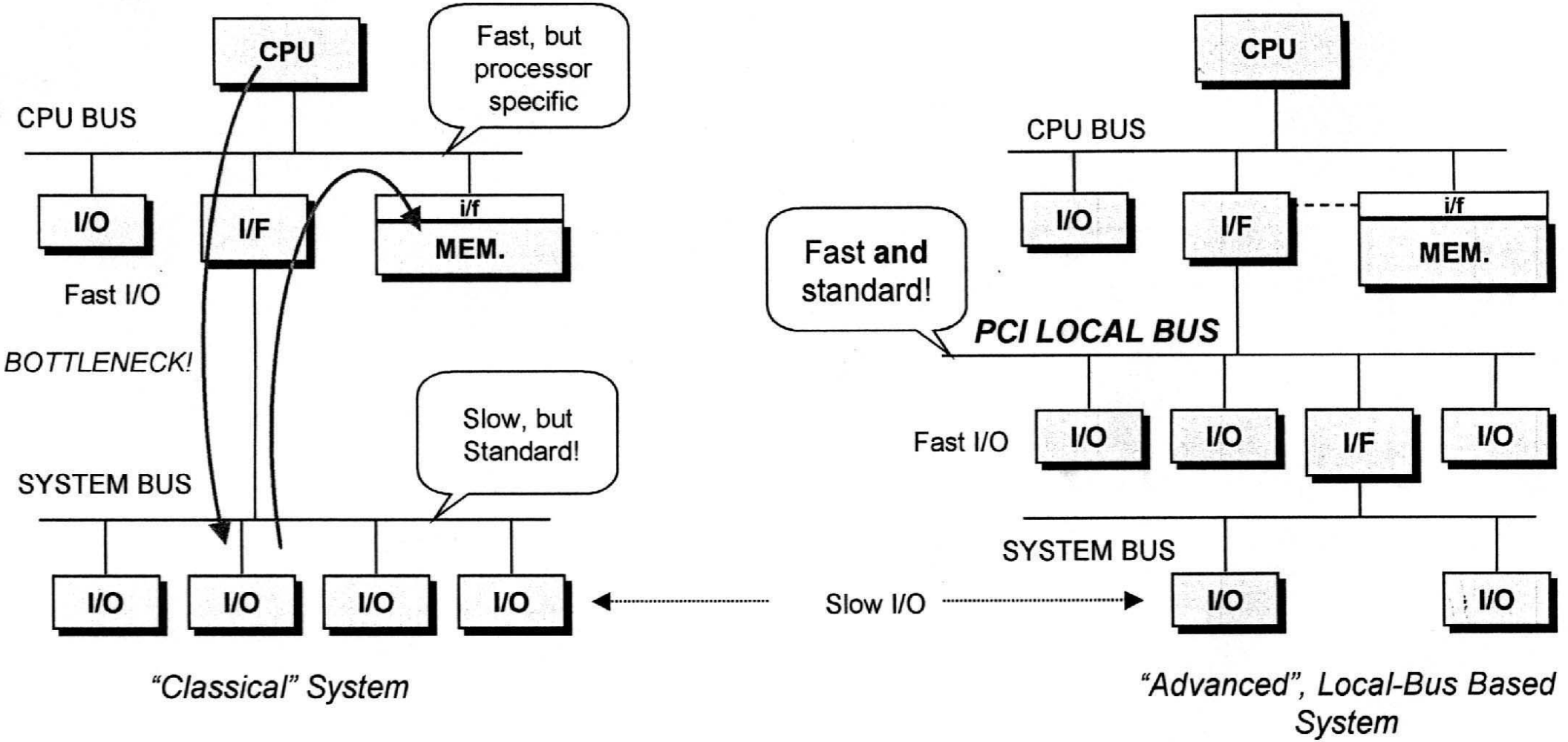
Definition

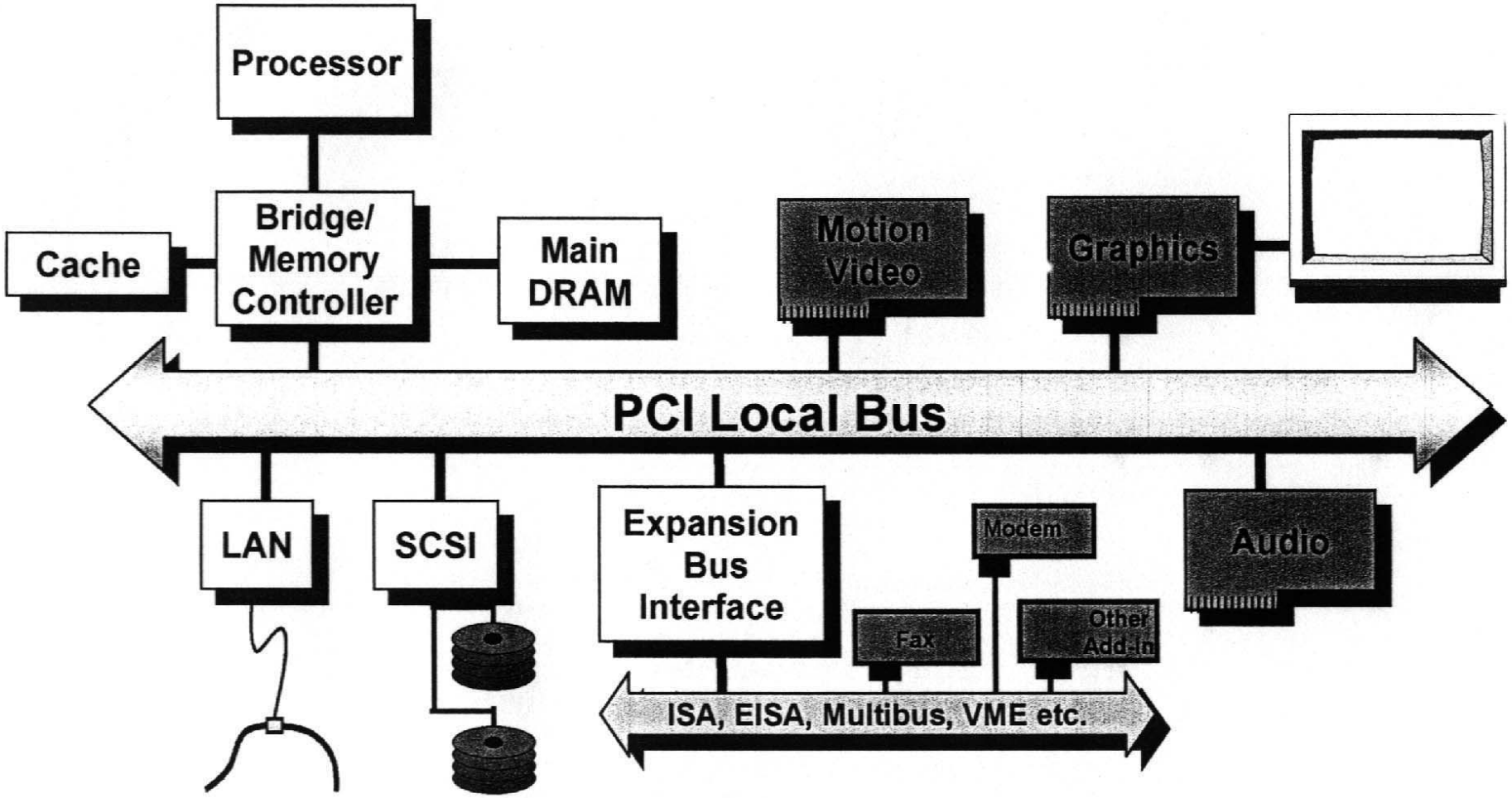
- PCI - *Peripheral Components Interconnect* implies:
  - A local bus
  - Originally designed for glueless interconnection of high performance peripheral components such as graphic accelerators, disk controllers etc.
- Initiated by Intel Corp.
- Currently an open, public standard, maintained by the "PCI Special Interest Group"

Motivation



- Providing large bandwidth connection to the I/O subsystem as exists for the memory subsystem in a PC
- Data bottlenecks created in OS/2 and Windows' "heavy" applications
- Moving critical I/O functions closer to the CPU (local bus approach) proved to be the right solution
- A local bus is required, but open and standardized like the industry standards for system I/O buses

PCI System Block Diagram

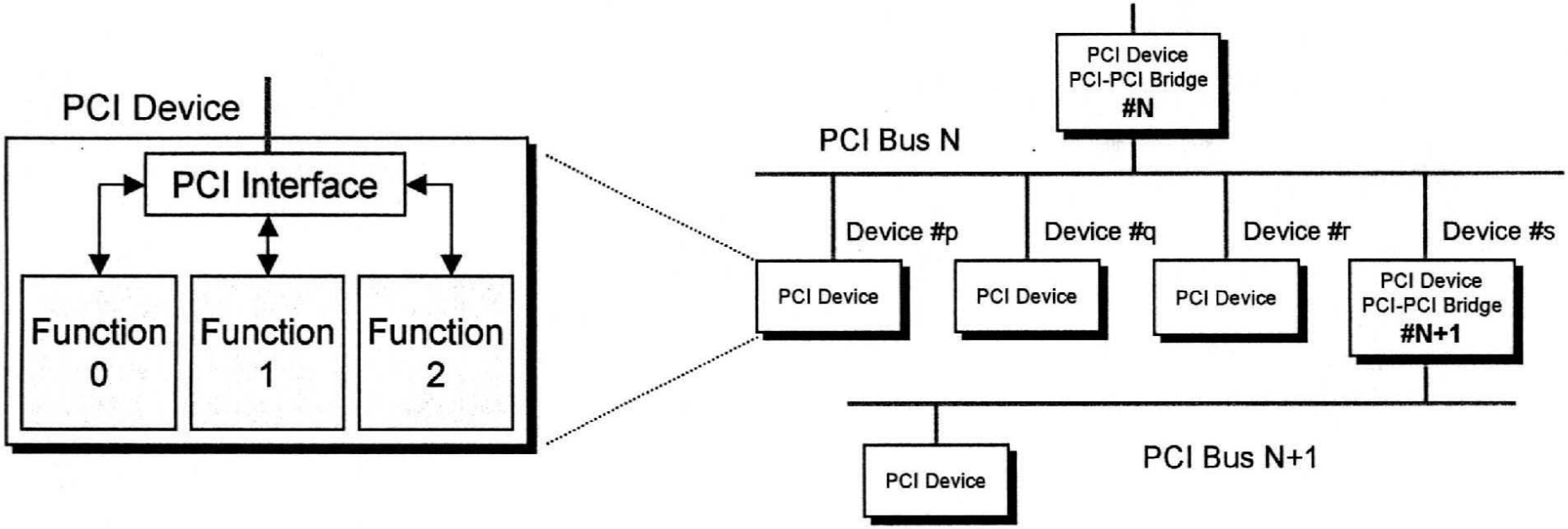


- The "PCI bridge" connects between the CPU/Cache/Memory subsystem and the PCI bus
- It provides low latency path for the CPU to access any PCI device mapped into the I/O or memory space
- It provides path for PCI masters to access main memory

PCI Bus Features (a)

