



PCI Tutorial

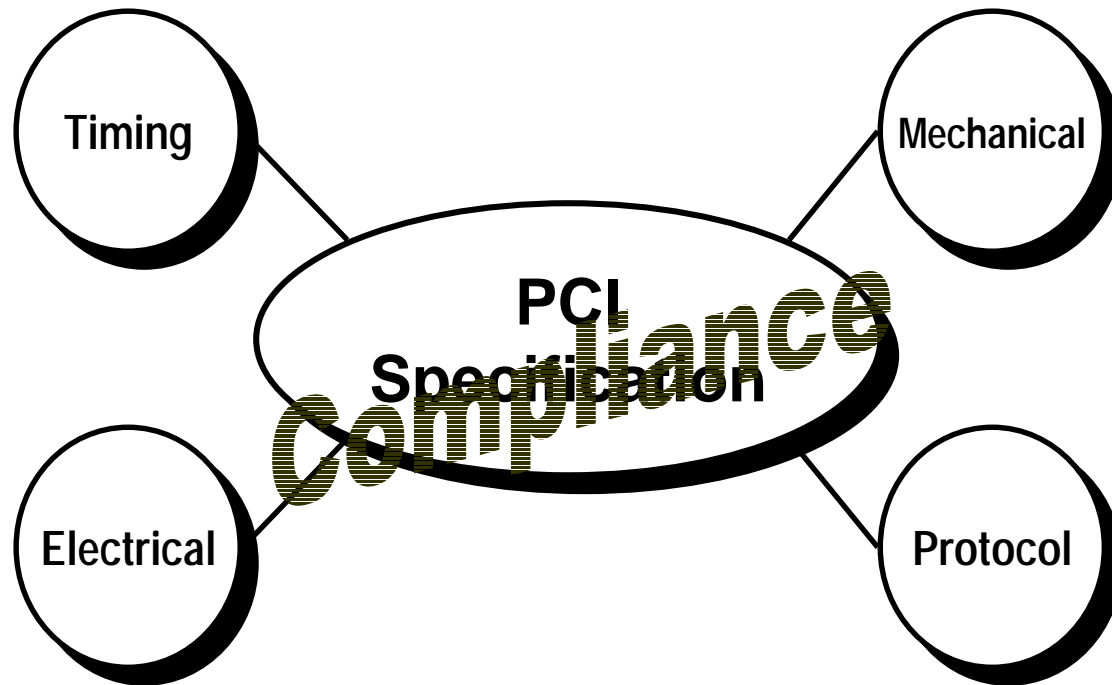




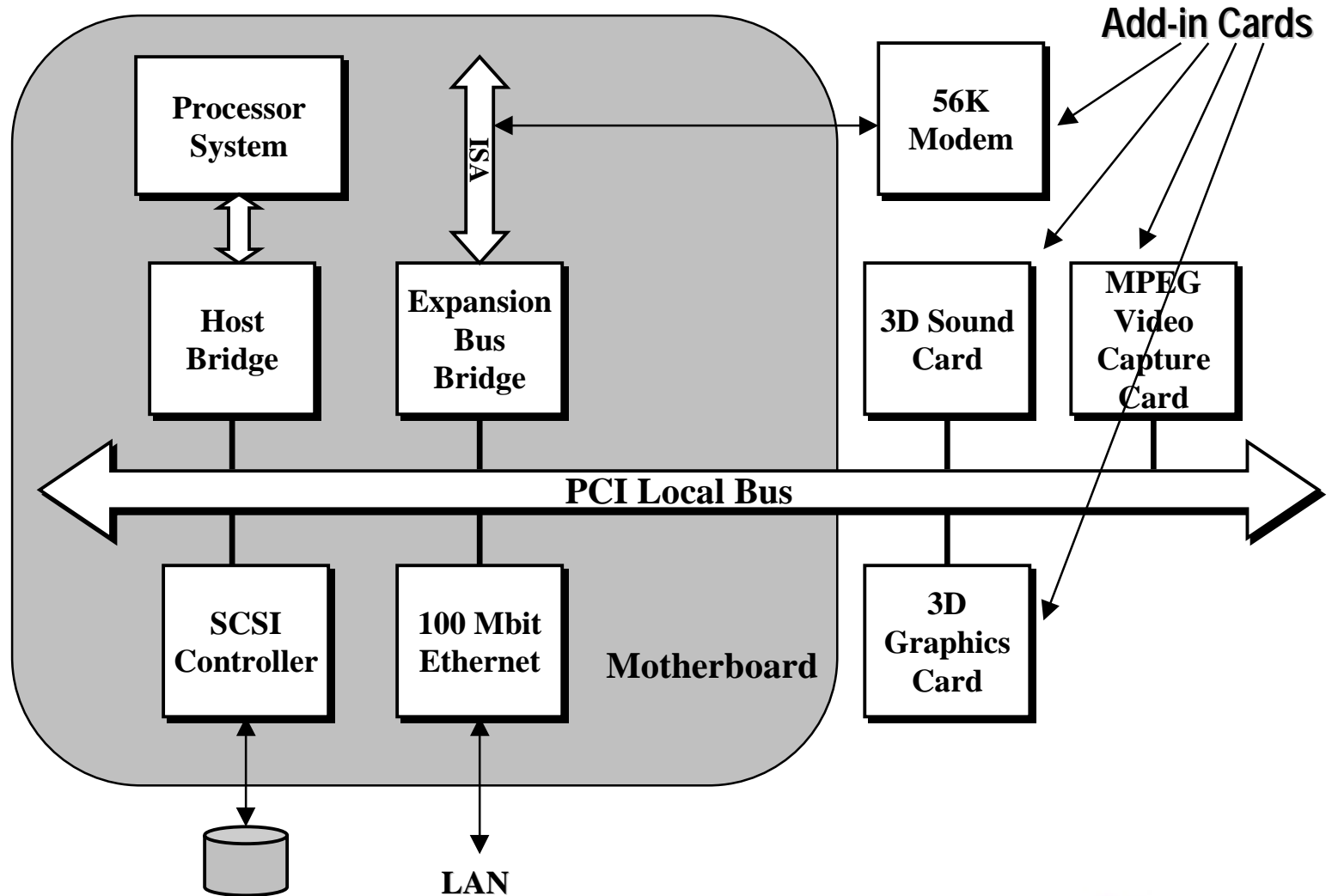
PCI Fundamentals and Concepts

Overview of the PCI Specification

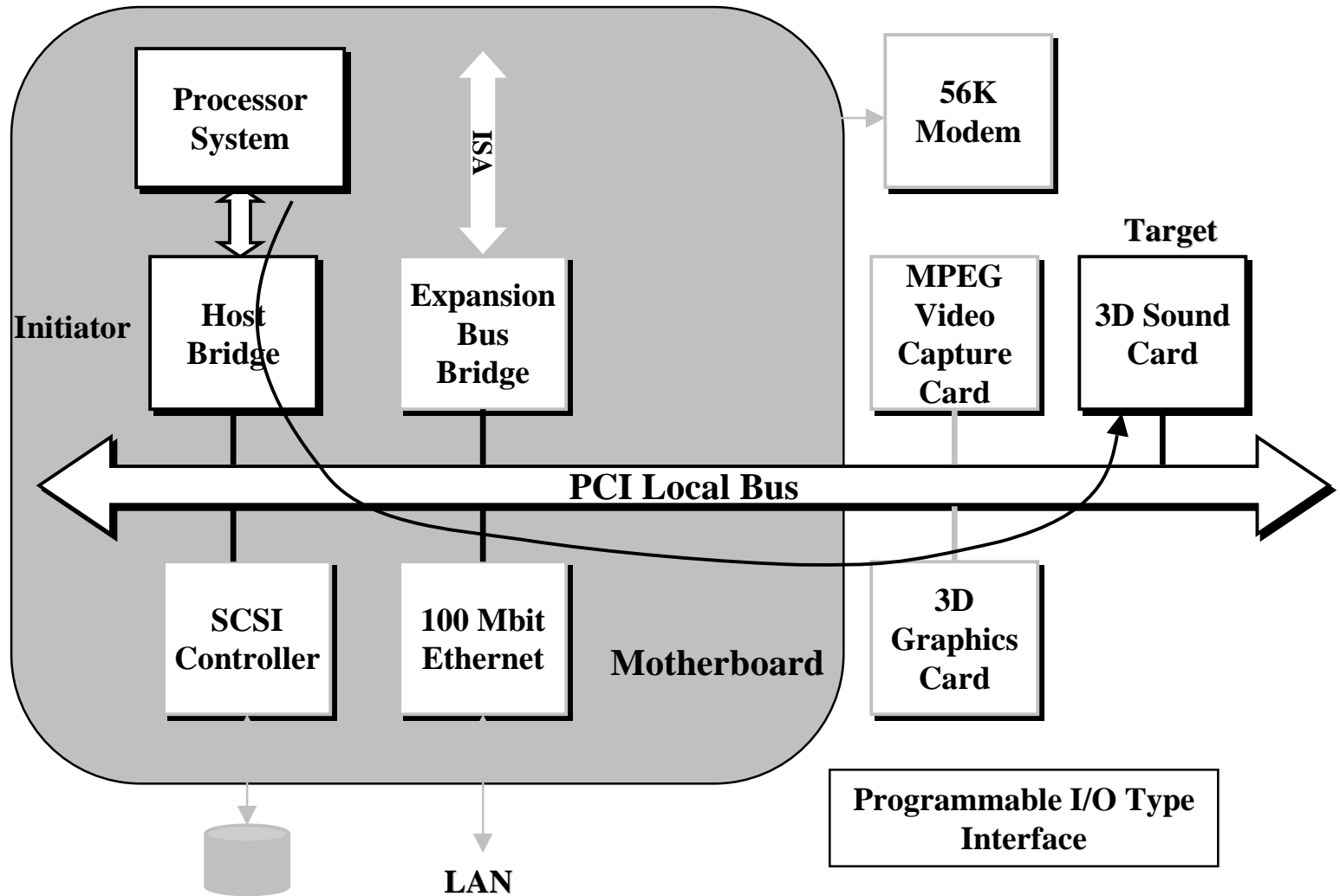
- ◆ *The PCI Local Bus Specification* covers many different requirements for PCI compliance



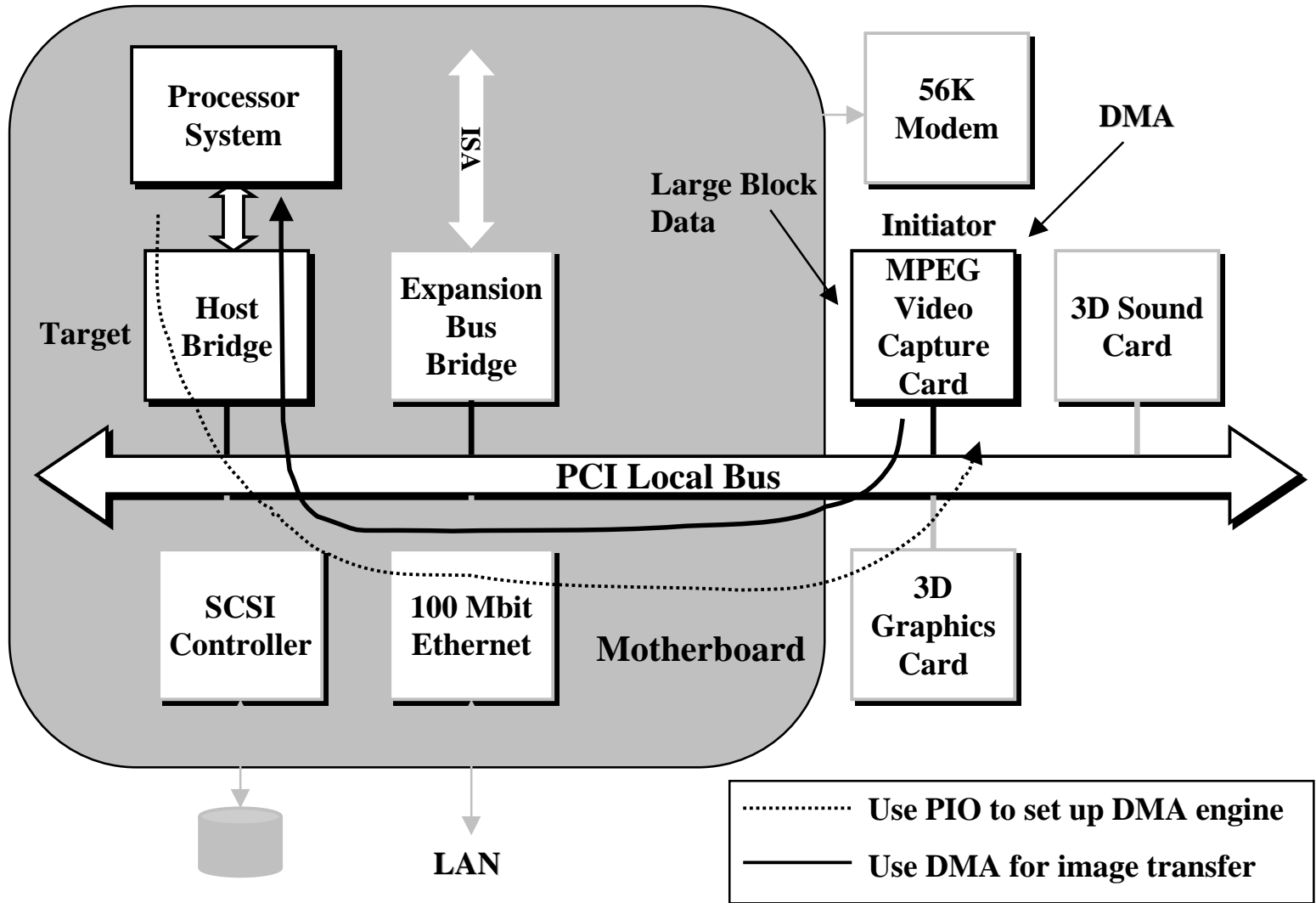
Basic Bus Architecture



Sample Transaction - PIO



Sample Transaction - DMA



Key Terms

- ◆ **Initiator**
 - *Or Master*
 - *Owns the bus and initiates the data transfer*
 - *Every Initiator must also be a Target*
- ◆ **Target**
 - *Or Slave*
 - *Target of the data transfer (read or write)*
- ◆ **Agent**
 - *Any initiator/target or target on the PCI bus*

PCI Signals

Xilinx PCI

Clock and Reset

◆ CLK

- *PCI input clock*
- *All signals sampled on rising edge*
- *33MHz is really 33.33333MHz (30ns clk. period)*
- *The clock is allowed to vary from 0 to 33 MHz*
 - The frequency can change "on the fly"
 - Because of this, no PLLs are allowed

◆ RST#

- *Asynchronous reset*
- *PCI device must tri-state all I/Os during reset*

Transaction Control Initiator Signals

- ◆ **FRAME# – I/O**
 - *Signals the start and end of a transaction*
- ◆ **IRDY# – I/O**
 - *“I-Ready”*
 - *Assertion by initiator indicates that it is ready to send or receive data*

Transaction Control Target Signals

- ◆ **DEVSEL# – I/O**
 - *Device select*
 - *Part of PCI's distributed address decoding*
 - Each target is responsible for decoding the address associated with each transaction
 - When a target recognizes its address, it asserts DEVSEL# to claim the corresponding transaction

Transaction Control Target Signals

- ◆ **TRDY# – I/O**
 - *“T-Ready”*
 - *When the target asserts this signal, it tells the initiator that it is ready to send or receive data*
- ◆ **STOP# – I/O**
 - *Used by target to indicate that it needs to terminate the transaction*

