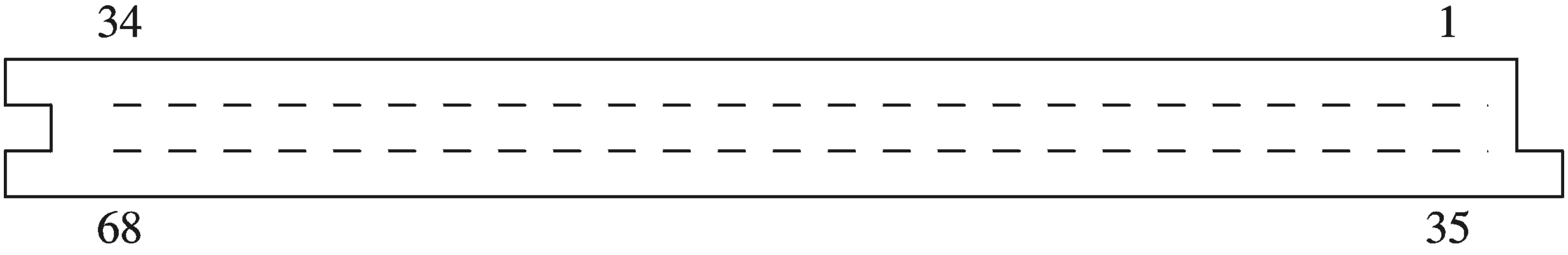
- Enhancements include
  - ∗ Attribute phase
    - » Describes the transaction in more detail than PCI
      - – Transaction size, relaxed transaction ordering, identity of transaction initiator
  - ∗ Split transaction support
    - » PCI: Treats request and reply as a single transaction
    - » PCI-X: Splits them into two transactions
  - ∗ Optimized wait states
    - » Bus can be released due to split transaction support
  - ∗ Standard block size movement

PCMCIA Bus

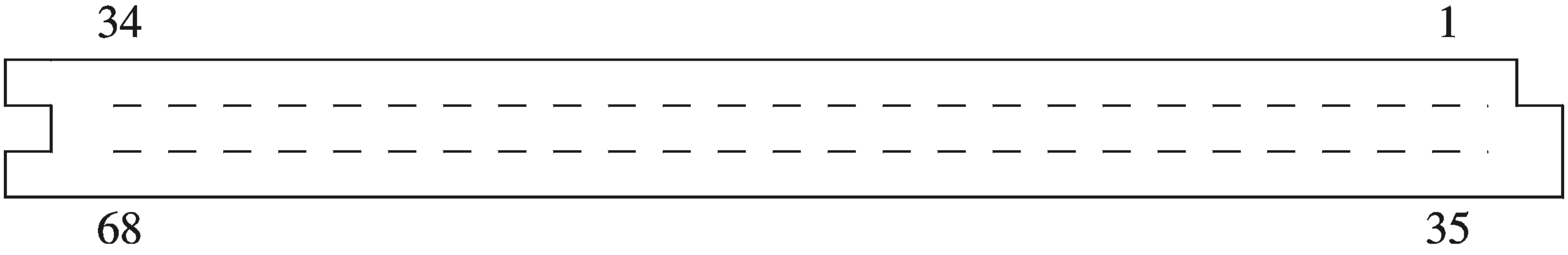
- Started as a standard for memory cards
  - » PCMCIA = Personal Computer Memory Card International Association
    - – Also called PC Card standard
  - » First standard (version 1.0) released in 1990
  - » Release 2.0 also supports I/O devices
  - » Small form factor
    - – 85.5 mm X 54 mm (credit card size)
  - » Thickness varies
    - – Type I: 3.3 mm thick (used for memory)
    - – Type II: 5 mm thick (I/O devices---modems, NICs)
    - – Type III: 10.5 thick (I/O devices --- hard drives…)

PCMCIA Bus (cont'd)

- PCMCIA connectors (68 pins)
  - ∗ Sockets are keyed such that low-voltage card cannot be inserted into a 5 V standard socket



(a) Standard card connector



(b) Low-voltage card connector

PCMCIA Bus (cont’d)

- PCMCIA supports three address spaces
  - ∗ Common address space (64 MB)
    - » Used for memory expansion
  - ∗ Attribute address space (64 MB)
    - » Used for automatic configuration
  - ∗ I/O address space
    - » I/O devices use either Type II or III card
- PC card uses 16-bit data bus
- Also defines a 32-bit standard
  - ∗ It is called CardBus

PCMCIA Bus (cont’d)

- Memory Interface
  - ∗ Address signals
    - » Address lines (A0 – A25)
      - – Support 64 MB of address space
    - » Card Enable Signals (CE1#, CE2#)
      - – Similar to chip select
      - – Controls data transfer
        - ✈CE1 = CE2 = High : No data transfer
        - ✈CE1 = low, CE2 = High : Data transfer lower data path (D0 – D7)
        - ✈CE1 = high, CE2 = low : Data transfer upper data path (D8 – D15)
        - ✈CE1 = CE2 = low : 16-bit data transfer (D0 – D15)