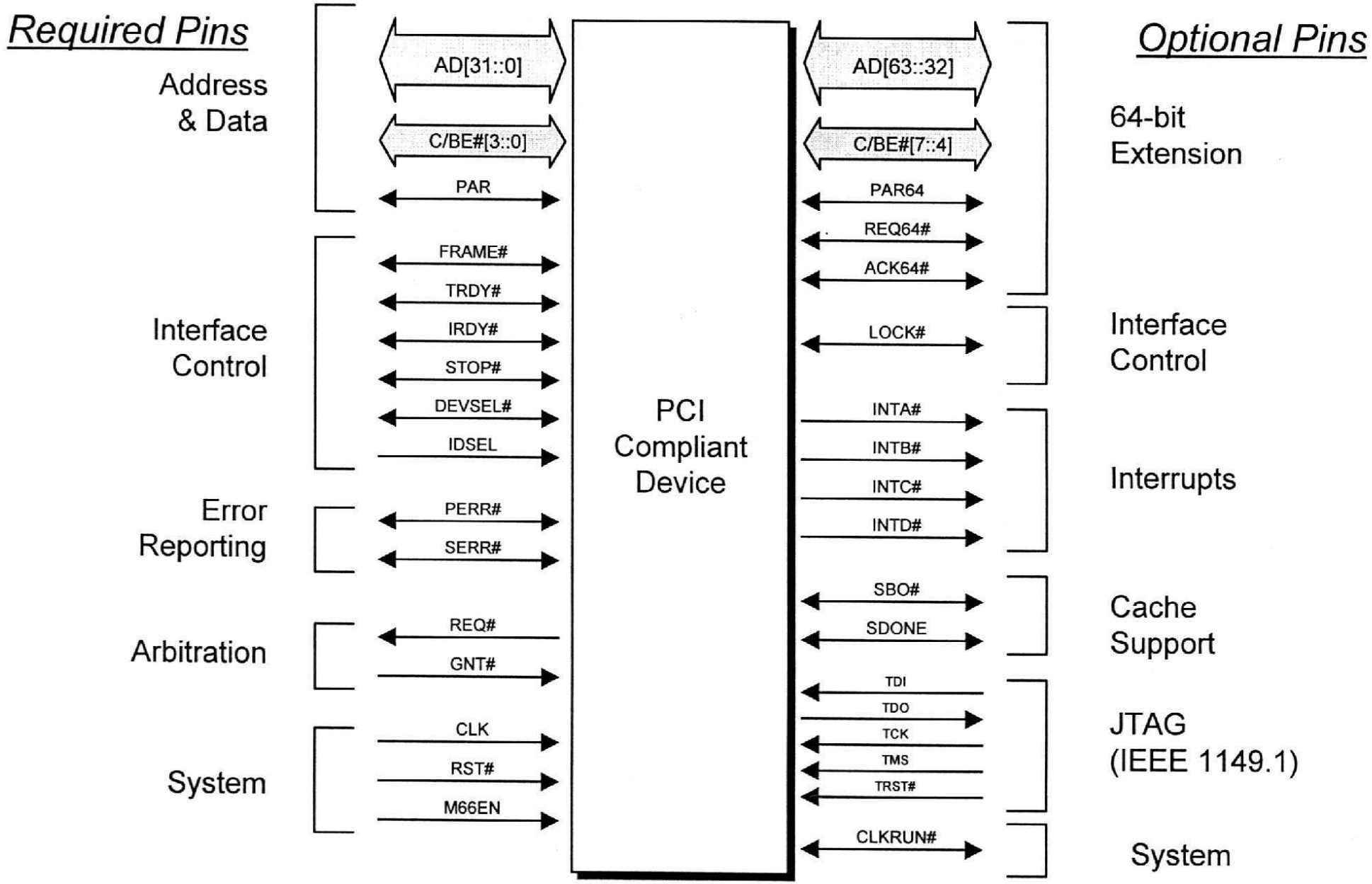
- Processor Independent, fits all levels of computing
  - Mobile and Desktop PC's, Servers, Workstations, Embedded Systems
- Supports both 5V and 3.3V Signaling Environments
- High Performance
  - Synchronous to a clock of up to 33MHz (most common clock)
  - Specification defines working with 66 MHz clock
  - Burst modes for both read and write
  - 32-bit data width, 132 MByte/Sec peak throughput (33 MHz clock)
  - Supports 64-bit data width, 264 MByte/Sec peak throughput (33 MHz clock)
  - Capable of full concurrency with the processor/memory subsystem
  - Hidden (overlapped) central arbitration
- Ease of use
  - Enables full auto configuration support of PCI local bus add-in boards and components. PCI devices contain registers with the device information required for configuration

Device/Function Numbering



- Each PCI device may include from 1 to 8 internal, independent functions, numbered 0 to 7.
- These functions are separate hardware entities that share the same PCI-Bus interface
- A specific function is referred to by "Bus Number / Device Number / Function Number"
- This type of reference is used in Configuration accesses to a specific function

PCI Signal Groups



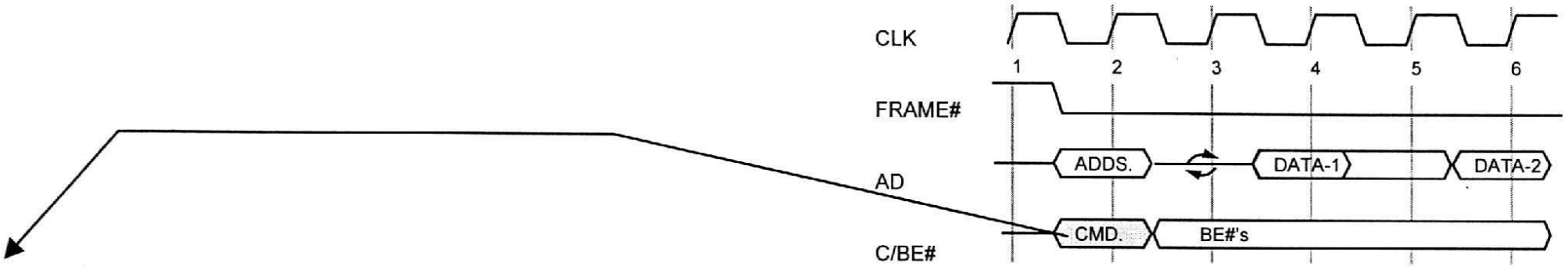
Signal Types

Type	***Definition***	***Where Used***
OUT	Totem-Pole output of a standard driver	Test Output (TDO), SBO#, SDONE
IN	Standard input-only signal	RST#. CLK, IDSEL, M66EN CLKRUN#, Test inputs
t/s	Bi-directional, Tri-State input/output	Address/Data lines , Parity Control/Byte-Enable lines, Arbitration signals
s/t/s	Sustained Tri-State[1] A pull-up resistor is required	Control signals, PERR#
o/d	Open Drain[2] A pull-up resistor is required	SERR#, Interrupt lines

[1] This signal must be driven HIGH for at least one clock before being floated. A new driver must wait at least 1 clock before driving the line. A pull-up resistor is required to sustain the HIGH level at the time no driver drives the line

[2] This signal allows multiple devices to share as a wired-OR. A pull-up resistor is required to sustain the inactive state until another agent drives it

PCI Bus Commands



C/BE#[3::0]	Command Type	Notes
0010	**I/O Read**	
0011	**I/O Write**	
0110	**Memory Read**	
1110	Memory Read Line	Same as Memory Read, indicates that the Master intends to make more than two 32-bit data phases
1100	Memory Read Multiple	Same as Memory Read, indicates that the Master intends to fetch more than one cache-line
0111	**Memory Write**	
1111	Memory write & invalidate	Same as Memory Write, indicates that the Master intends to write all bytes in the addressed cache-line
1010	**Configuration Read**	
1011	**Configuration Write**	
0000	Interrupt Acknowledge	Address bits are "Don't Care". Agent with the system Interrupt Controller should supply the vector
0001	Special cycle	Message broadcast to all targets, no DEVSEL# and no TRDY# generated
1101	Dual address cycle	Used to transfer 64-bit address to devices that support 64-bit addresses
0100, 0101, 1000, 1001	Reserved	A Target should not respond

PCI Required Signals (a)

- CLK — The basic PCI bus clock. Its rising edge is for timing of all signals (except RST# and IRQx#). From 0 to 66 MHz

- RST# — Sets all PCI agents' bus specific registers and signals to a known, initial state

- AD[31::00] — Multiplexed address and data lines:
  - During **address** phase:
    - I/O space Byte address
    - Memory space Dword address
    - Config. space Dword address
  - During **data** phase:
    - AD[31::24] MSB
    - .....
    - AD[07::00] LSB

- C/BE[3::0]# — Command/Byte enable signal
  - During **address** phase: Bus command
  - During **data** phase: Byte enable

- PAR — Even parity bit for AD[31::00] and C/BE[3::0]

PCI Required Signals (b)

- FRAME# — Asserted by the Master during a transaction. Its beginning indicates the address phase. When deasserted, indicates the final data phase
- IRDY#/TRDY# — Initiator (Master)/Target handshake, used also to generate wait-states
- STOP# — Request by the Target to the Master to stop the current transaction
- LOCK# — Indicates an "atomic" operation that may require multiple bus transactions to complete. Only one Master can own the control on this signal at a certain time. When a Master that asserts LOCK# accesses a memory Target that supports LOCK#, the Target will lock the corresponding memory area. Other PCI Masters can still access, other, non-locked memory areas
- IDSEL — A "chip select" signal to an agent during configuration read or write

PCI Required Signals (c)

- DEVSEL# — Driven by a Target that detects itself as being selected. From a Master's point of view when this signal is received, it means that an agent on the bus has been selected
- REQ#/GNT# — Arbitration (point to point) lines, routed between each Master and a central arbitration logic. The PCI specification does not define any specific arbitration scheme
- PERR# — Data parity error. An agent that receives data (master during read or target during write) asserts this line while detecting a parity error. Not used for Special Cycle"s data
- SERR# — Address parity error or data parity error detected during special bus cycle, or any error with possible catastrophic results. Can by used by a PCI bridge or the Central Resource to generate NMI to the host CPU
- M66EN — Indicates to a device and to the clock generator that the bus is operating at 66 MHz. This line is bussed through all connectors and pulled up to Vcc by a pull-up resistor. An add-in card capable of 33 MHz only will have the corresponding pin grounded, resulting in automatic clock speed reduction by the clock generator, or in a warning message issued by the configuration software