Transaction Control Configuration Signals

- ◆ Uses the same signals as the target, plus . . .
- ◆ IDSEL – I
 - *“ID-SEL”*
 - *Individual device select for configuration – one unique IDSEL line per agent*
 - *Solves the “chicken-and-egg” problem*
 - Allows the system host to configure agents before these agents know the PCI addresses to which they must respond

Address and Data Signals

- ◆ **AD[31:0] – I/O**
 - *32-bit address/data bus*
 - *PCI is little endian (lowest numeric index is LSB)*
- ◆ **C/BE#[3:0] – I/O**
 - *4-bit command/byte enable bus*
 - *Defines the PCI command during address phase*
 - *Indicates byte enable during data phases*
 - Each bit corresponds to a “byte-lane” in AD[31:0] – for example, C/BE#[0] is the byte enable for AD[7:0]

Address and Data Signals

- ◆ **PAR – I/O**
 - *Parity bit*
 - *Used to verify correct transmittal of address/data and command/byte-enable*
 - *The XOR of AD[31:0], C/BE#[3:0], and PAR should return zero (even parity)*
 - In other words, the number of 1's across these 37 signals should be even

Arbitration Signals

- ◆ For initiators only!
- ◆ REQ# – 0
 - Asserted by initiator to **request** bus ownership
 - Point-to-point connection to arbiter – each initiator has its own REQ# line
- ◆ GNT# – 1
 - Asserted by system arbiter to **grant** bus ownership to the initiator
 - Point-to-point connection from arbiter – each initiator has its own GNT# line

Error Signals

- ◆ **PERR# – I/O**

- *Indicates that a data **parity error** has occurred*
- *An agent that can report parity errors can have its PERR# turned off during PCI configuration*

- ◆ **SERR# – I/O**

- *Indicates a serious **system error** has occurred*
 - Example: Address parity error
- *May invoke NMI (non-maskable interrupt, i.e., a restart) in some systems*

Basic Bus Operations

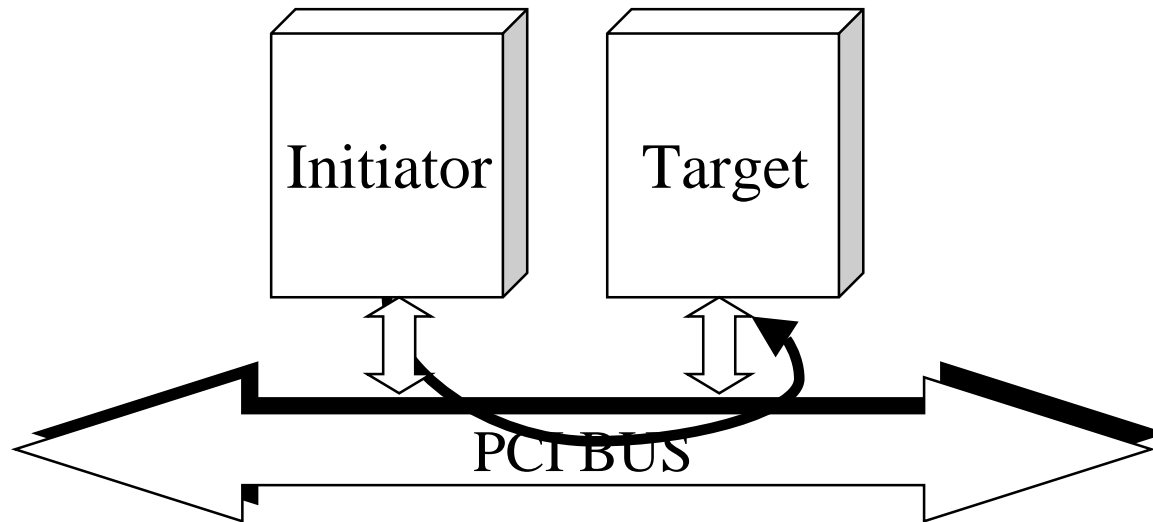
Xilinx PCI

Terms

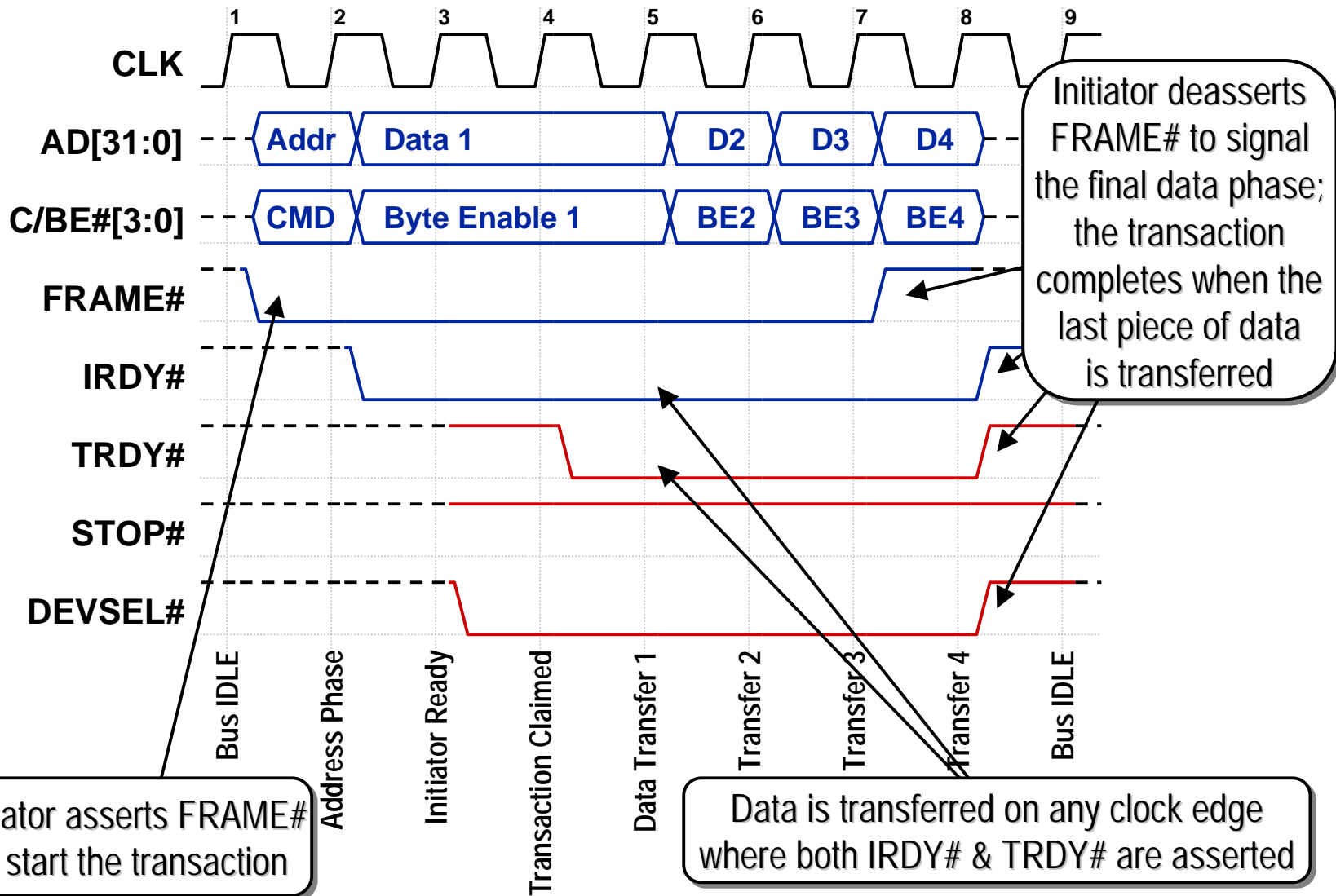
- ◆ **Doubleword**
 - *32 bits, most often known as a "DWORD"*
- ◆ **Quadword**
 - *64 bits, sometimes known as a "QWORD"*
- ◆ **Burst transaction**
 - *Any transaction consisting of more than one data phase*
- ◆ **Idle state (no bus activity)**
 - *Indicated by FRAME# and IRDY# deasserted*