PCMCIA Bus (cont'd)

- Memory Interface
  - ∗ Transaction Signals
    - » Data lines (D0 – D15)
      - – Used to transfer data
    - » Output Enable (OE#)
      - – Memory read signal
    - » Write Enable (WE#)
      - – Memory write signal
    - » Wait Signal (WAIT#)
      - – Can be use to extend the transaction cycle
    - » Register Select (REG#)
      - – Used to select common memory (high) or attribute memory (low)

Four types of memory transactions:

1. Common memory read
2. Common memory write
3. Attribute memory read
4. Attribute memory write

PCMCIA Bus (cont'd)

- Memory Card Status Signals
  - Card Detect Signals (CD1#, CD2#)

CD1#	CD2#	Interpretation
0	0	Card properly inserted
0	1	Card improperly inserted
1	0	Card improperly inserted
1	1	No card inserted

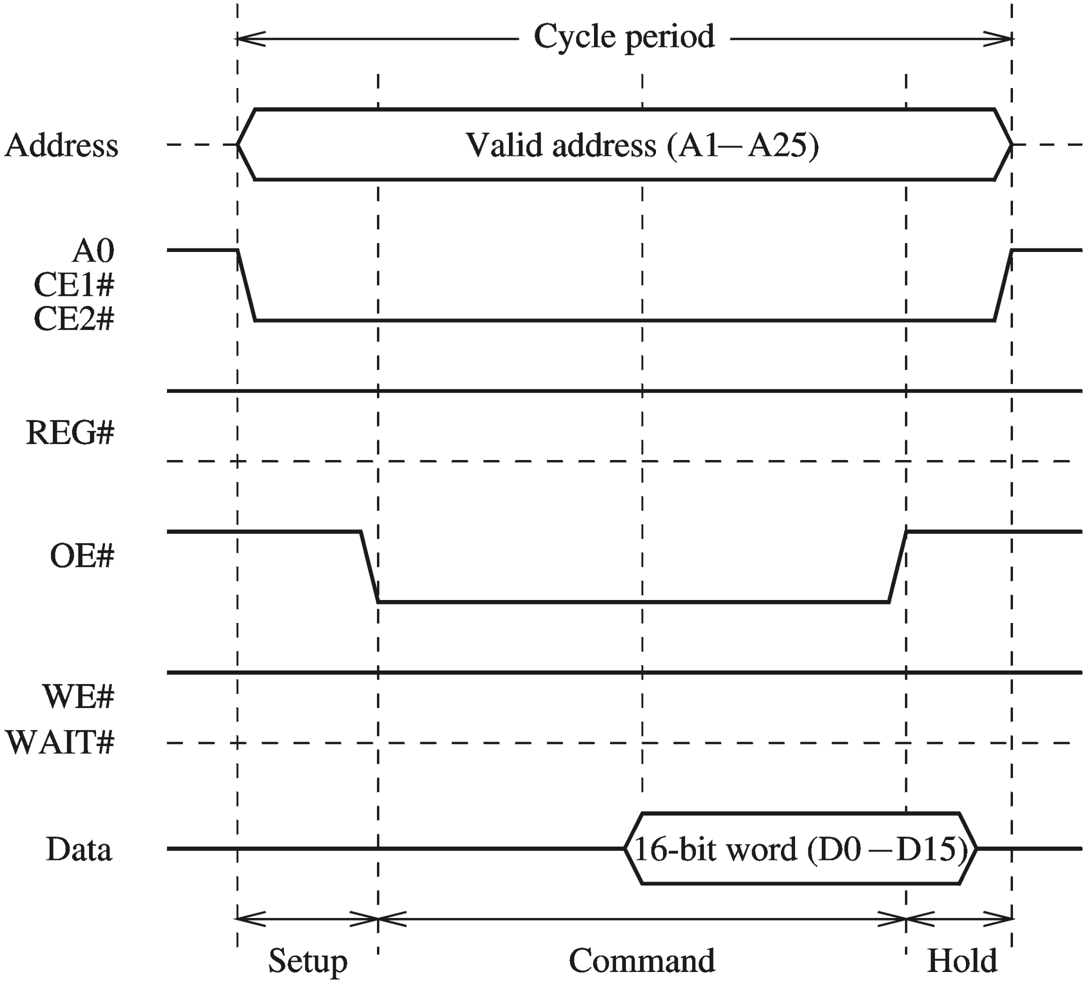
PCMCIA Bus (cont’d)

- Ready/Busy Signal (READY)
  - High: indicates the card is ready to be accessed
  - Low: Busy executing a command
- Write Protect (WP)
  - Gives status of the write protect switch on the card
- Battery Voltage Detect Signals (BVD1, BVD2)