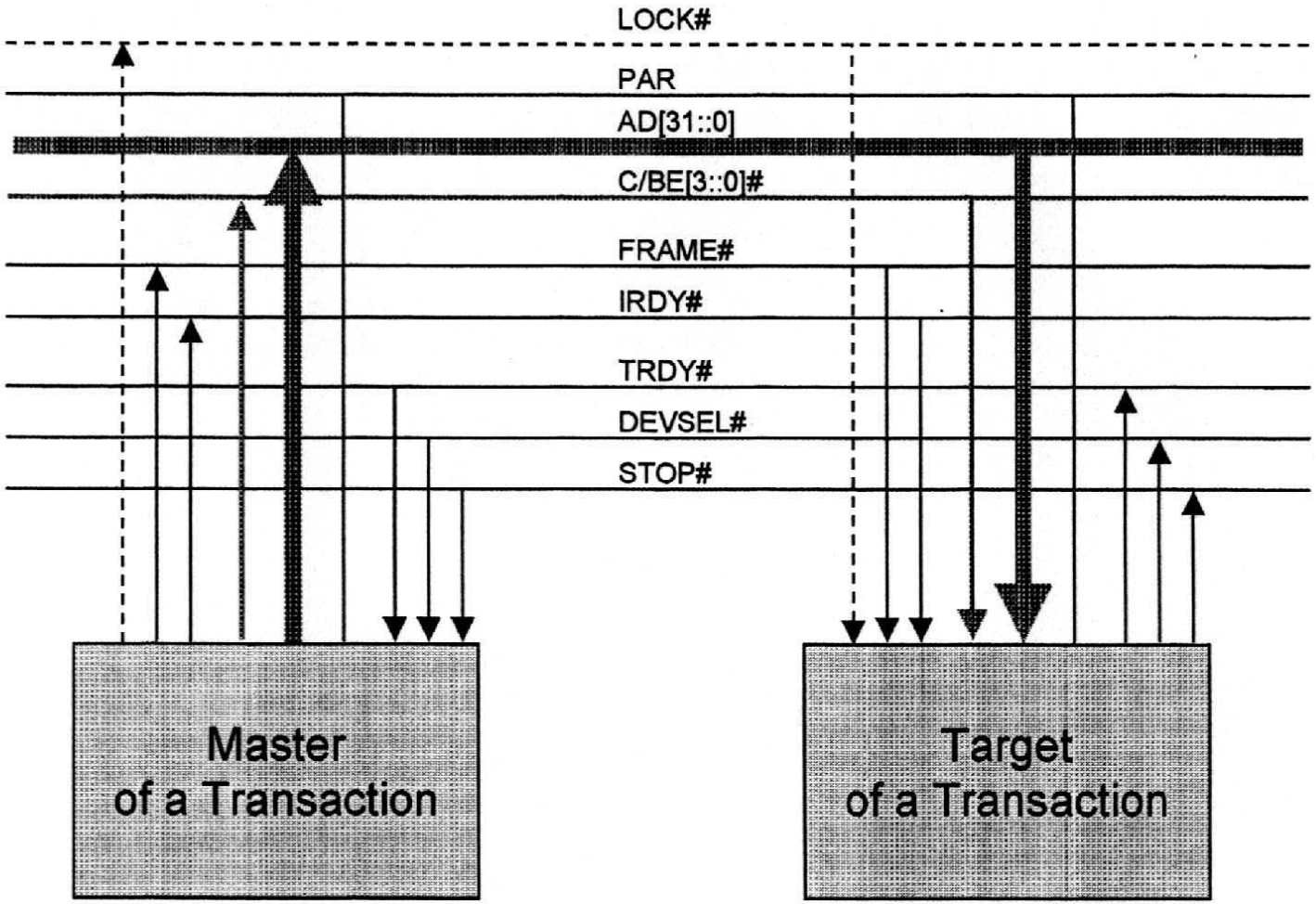
PCI Optional Signals (a)

- INTA# INTB# INTC# INTD# — Level Sensitive, asynchronous interrupt outputs, routed to the system interrupt controller. Single-function devices may use INTA# only, whereas multifunction devices incrementally use INTB#, INTC# and INTD#
- SBO# — Snoop Backoff. Indicates a hit to a modified line. When deasserted (and SDONE is asserted) it indicates a "clean" snoop result. Used for cache support
- SDONE — Snoop Done. Indicates the status of the snoop for the current access. Used for cache support
- PRSNT[1:2]# — Indicates to the motherboard whether an add-in board is physically present in the slot, and if present, its total power requirement. **These two signals are optional for the motherboard but required on add-in boards**
- CLKRUN# — Used as an input for a device to determine the status of CLK and an open-drain output used by the device to request starting or speeding-up CLK

PCI Optional Signals (b)

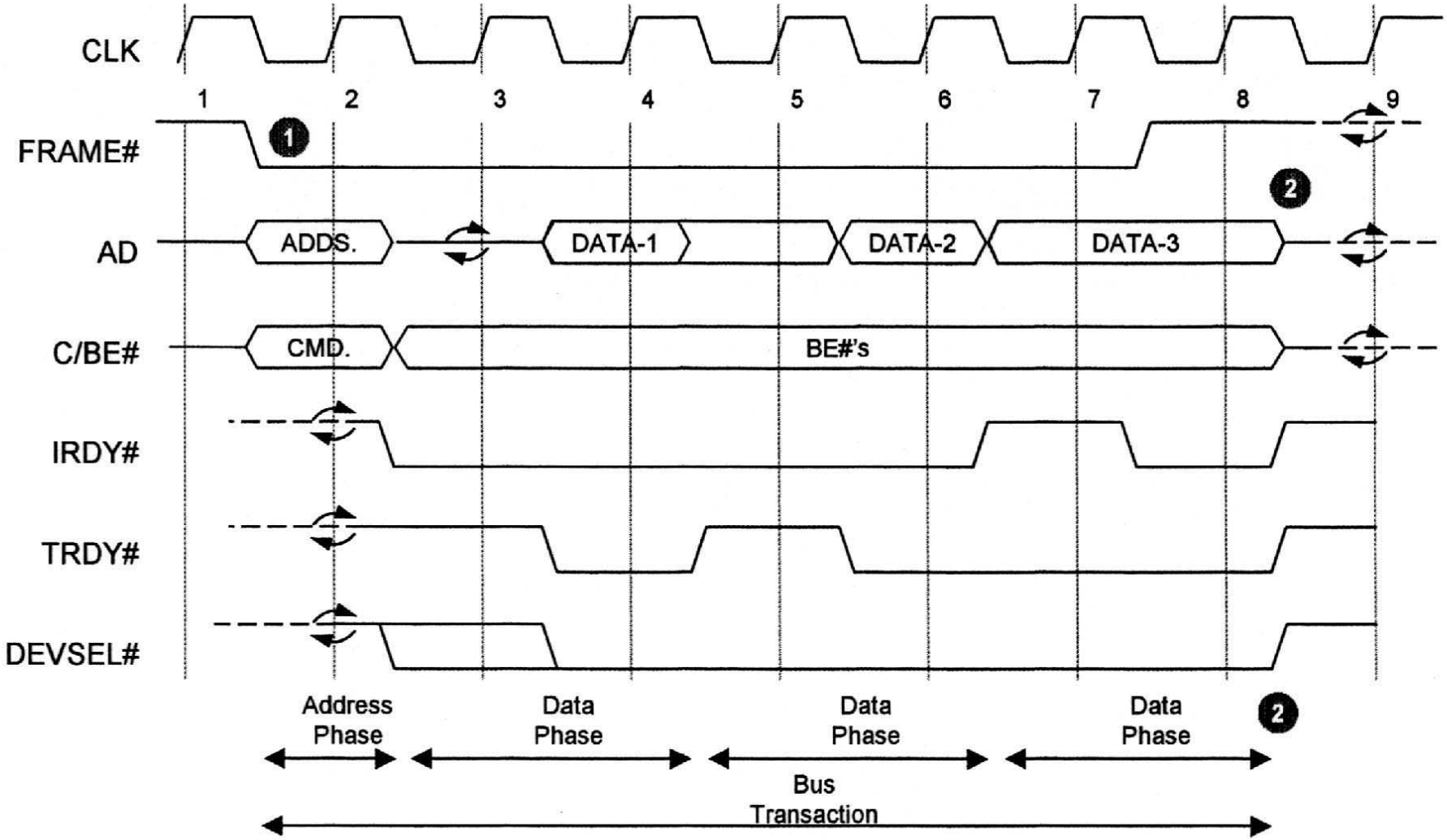
- AD[63::32] — Multiplexed Address/Data lines that extend data transfers to 64 bits and addressing to 64-bit address space
- C/BE[7::4]# — During Dual-Address address phase, these line carry the bus command. During the data phases they indicate the meaningful bytes
- REQ64# — Used by a Master to indicate its intention to transfer data using 64 bits. Same timing as FRAME#
- ACK64# — Used by the selected target to indicate its willingness to transfer data using 64 bits. Same timing as DEVSEL#
- PAR64 — Even Parity bit for AD[63::32]# and C/BE[7::4]#
- TCK TDI TDO TMS TRST# — IEEE 1149.1 compliant Test Access Port (TAP) signals

Master and Target of a Transaction



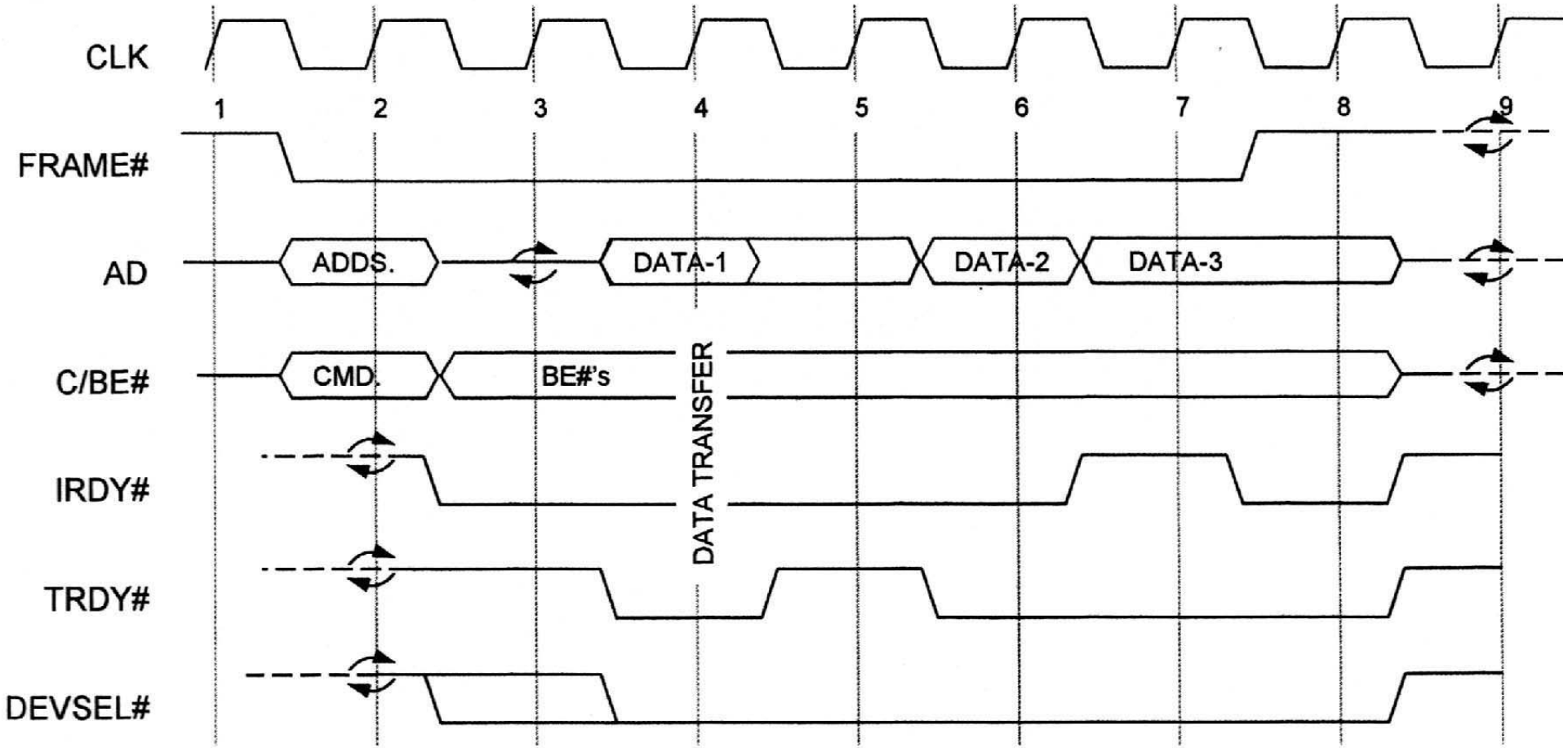
- Only main signals are shown. All other signals are implied
- Address always driven by the Master
- Data driven by Master in Write Transactions and by Target in Read Transactions
- PAR is driven by the agent who drives AD lines
- LOCK# is driven by a Master who has acquired the right to control this signal (the current "Lock-Master")