Example #1 – Basic Write

- ◆ A four-DWORD burst from an initiator to a target



Basic Write Transaction



Write Example – Things to Note

- ◆ The initiator has a phase profile of 3-1-1-1
 - *First data can be transferred in three clock cycles (idle + address + data = "3")*
 - *The 2nd, 3rd, and last data are transferred one cycle each ("1-1-1")*

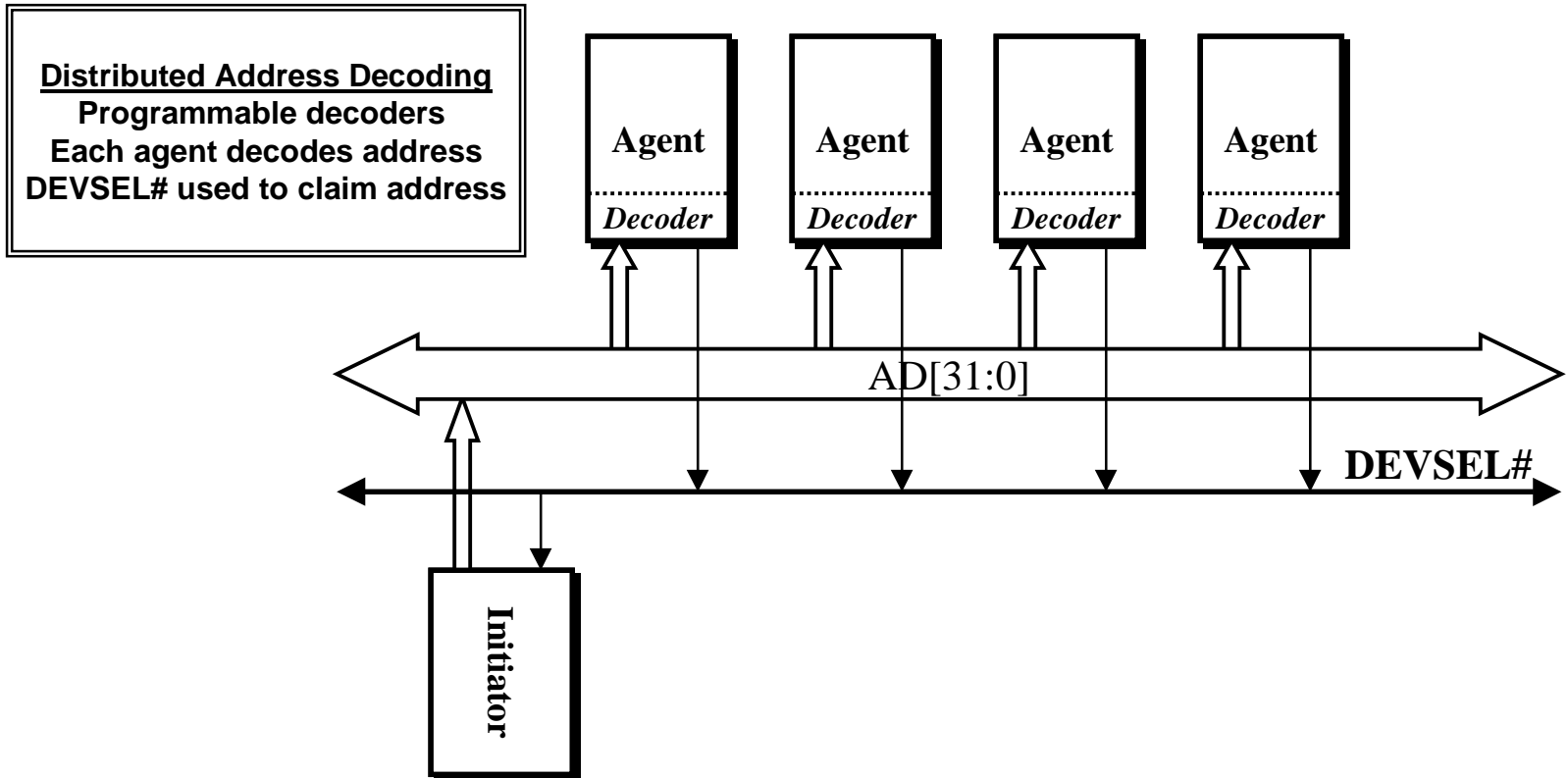
Write Example – Things to Note

- ◆ **The target profile is 5-1-1-1**
 - *Medium decode – DEVSEL# asserted on 2nd clock after FRAME#*
 - *One clock period of latency (or wait state) in the beginning of the transfer*
 - DEVSEL# asserted on clock 3, but TRDY# not asserted until clock 4
 - *Ideal target write is 3-1-1-1*
- ◆ **Total of 4 data phases, but required 8 clocks**
 - *Only 50% efficiency*

Target Address Decoding

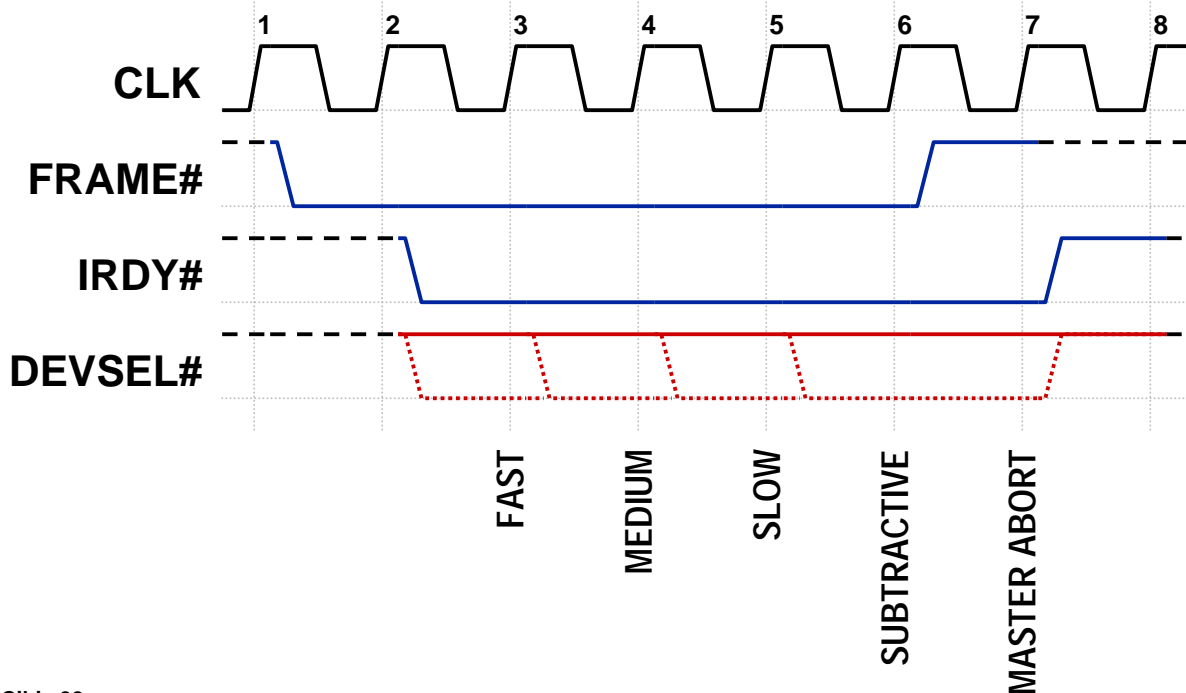
- ◆ **PCI uses distributed address decoding**
 - *A transaction begins over the PCI bus*
 - *Each potential target on the bus decodes the transaction's PCI address to determine whether it belongs to that target's assigned address space*
 - One target may be assigned a larger address space than another, and would thus respond to more addresses
 - *The target that owns the PCI address then claims the transaction by asserting DEVSEL#*