BVD2	**BVD1**	**Interpretation**
0	0	Battery cannot maintain data integrity
0	1	Battery replacement warning
1	0	Battery cannot maintain data integrity
1	1	Battery is in good condition

PCMCIA Bus (cont'd)

- 16-bit Common memory read cycle
- No wait states



PCMCIA Bus (cont'd)

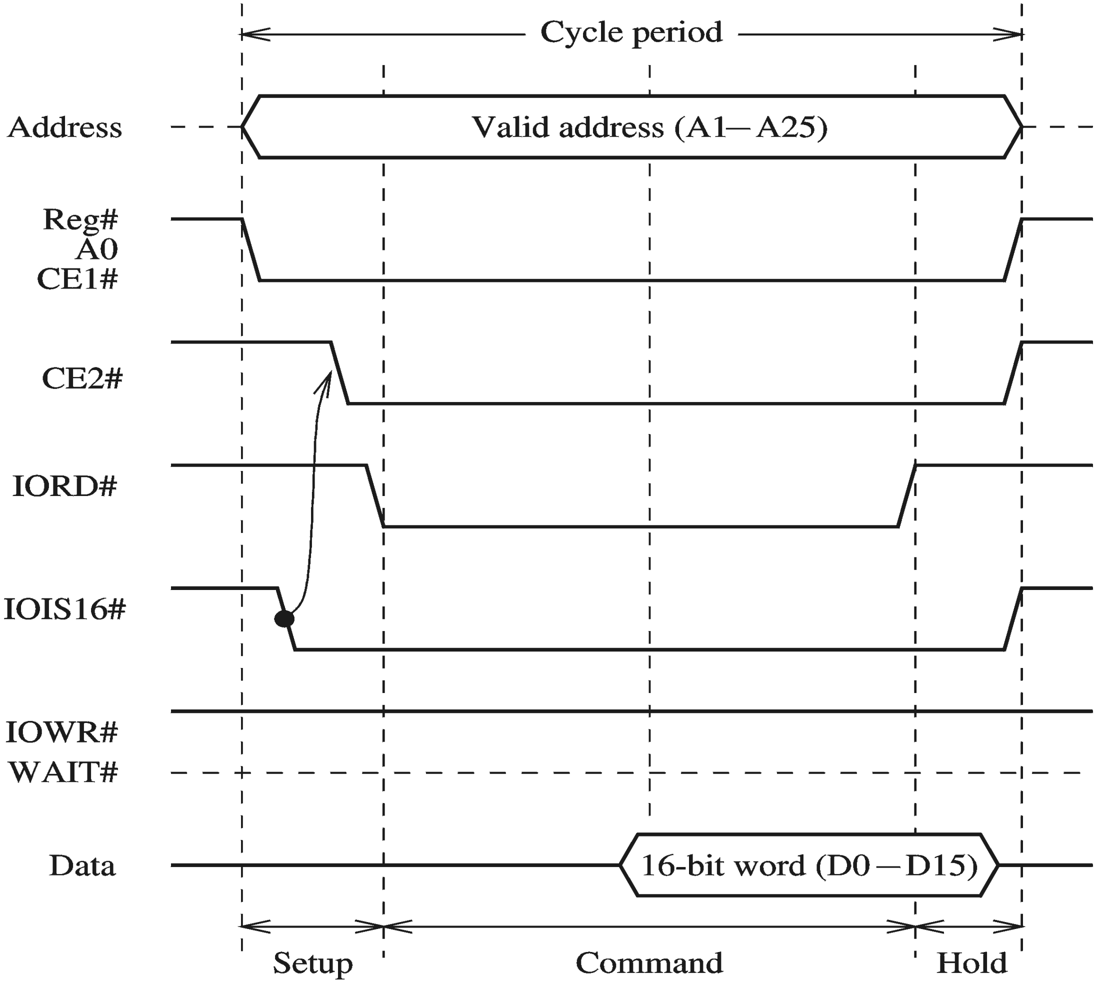
- I/O Interface
  - ∗ Uses Type II or III card
    - » Some memory signals are changed for I/O interfacing
    - » Some reserved signals are also used for I/O interfacing
  - ∗ I/O Read and Write (IORD#, IOWR#)
    - » Reserved in memory interface
  - ∗ Interrupt Request (IREQ#)
    - » Replaces memory READY signal
    - » PC card asserts to request interrupt service

PCMCIA Bus (cont’d)

- I/O Size is 16 Bits (IOIS16#)
  - Replaces Write Protect memory signal
  - Low: Indicates I/O is a 16-bit device
  - High: Indicates I/O is a 8-bit device
- System Speaker Signal (SPKR#)
  - Sends audio signal to system speaker
- I/O Status Change (STSCHG#)
  - Replaces BVD1 memory signal
  - Useful in multifunction PC cards
    - Containing memory and I/O functions

PCMCIA Bus (cont'd)

- 16-bit I/O read cycle
- No wait states



Last slide