Basic Read (a)



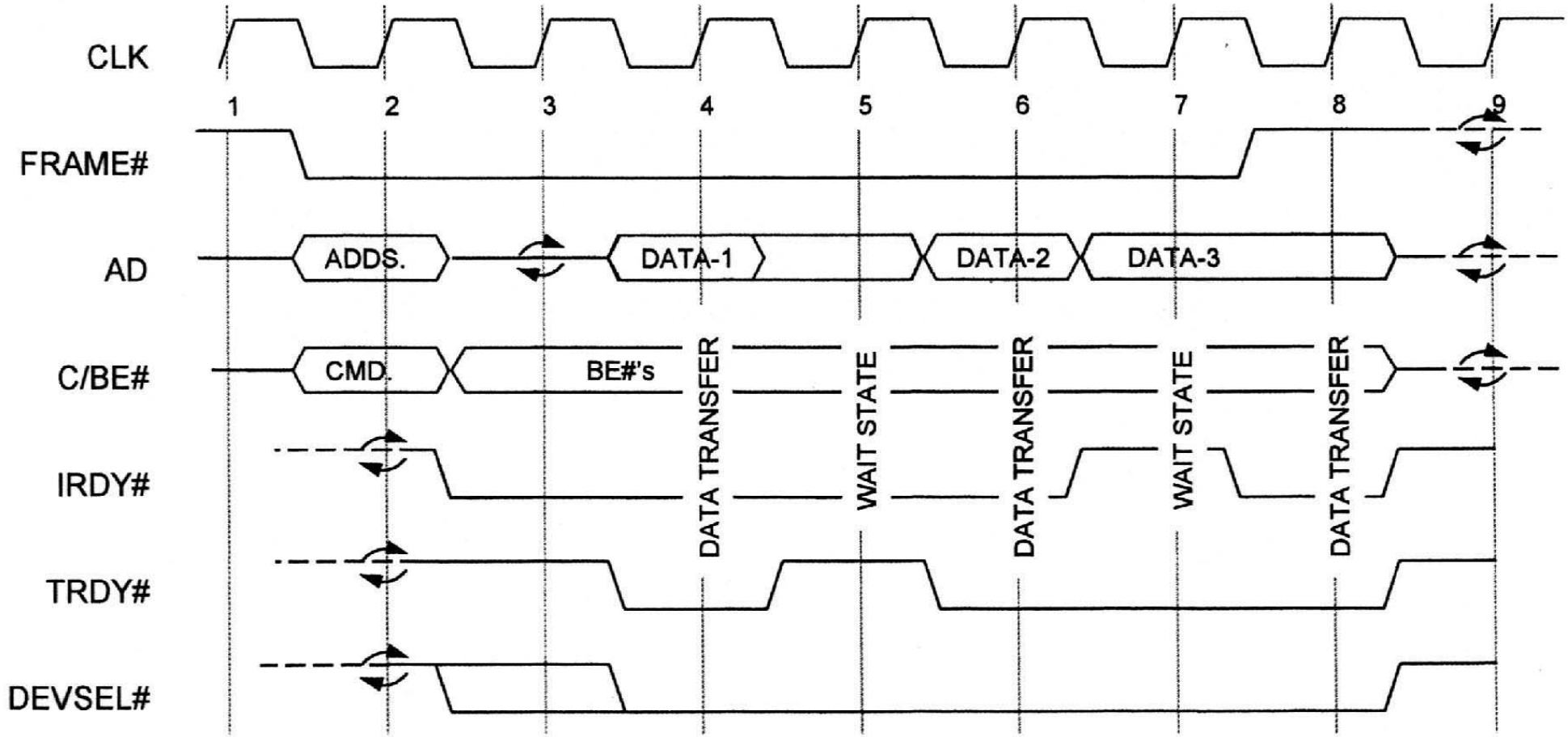
- A Bus Transaction begins when FRAME# is asserted (**1**). Its first "Phase" is "Address Phase" which lasts 1 clock period
- The Transaction ends when all "Data Phases" are complete, the control signals are deasserted and AD and C/BE# lines are tri-stated (**2**)

Basic Read (b)



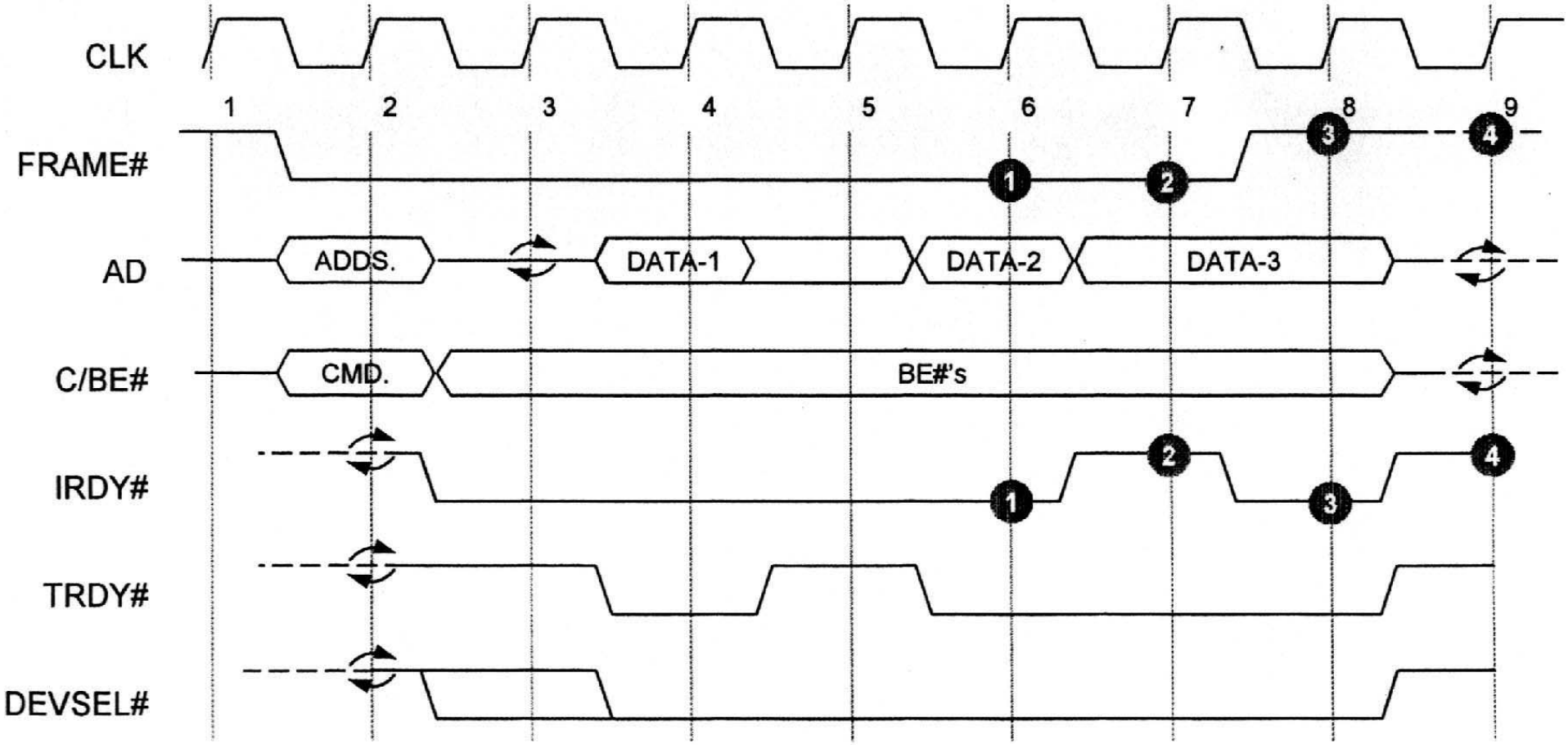
- Clock #2: Address valid and Command (e.g. Memory Read) Valid
- Clock #3: AD lines must switch (read operation!) and 1 clock waiting is enforced. This is done by the Target by not asserting TRDY# yet. The Master indicates its readiness by asserting IRDY#. The Master must, at this time, assert valid BE#'s.
- Clock #4: Target responds by DEVSEL# (could be done at clock #3, but not required), and it happens to be ready as well - TRDY# asserted. The Master reads data-1. First Data Phase is complete

Basic Read (c)



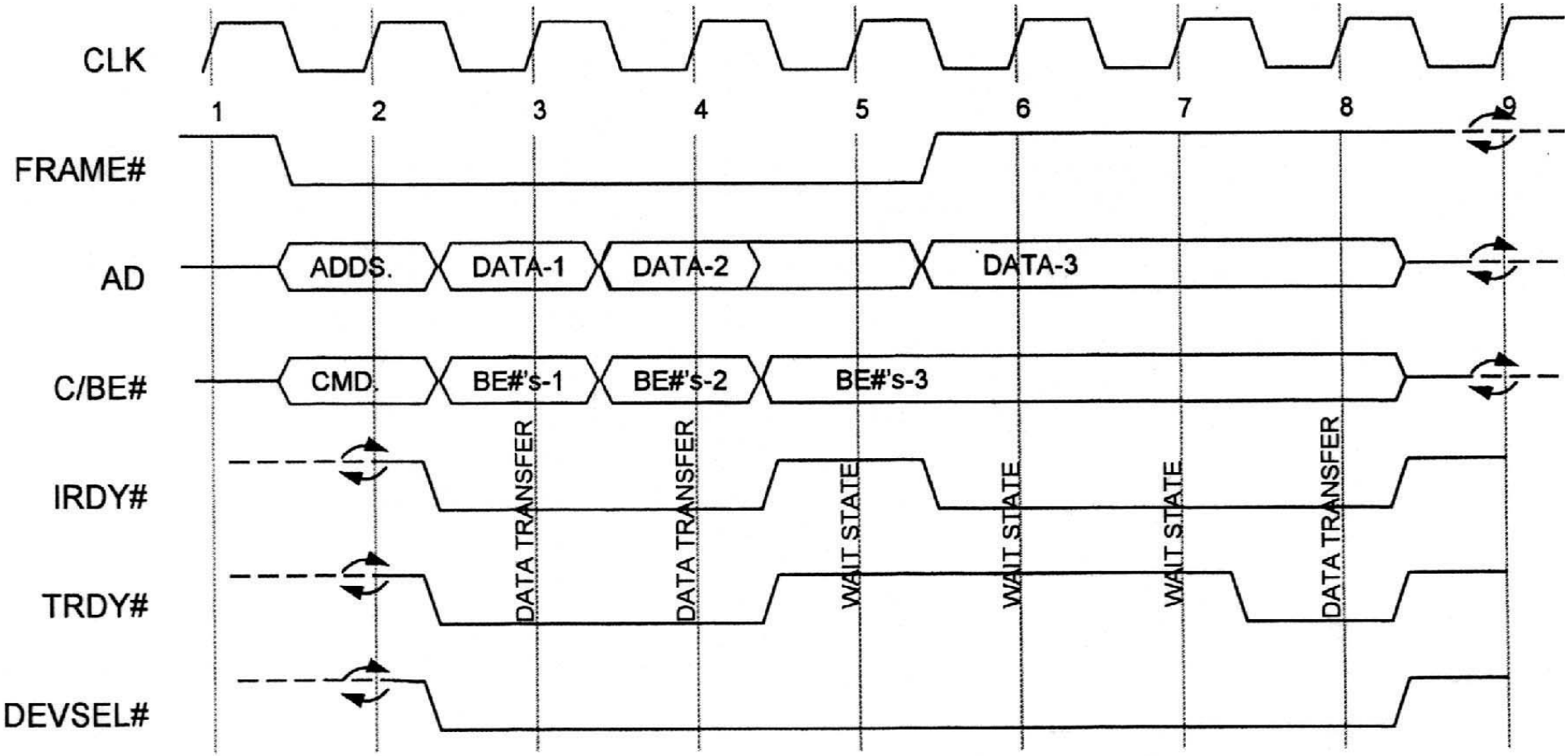
- The Master must keep BE#'s valid all the time. Since most transactions are 32-bit bursts BE#'s value will not change during the transaction although the Master may change them as long as they are valid and stable at the clock's rising edges
- Clock #5: The Target inserts a wait-state by deasserting TRDY#. Clock #6: Data-2 is read by the Master
- Clock #7: A wait state is inserted by the Master. Also, the Master knows it is the last data transfer
- Clock #8: The Master deasserts FRAME# to indicate last data transfer. This can be done only when IRDY# is asserted, otherwise it will signify "Bus Idle". Since TRDY# is also asserted, the last data transfer will take place