Distributed Address Decoding



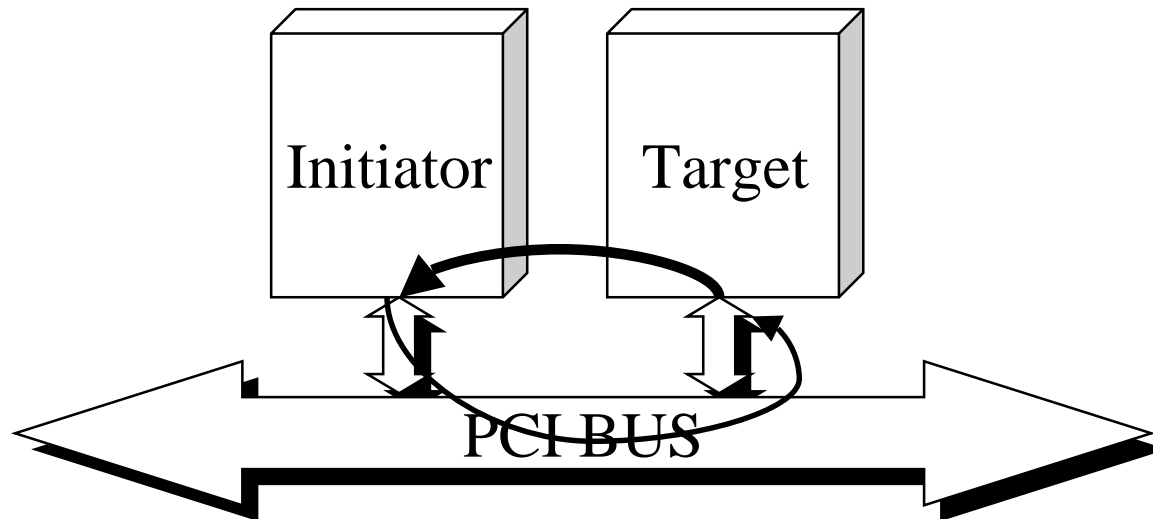
Target Decode

- ◆ Address decoders come in different speeds
- ◆ If a transaction goes unclaimed (nobody asserts DEVSEL#), "Master Abort" occurs



Example #2 - Target Read

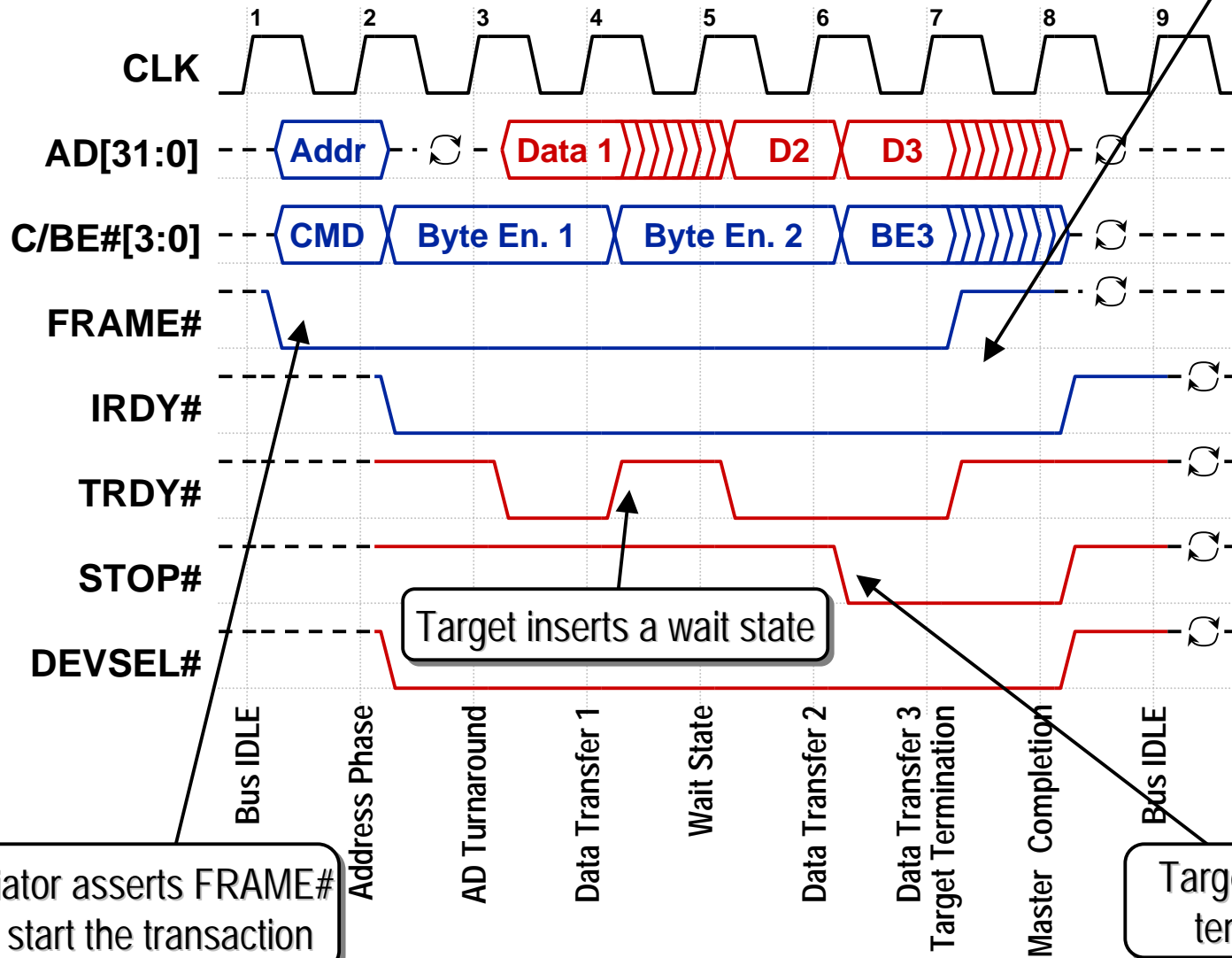
- ◆ A four-DWORD burst read from a target by an initiator



More Terms

- ◆ **Turnaround cycle**
 - *“Dead” bus cycle to prevent bus contention*
- ◆ **Wait state**
 - *A bus cycle where it is possible to transfer data, but no data transfer occurs*
 - *Target deasserts TRDY# to signal it is not ready*
 - *Initiator deasserts IRDY# to signal it is not ready*
- ◆ **Target termination**
 - *Target asserts STOP# to indicate that it needs to terminate the current transaction*

Target Read Transaction



Target Read – Things to Note

- ◆ Wait states may be inserted dynamically by the initiator or target by deasserting IRDY# or TRDY#
- ◆ Either agent may signal the end of a transaction
 - *The target signals termination by asserting STOP#*
 - *The initiator signals completion by deasserting FRAME#*

Zero and One Wait State

- ◆ These terms are used in popular PCI parlance* to describe how an agent signals its $xRDY\#$ signal during each data phase

**Although the PCI spec uses the term "wait state," it does not use terms such as "zero-wait-state agent" and "one-wait-state device."*

- ◆ A one-wait-state agent inserts a wait state at the beginning of each data phase
 - *This is done if an agent – built in older, slower silicon – needs to pipeline critical paths internally*
 - *Reduces bandwidth by 50%*

Zero and One Wait State

- ◆ The need to insert a wait state is typically an issue only when the agent is sourcing data (initiator write or target read)
 - *This is because such an agent would have to sample its counterpart's $xRDY\#$ signal to see if that agent accepted data, then fan out to 36 or more clock enables (for $AD[31:0]$ and possibly $C/BE\#[3:0]$) to drive the next piece of data onto the PCI bus . . . all within 11 ns!*
 - And even that 11 ns would be eaten up by a chip's internal clock-distribution delay

Types of Target Termination

- ◆ Target Retry
"I'm not ready, try again later."
- ◆ Target Disconnect with Data
"I couldn't eat another bite . . . OK, just one more."
- ◆ Target Disconnect Without Data
"I couldn't eat another bite . . . and I'm not kidding!"
- ◆ Target Abort
"Major snafu alert!"