Points To Remember



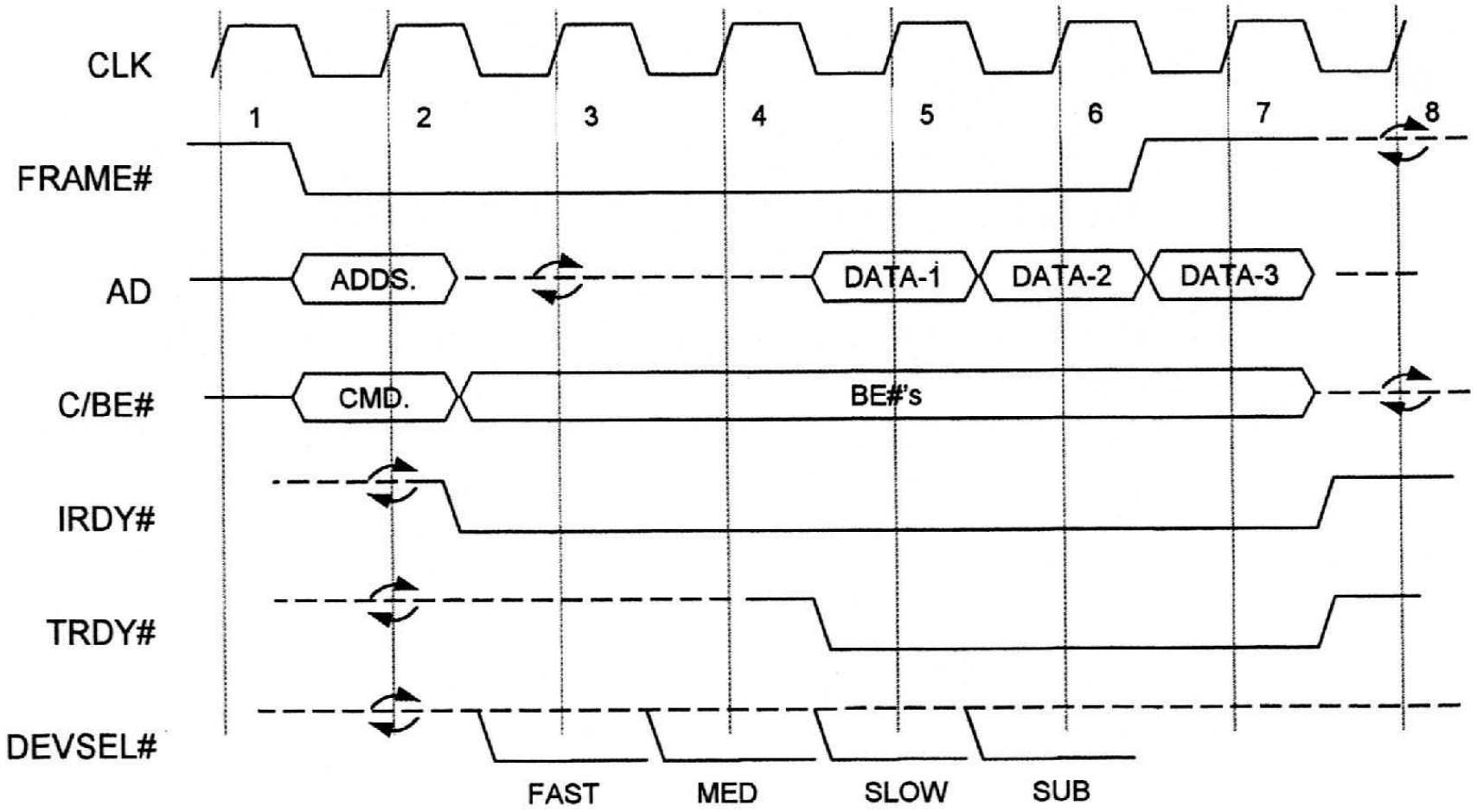
- Combinations **1**, **2**, and **3** are "Bus Busy". **4** is "Bus Idle"
- The combination IRDY# active, TRDY# active and FRAME# inactive indicate the last data transfer!
- The basic PCI transaction is a burst
- Any number of wait states, on any data phase, can be mutually generated by the master and target, using IRDY# and TRDY#

Basic Write



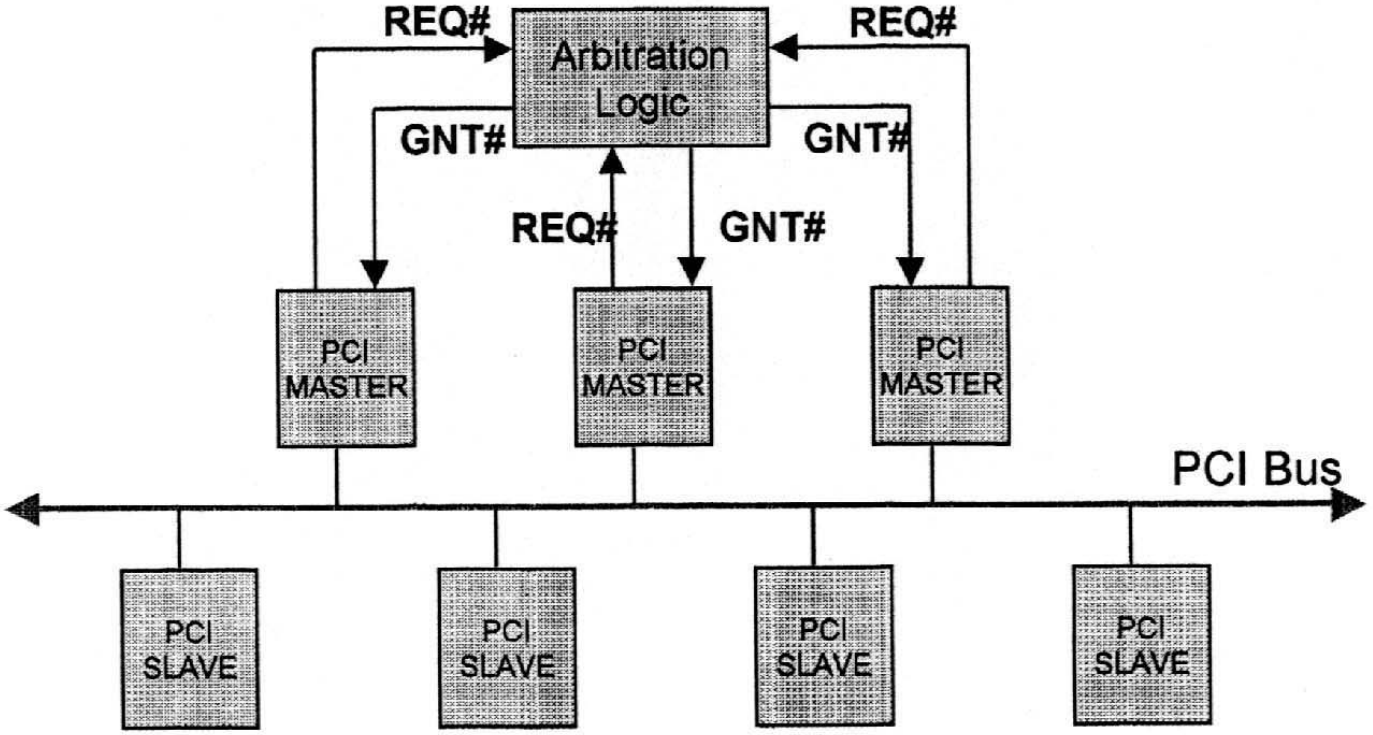
- Since the Master drives both address and later on the data, no turn-around of AD lines is required
- Clock #4: The Master knows it needs one more data transfer.
- Clock #5: The Master is not ready, therefore it **must** keep FRAME# asserted
- Clock #6: The Master asserts IRDY# and deasserts FRAME#, but the Target is not yet ready, nor at clock #7
- Clock #8: The Target is ready now and receives Data-3

Device Selection



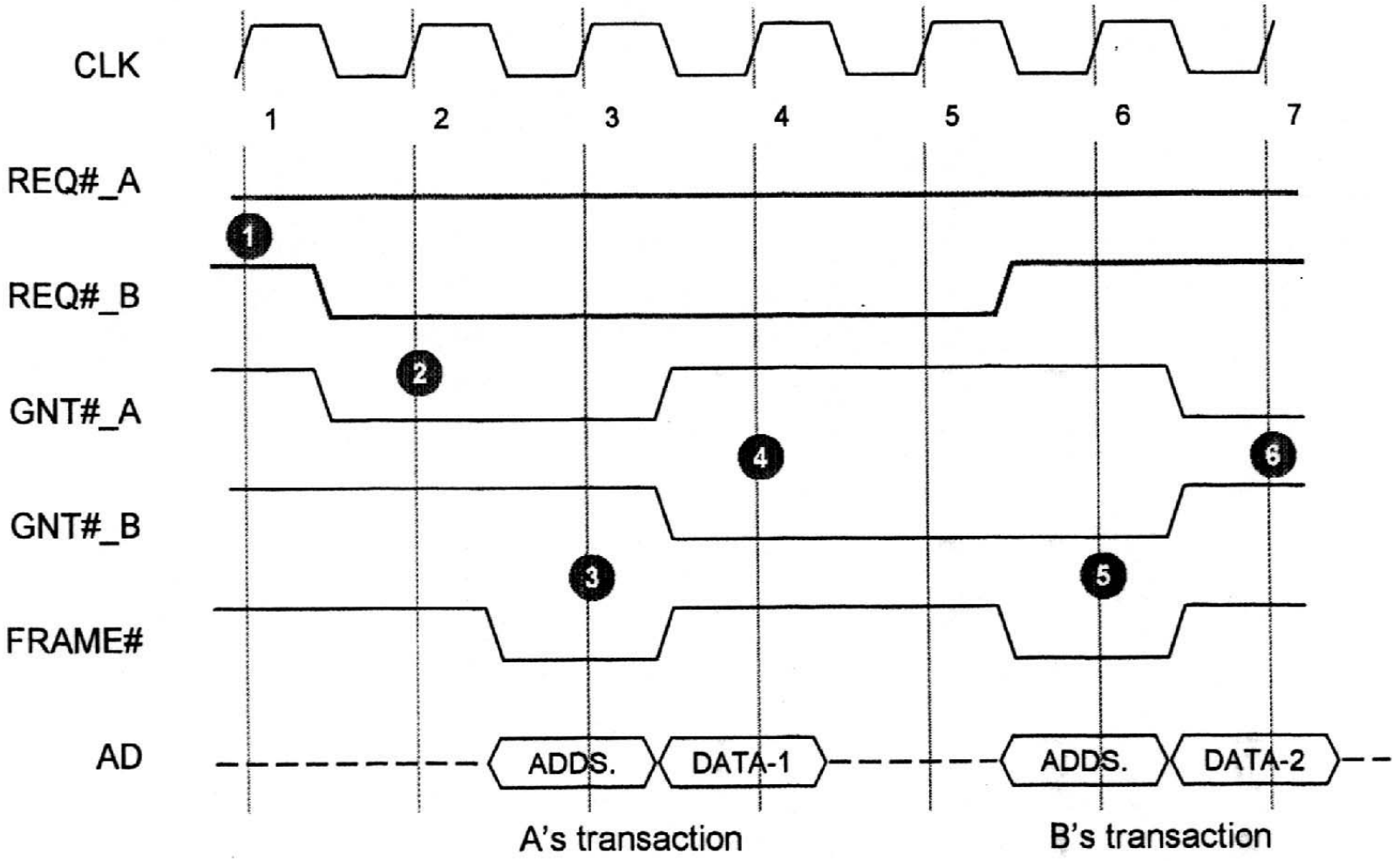
- A target is given a chance to respond 1, 2 or 3 clocks after the one at which FRAME# was asserted, according to its being "Fast", Medium" or "Slow" respectively
- A Subtractive Decode device can claim the transaction at the 4th clock when DEVSEL# line is still inactive at clock #5
- DEVSEL# must be the first response signal from the target, after complete address decoding. It must stay active as long as FRAME# is still active

Arbitration Scheme



- PCI defines REQ# and GNT# lines for each Master.
- The REQests from the Masters are arbitrated centrally, usually by the PCI "Central Resource"
- PCI does not define the Arbitration Technique (algorithm)
- The Arbitration Logic must not assert more than **one** GNT# line

Arbitration Example

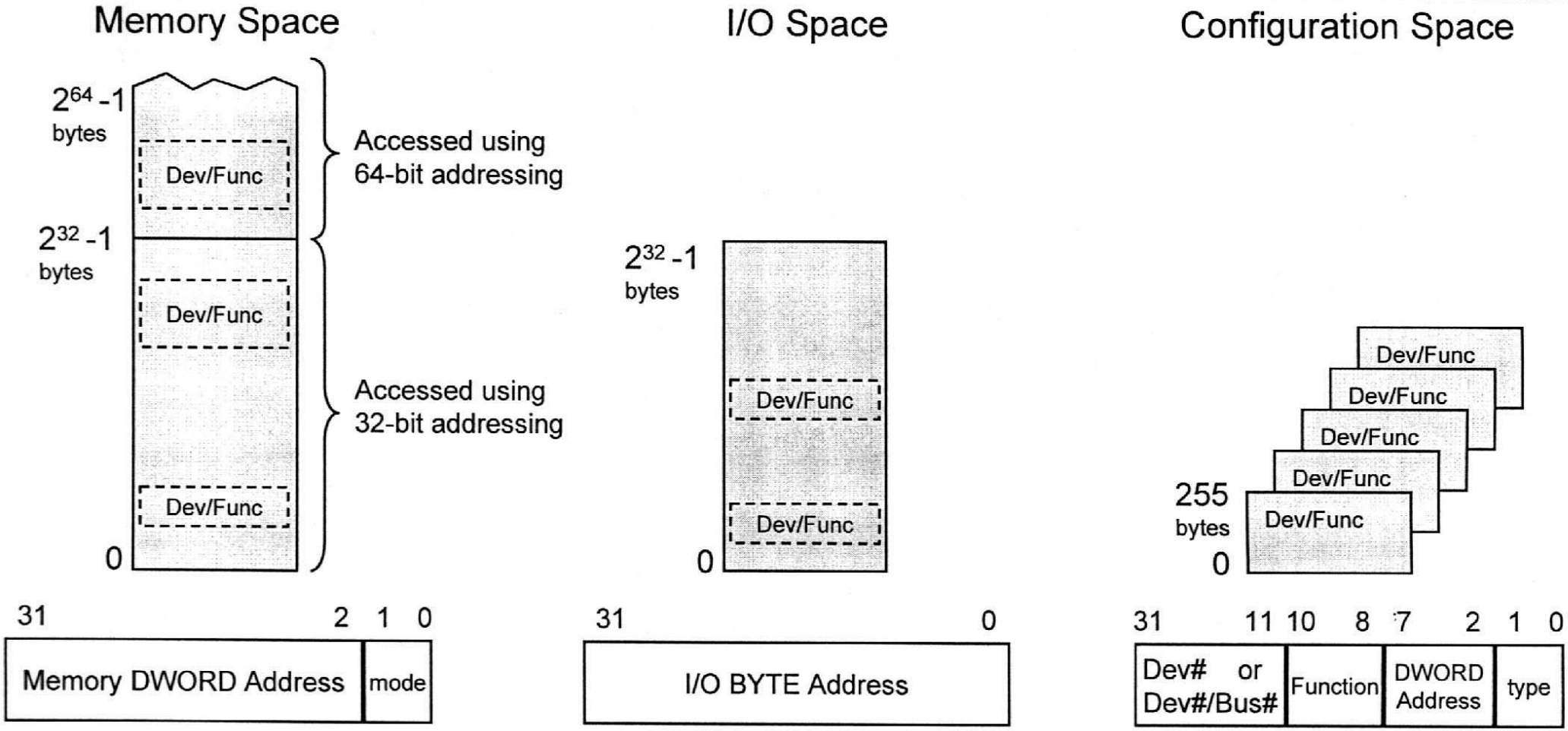


1. Agent A wants the bus (for many transactions), B doesn't for the moment. **2.** Arbiter grants the bus to A, B now wants the bus too but it is a bit too late... Arbiter records request **3.** A starts transaction and does not need the GNT# line. **4.** Arbiter removes GNT# from A and gives now GNT# to B. A will not be able to perform its next transaction at this time. **5.** B waits until bus is Idle and starts its transaction. It also de-asserts REQ#_B since it only needs to make one transaction. **6.** While B's transaction is in progress Arbiter can remove GNT#_B and assert GNT#_A since A needs the bus constantly.

Arbitration Rules

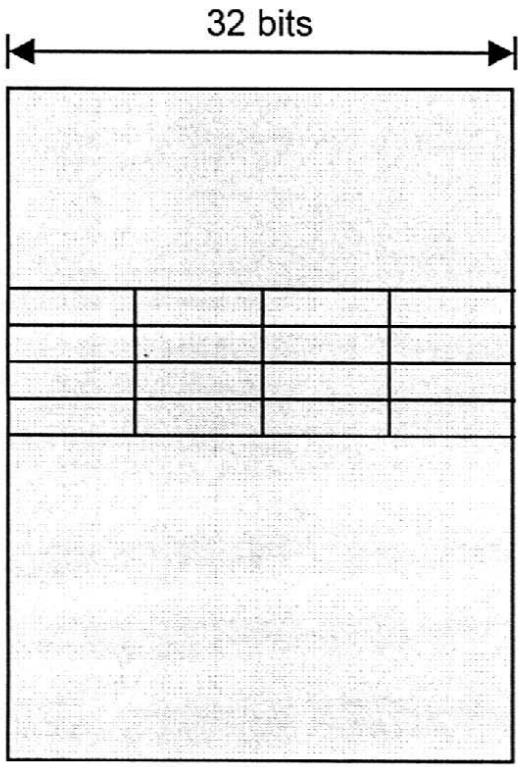
- A Master must arbitrate for each access to the bus
- When the bus is not Idle, arbitration is done simultaneously with the on-going transaction
- The Arbiter may employ any algorithm which will guarantee "Worst Case Latency" and assertion of only one GNT# line
- A Master should not assert constant REQ#
- The Arbiter may define a "Default Bus Owner" whose GNT# line will be asserted when no other Master requires the bus (the bus is said to be "Parked" at this Master)
- When a GNT# line has been asserted by the Arbiter
  - ...the Arbiter may de-assert the GNT# when the Master has already asserted FRAME#. The transaction will continue
  - ...and the bus is Idle, the arbiter must wait 1 clock after de-asserting GNT# and before asserting another GNT# line
  - ...and the bus is not Idle, the Arbiter may de-assert a GNT# line and assert another one at the same clock
- A Master at which the bus is "Parked", that needs to make one transaction only, should not assert REQ# (it already has the GNT# asserted). It only asserts FRAME#
- When the bus is Idle (no REQ#'s), the Default Bus Owner must:
  - Within 8 clocks, enable its AD, C/BE# and PAR buffers
  - Within 1 clock disable these buffers when the Arbiter de-asserts GNT# (in order to grant the bus to another Master).
- Arbitration Latency
  - 0 clocks for the Master at which the bus is parked, 2 clocks for all other Masters
  - 1 clock for all Masters when the bus is not parked

Three Address Spaces

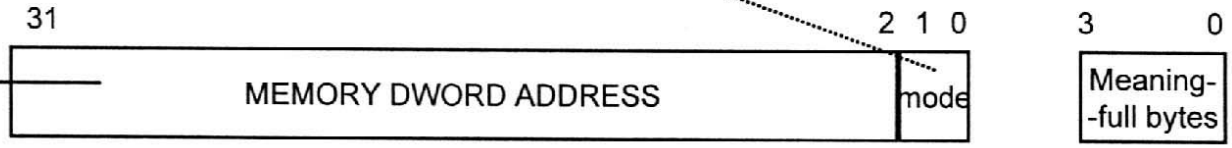


- A Device/Function mapped (visible) in the Memory Space will be selected by using a memory address that falls within its address range
- A Device/Function mapped (visible) in the I/O Space will be selected by using an I/O address that falls within its address range
- A Device/Function mapped (visible) in the Configuration Space will be selected by **its IDSEL pin**. This is required because the entire 64-DWORD address space exists in every Device/Function