PCI Addressing and Bus Commands

Xilinx PCI

PCI Address Space

- ◆ A PCI target can implement up to three different types of address spaces
 - *Configuration space*
 - Stores basic information about the device
 - Allows the central resource or O/S to program a device with operational settings
 - *I/O space*
 - Used mainly with PC peripherals and not much else
 - *Memory space*
 - Used for just about everything else

Types of PCI Address Space

- ◆ **Configuration space**

- *Contains basic device information, e.g., vendor or class of device*
- *Also permits Plug-N-Play*
 - Base address registers allow an agent to be mapped dynamically into memory or I/O space
 - A programmable interrupt-line setting allows a software driver to program a PC card with an IRQ upon power-up (without jumpers!)

Types of PCI Address Space

- ◆ Configuration space (cont'd)

- *Contains 256 bytes*

- The first 64 bytes (00h – 3Fh) make up the standard configuration header, predefined by the PCI spec
- The remaining 192 bytes (40h – FFh) represent user-definable configuration space
 - *This region may store, for example, information specific to a PC card for use by its accompanying software driver*

Types of PCI Address Space

- ◆ I/O space

- *This space is where basic PC peripherals (keyboard, serial port, etc.) are mapped*
- *The PCI spec allows an agent to request 4 bytes to 2GB of I/O space*
 - For x86 systems, the maximum is 256 bytes because of legacy ISA issues

Types of PCI Address Space

- ◆ **Memory space**

- *This space is used by most everything else – it's the general-purpose address space*
 - The PCI spec recommends that a device use memory space, even if it is a peripheral
- *An agent can request between 16 bytes and 2GB of memory space*
 - The PCI spec recommends that an agent use at least 4kB of memory space, to reduce the width of the agent's address decoder

PCI Commands

- ◆ PCI allows the use of up to 16 different 4-bit commands
 - *Configuration commands*
 - *Memory commands*
 - *I/O commands*
 - *Special-purpose commands*
- ◆ A command is presented on the C/BE# bus by the initiator during an address phase (a transaction's first assertion of FRAME#)

PCI Commands

| C/BE# | Command | |
|------------|-----------------------|-----------------------------|
| 0000 | Interrupt Acknowledge | Memory |
| 0001 | Special Cycle | Memory |
| 0010 | I/O Read | I/O |
| 0011 | I/O Write | I/O |
| 0100 | Reserved | Reserved |
| 0101 | Reserved | Reserved |
| 0110 | Memory Read | Memory |
| 0111 | Memory Write | Memory |
| 1000 | Reserved | Reserved |
| 1001 | Reserved | Reserved |
| With IDSEL | 1010 | Configuration Read |
| With IDSEL | 1011 | Configuration Write |
| | 1100 | Memory Read Multiple |
| | 1101 | Dual Address Cycle |
| | 1110 | Memory Read Line |
| | 1111 | Memory Write and Invalidate |

PCI Configuration

Xilinx PCI

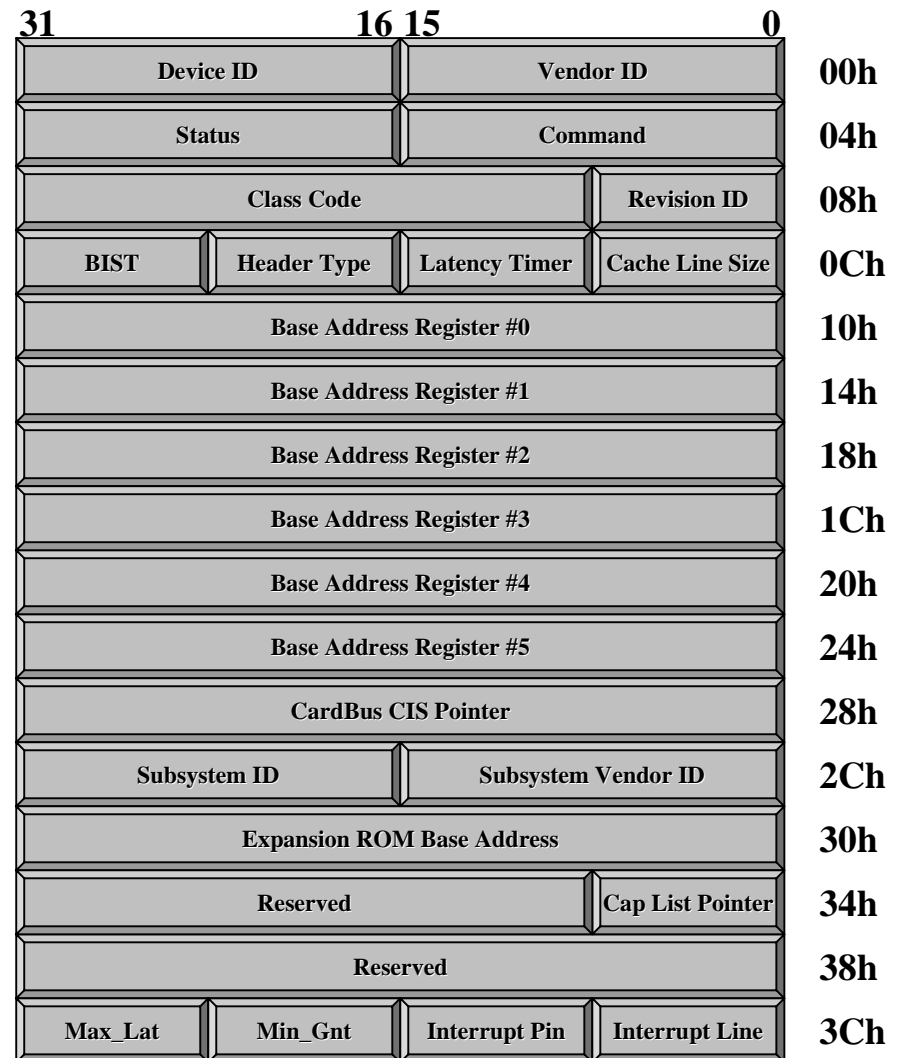
The Plug-and-Play Concept

◆ Plug-and-Play (PNP)

- *Allows add-in cards to be plugged into any slot without changing jumpers or switches*
 - Address mapping, IRQs, COM ports, etc., are assigned dynamically at system start-up
- *For PNP to work, add-in cards must contain basic information for the BIOS and/or O/S, e.g.:*
 - Type of card and device
 - Memory-space requirements
 - Interrupt requirements

The Plug-and-Play Concept

- ◆ To make PNP possible in PCI, each PCI device maintains a 256-byte configuration space
 - *The first 64 bytes (shown here) are predefined in the PCI spec and contain standard information*
 - *The upper 192 bytes may be used to store device-specific information*