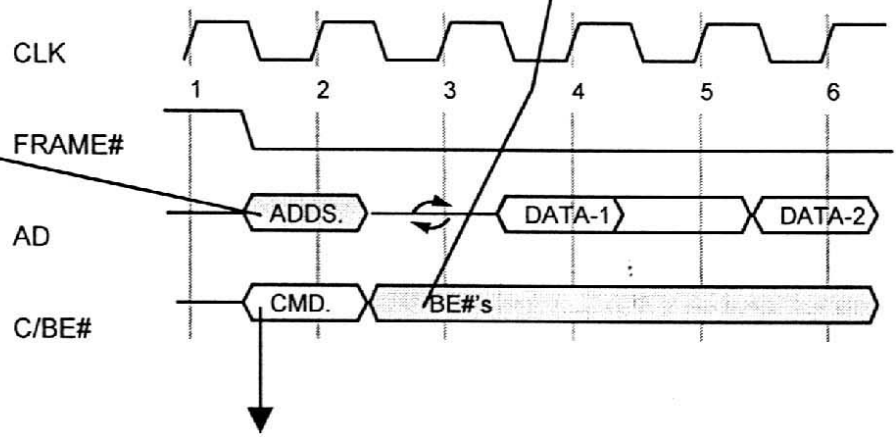
Accessing Memory Space



AD1	AD0	
0	0	Linear Incrementing
0	1	Cache Line toggle mode
1	x	Reserved



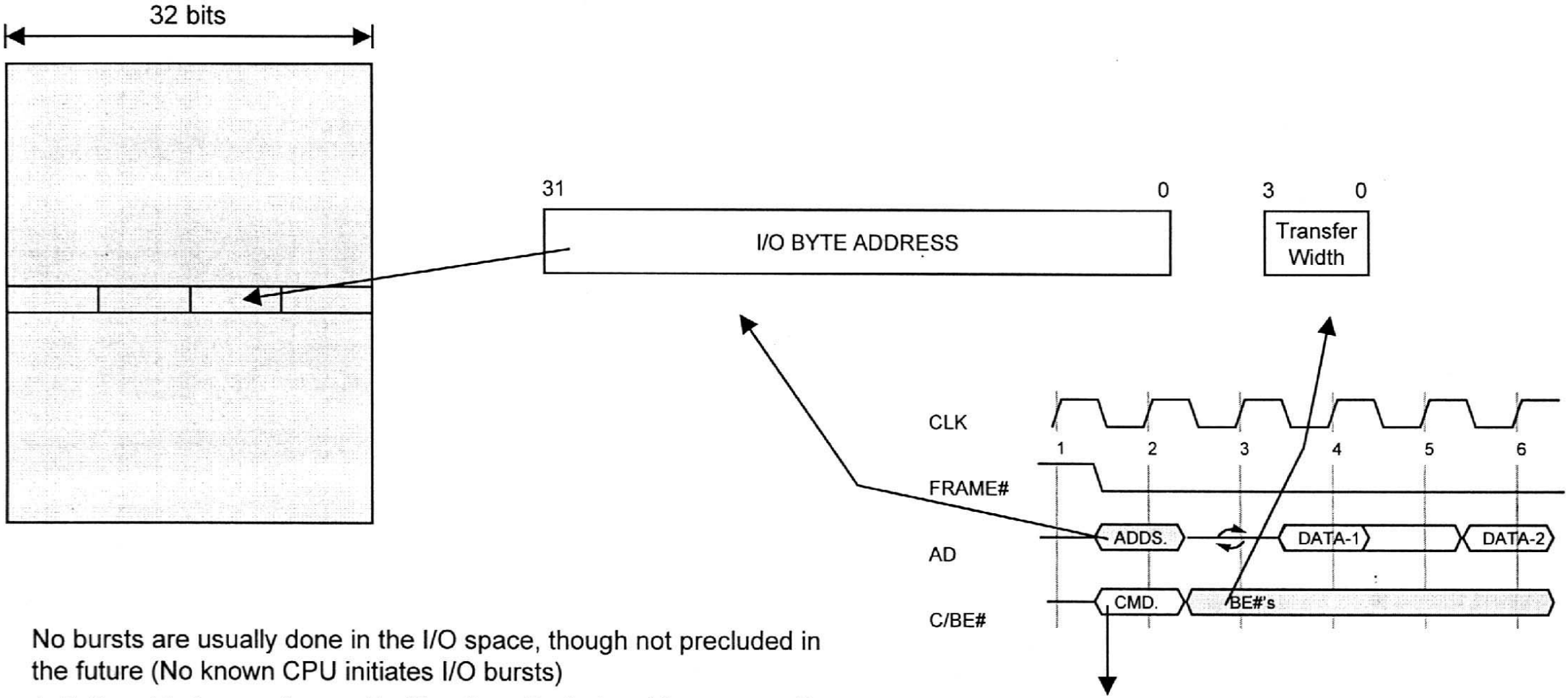
X
multiple of 4



- Reading: full DWORD-aligned double-word is read, all BE#'s are active
- Writing: DWORD address used, active BE#'s cause actual writing in the Target
- In a general burst (read or write), the mode will be 00 and the Target will increment the address from the initial address given in clock 2
- During Cache-Line fill (initiated by the Cache Controller) the mode will be 01 so that if, for instance, the Target gets an initial address x+8, and the known line size is 16 bytes, it will increment it through x+12, x, x+4 - as in the Intel i486

Memory Read
Memory Read Line
Memory Read Multiple
Memory Write
Memory Write & Invalidate

Accessing I/O Space



- No bursts are usually done in the I/O space, though not precluded in the future (No known CPU initiates I/O bursts)
- 1, 2, 3 or 4 bytes can be read/written from the byte address according to the following table:

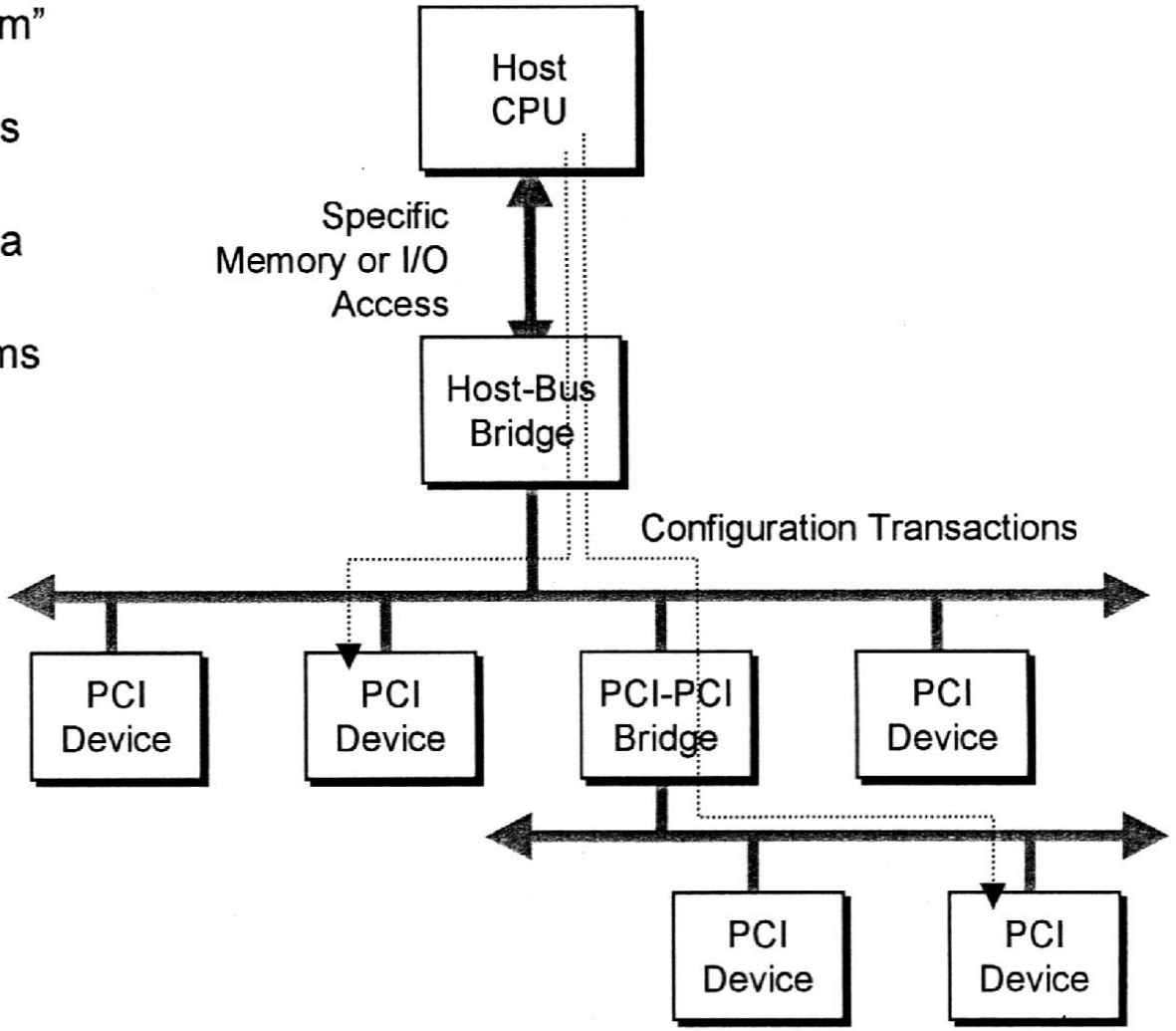
AD1	AD0	C/BE3#	C/BE2#	C/BE1#	C/BE0#
0	0	X	X	X	0
0	1	X	X	0	1
1	0	X	0	1	1
1	1	0	1	1	1

x = 0 or 1

I/O Read
I/O Write

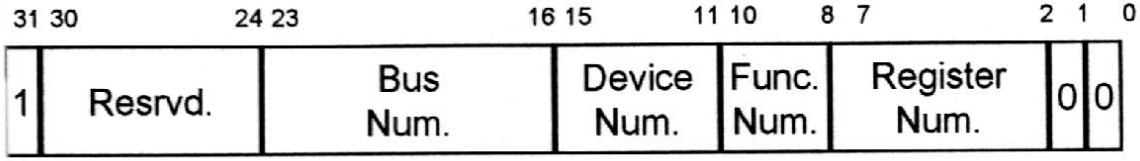
Generating Configuration Space Accesses

- Host CPU's (usually) do not have a "Configuration Address" Space, just Memory Space. Some of them have I/O space as well
- The Host Bus Bridge should have a "mechanism" by which the host CPU could initiate a Configuration Space Transaction. The Host Bus Bridge will perform the actual transaction
- The PCI spec. 2.1 defines NO mechanism for a platform other than a PC-AT
- The PCI spec. 2.1 defines two such mechanisms for a PC-AT platform
- These mechanisms also provide PCI "Special Cycles"