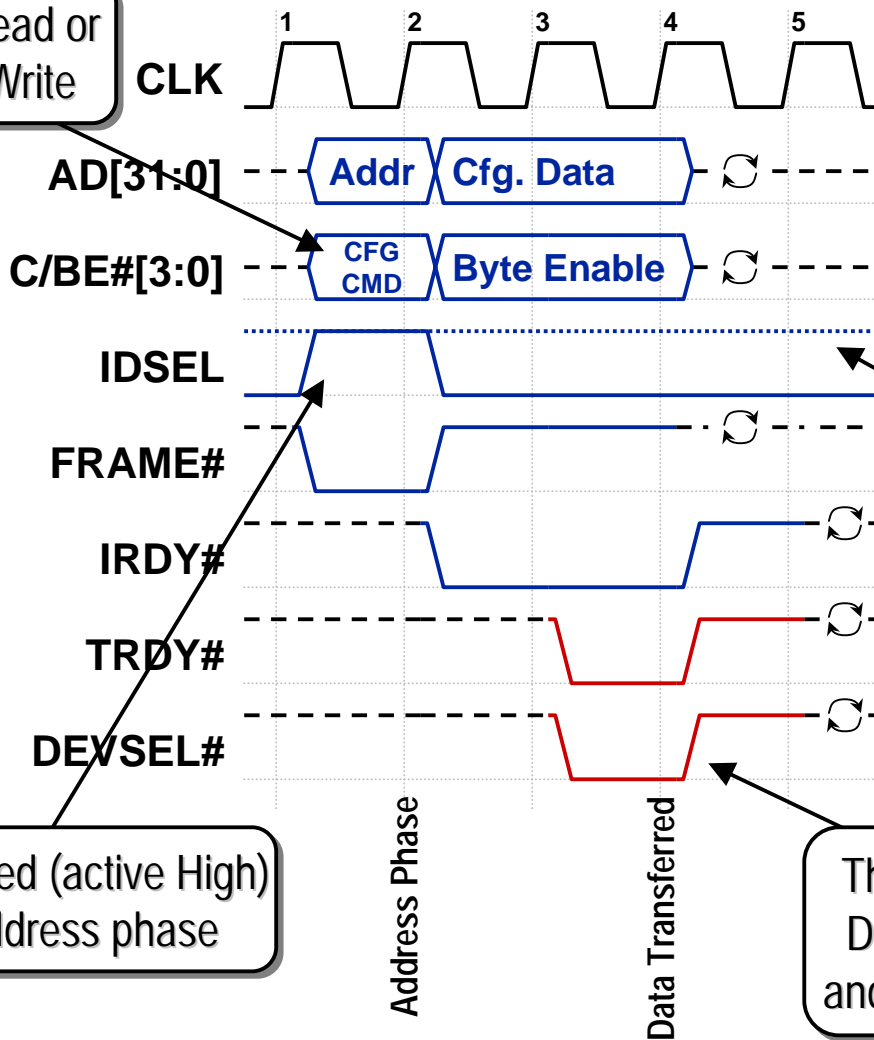
Configuration Transactions

- ◆ Are generated by a host or PCI-to-PCI bridge
- ◆ Use a set of IDSEL signals as chip selects
 - *Dedicated address decoding*
 - *Each agent is given a unique IDSEL signal*
- ◆ Are typically single data phase
 - *Bursting is allowed, but is very rarely used*
- ◆ Two types (specified via AD[1:0] in addr. phase)
 - *Type 0: Configures agents on same bus segment*
 - *Type 1: Configures across PCI-to-PCI bridges*

Configuration Example

| C/BE# | Command |
|-------|---------------------|
| 0000 | Interrupt Ack. |
| 0001 | Special Cycle |
| 0010 | I/O Read |
| 0011 | I/O Write |
| 0100 | Reserved |
| 0101 | Reserved |
| 0110 | Memory Read |
| 0111 | Memory Write |
| 1000 | Reserved |
| 1001 | Reserved |
| 1010 | Config. Read |
| 1011 | Config. Write |
| 1100 | Memory Read Mult. |
| 1101 | Dual Address Cycle |
| 1110 | Memory Read Line |
| 1111 | Memory Write & Inv. |

Configuration Read or Configuration Write

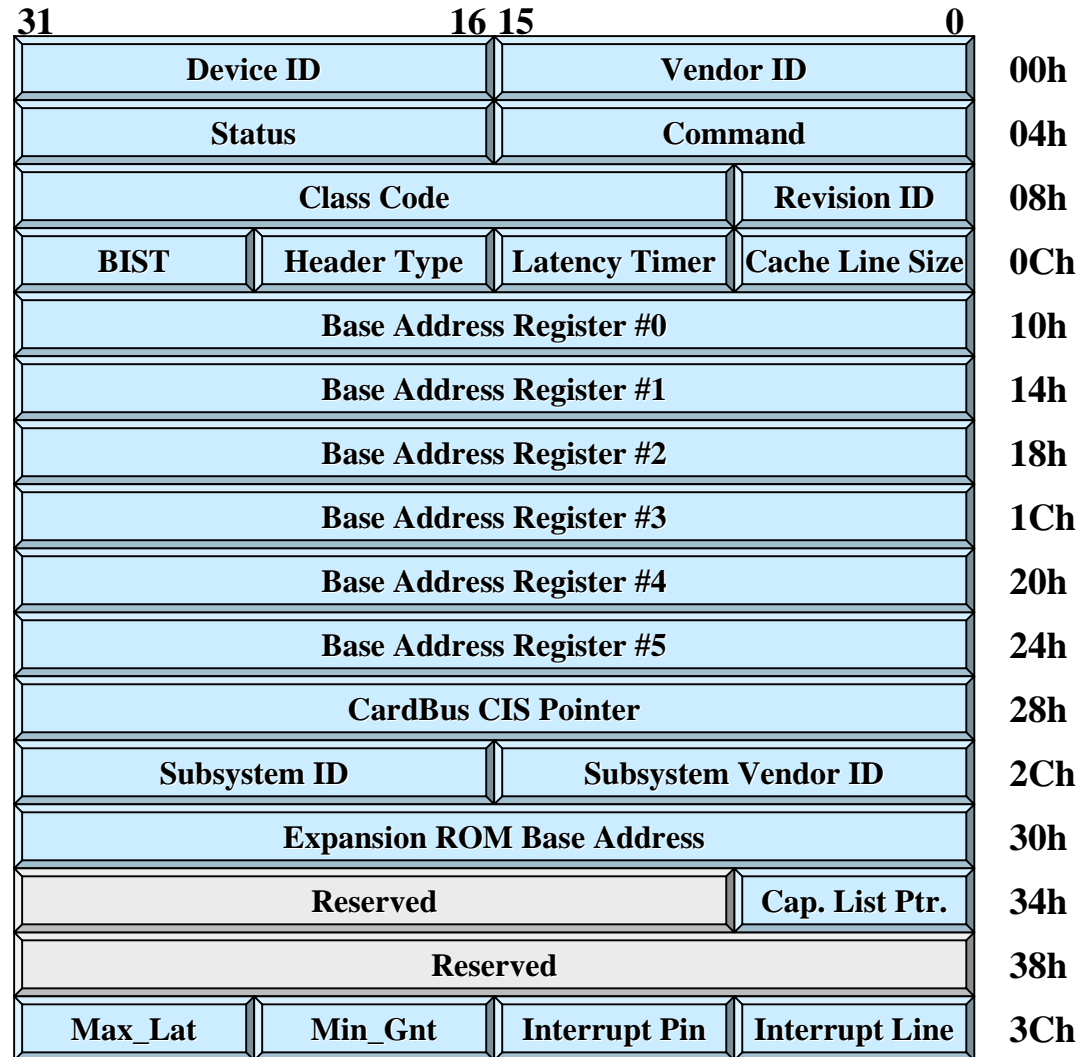


Note that the host can do anything it wants to IDSEL outside of a configuration address phase

IDSEL is asserted (active High) during the address phase

This time, the target asserts DEVSEL# based on IDSEL and not based on the address

Standard PCI Configuration Header



Required by PCI Spec 2.2

| | | | | | | |
|----------------------------|-------------|---------------|---------------------|---------------|--|--------------|
| 31 | | 16 15 | | 0 | | |
| Device ID | | Vendor ID | | 00h | | Required |
| Status | | Command | | 04h | | Not Required |
| Class Code | | | | Revision ID | | 08h |
| BIST | Header Type | Latency Timer | Cache Line Size | 0Ch | | |
| Base Address Register #0 | | | | | | 10h |
| Base Address Register #1 | | | | | | 14h |
| Base Address Register #2 | | | | | | 18h |
| Base Address Register #3 | | | | | | 1Ch |
| Base Address Register #4 | | | | | | 20h |
| Base Address Register #5 | | | | | | 24h |
| CardBus CIS Pointer | | | | | | 28h |
| Subsystem ID | | | Subsystem Vendor ID | | | 2Ch |
| Expansion ROM Base Address | | | | | | 30h |
| Reserved | | | | Cap. List Ptr | | 34h |
| Reserved | | | | | | 38h |
| Max_Lat | Min_Gnt | Interrupt Pin | Interrupt Line | 3Ch | | |