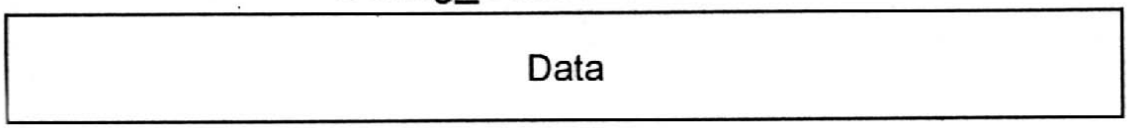


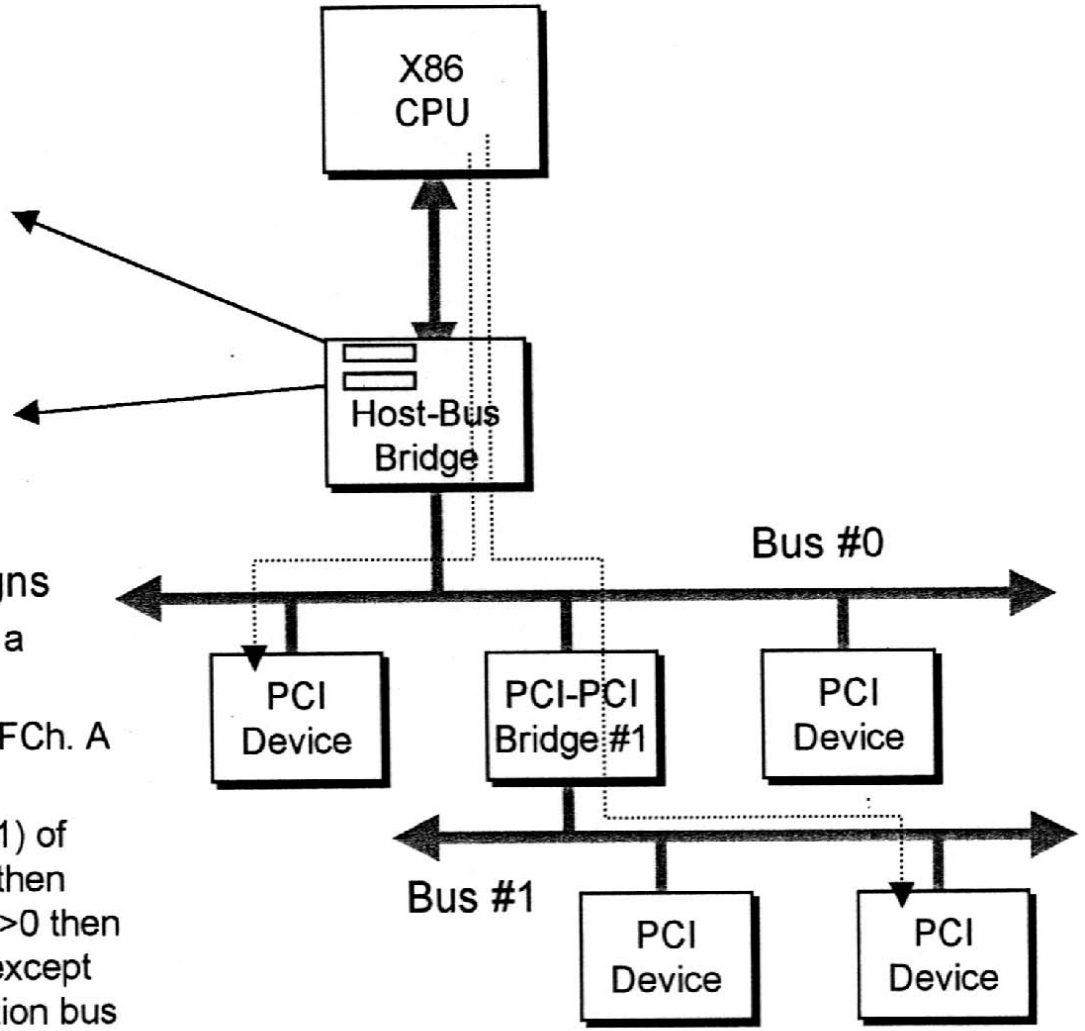
Accessing Configuration Space *Mechanism #1*

Config_Address I/O Adds. = CF8h



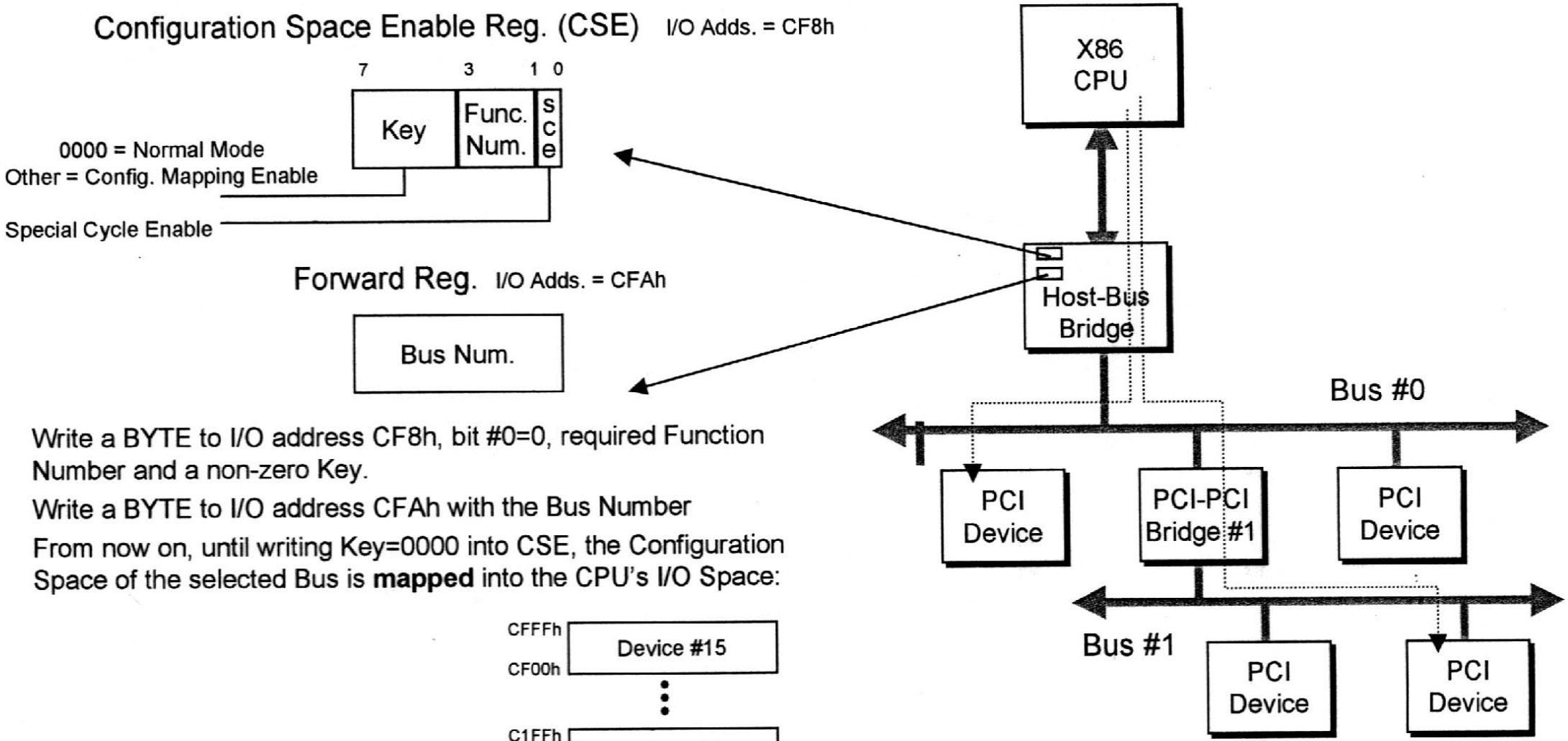
Config_Data I/O Adds. = CFCh





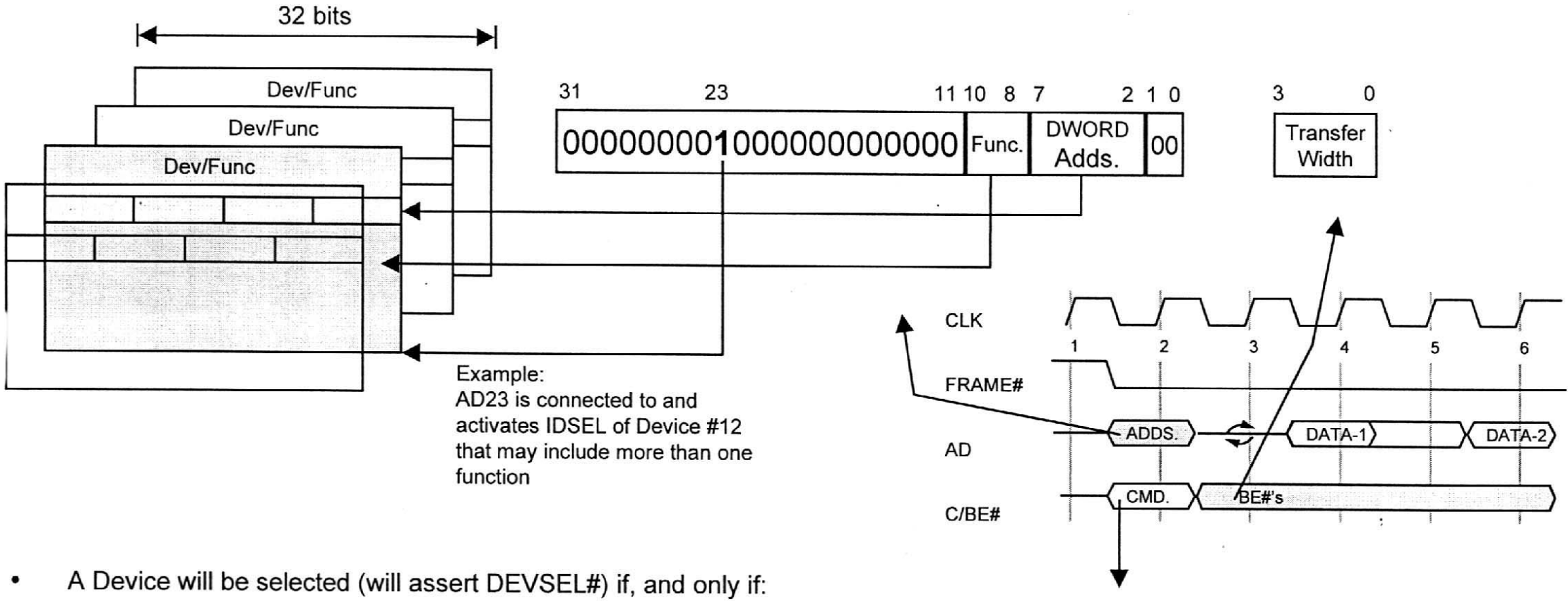
- This is the mechanism to be implemented in new designs
- Write a DWORD to I/O address CF8h with bit 31 = 1, otherwise a normal I/O transaction will take place
- Read or Write BYTE, WORD or DOWRD from/to I/O address CFCh. A Configuration Space Transaction will start
- The Bus Number field will tell the bridge what format type (0 or 1) of Configuration Address should be generated. If Bus Number =0 then Type0 Configuration Address will be generated. If Bus Number >0 then Type 1 will be generated. This type is ignored by PCI devices, except PCI-PCI bridges. The PCI-PCI bridge behind which the destination bus resides, will claim this transaction
- At the destination bridge (can be the Host Bridge too), the Device Number field will be transformed to a "1" in that part of the Configuration Address used to drive the IDSEL line. Bits 10:0 will not change

Accessing Configuration Space *Mechanism #2*



- Write a BYTE to I/O address CF8h, bit #0=0, required Function Number and a non-zero Key.
- Write a BYTE to I/O address CFAh with the Bus Number
- From now on, until writing Key=0000 into CSE, the Configuration Space of the selected Bus is **mapped** into the CPU's I/O Space:

Accessing Configuration Space



- A Device will be selected (will assert DEVSEL#) if, and only if:
  Its IDSEL pin is asserted (active high) and
  AD1::AD0 are 00 and
  The command is Configuration Read or Write
- C/BE#'s will determine which (contiguous) bytes will be accessed
- Example: In the Intel's 82439HX Dev #0 decodes to AD11, Dev #1 to AD12, Dev #20 to AD31. Dev #0 is the Host Bridge itself so that AD11 is never asserted