Electrical & Timing Specifications

Xilinx PCI

Signaling Environments

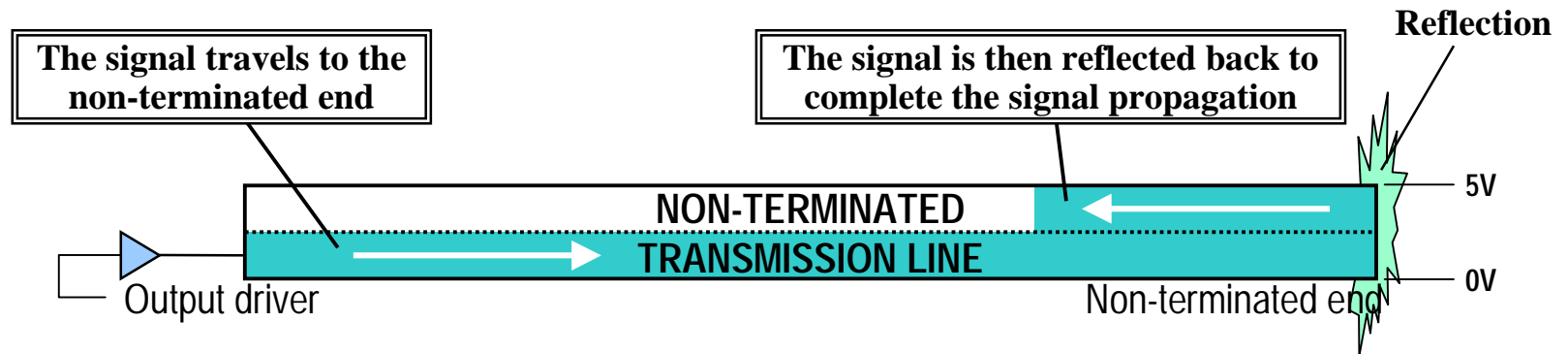
- ◆ The PCI spec describes two different electrical environments
 - *5V signaling*
 - *3.3V signaling*
- ◆ Technically, these names have *nothing* to do with the actual supply voltage
 - *Rather, they are tied to logic-level thresholds*

Signaling Environments

- ◆ 5V signaling is the most common
- ◆ 66MHz PCI buses can only use 3.3V signaling
 - *Note that 33MHz PCI can still use either*
- ◆ Some plug-in cards can support both signaling environments – these are known as “universal” cards

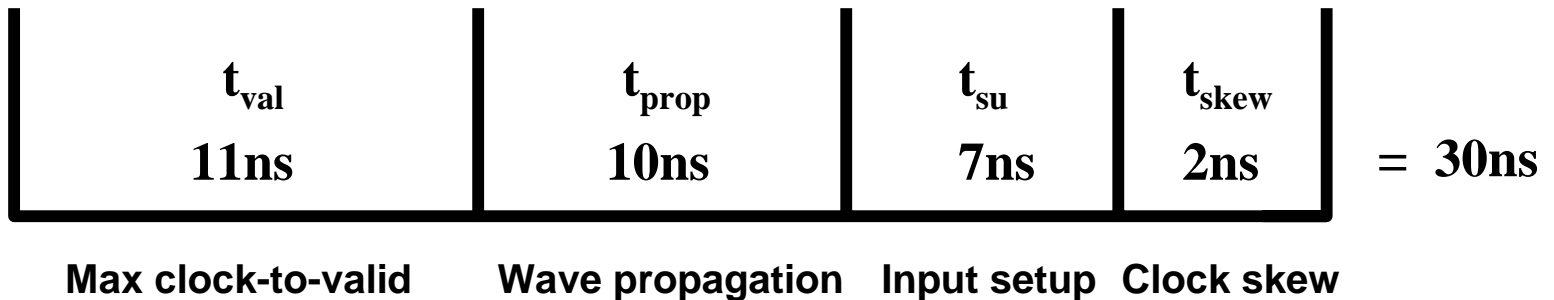
Reflective-Wave Switching

- ◆ PCI uses “reflective waves”
 - *Each wire on the PCI bus is a non-terminated transmission line, which causes signals to reflect over the length of the trace*
 - *Valid voltage levels are obtained after one reflection; this reduces the cost of PCI by not requiring high-powered output drivers*



33MHz PCI Timing Specification

★ 30ns Bus Cycle Time



Other Requirements:

Hold time : 0ns

Min clock-to-out : 2ns

Output off time : 28ns

All timing parameters are measured at the package pin

Add-In Card Design

- ◆ Trace length
 - *All 32-bit PCI signals must be no more than 1.5"*
 - *All 64-bit ext. signals must be no more than 2.0"*
- ◆ Clock trace must be exactly 2.5" (± 0.1 ")
 - *Routed to only one load*
 - *Needed for clock-skew control*
- ◆ PCI device requirements
 - *One pin per signal!*
 - *Max input capacitance is 10pF (unless the device is on the motherboard, where 16pF is OK)*

System Issues – Bus Loading

- ◆ No PCI spec requirement as to the loading on the bus; however:
 - *A driver must meet the 10ns propagation spec*
- ◆ **The rule of thumb is 10 loads max for 33MHz**
 - *Motherboard devices count as one load*
 - *Each add-in card slot counts as two load*
 - *Since most PC motherboards must have ≥ 2 PCI devices, they usually have no more than 4 slots*
- ◆ **More slots are available using PCI-to-PCI bridges or peer-to-peer PCI systems**

64-Bit and 66MHz PCI

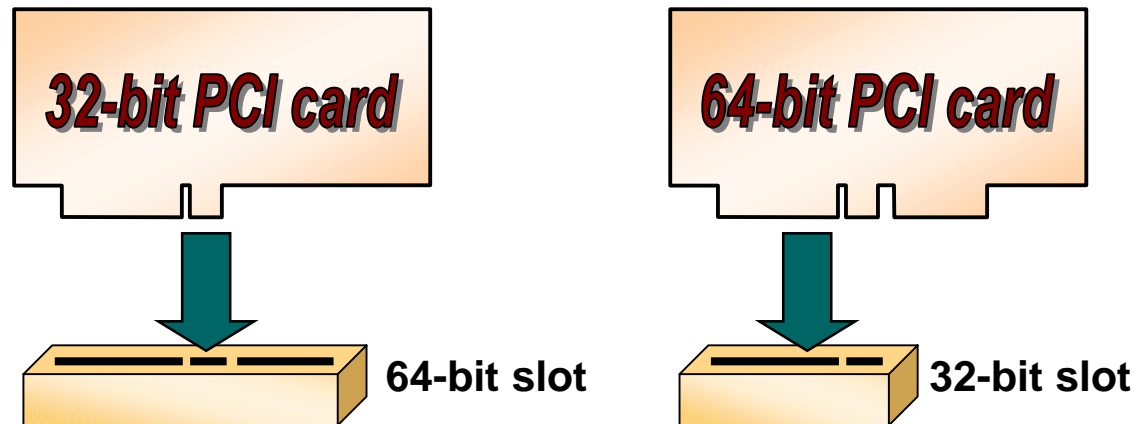
Xilinx PCI

The 64-Bit PCI Extension

- ◆ Doubles the available PCI bandwidth (keeping the clock frequency at 33MHz) to 264 MB/sec
- ◆ 64-bit PCI can use both 5V and 3.3V signaling
- ◆ Only used for memory transactions

The 64-Bit PCI Extension

- ◆ **Mixing and matching is allowed**
 - *A 32-bit card can be plugged into a 64-bit slot*
 - *A 64-bit card can be plugged into a 32-bit slot*
 - *Use of the 32-bit vs. 64-bit datapath is negotiated between the initiator and target at the start of each transaction*



Additional 64-Bit Pins

- ◆ **AD[63:32]**
- ◆ **C/BE#[7:4]**
 - *Used only for byte enables (not for PCI commands) except in the special case of Dual Address Cycle, discussed later*
- ◆ **PAR64**
 - *The XOR of AD[63:32], C/BE#[7:4], and PAR64 must equal zero (even parity)*

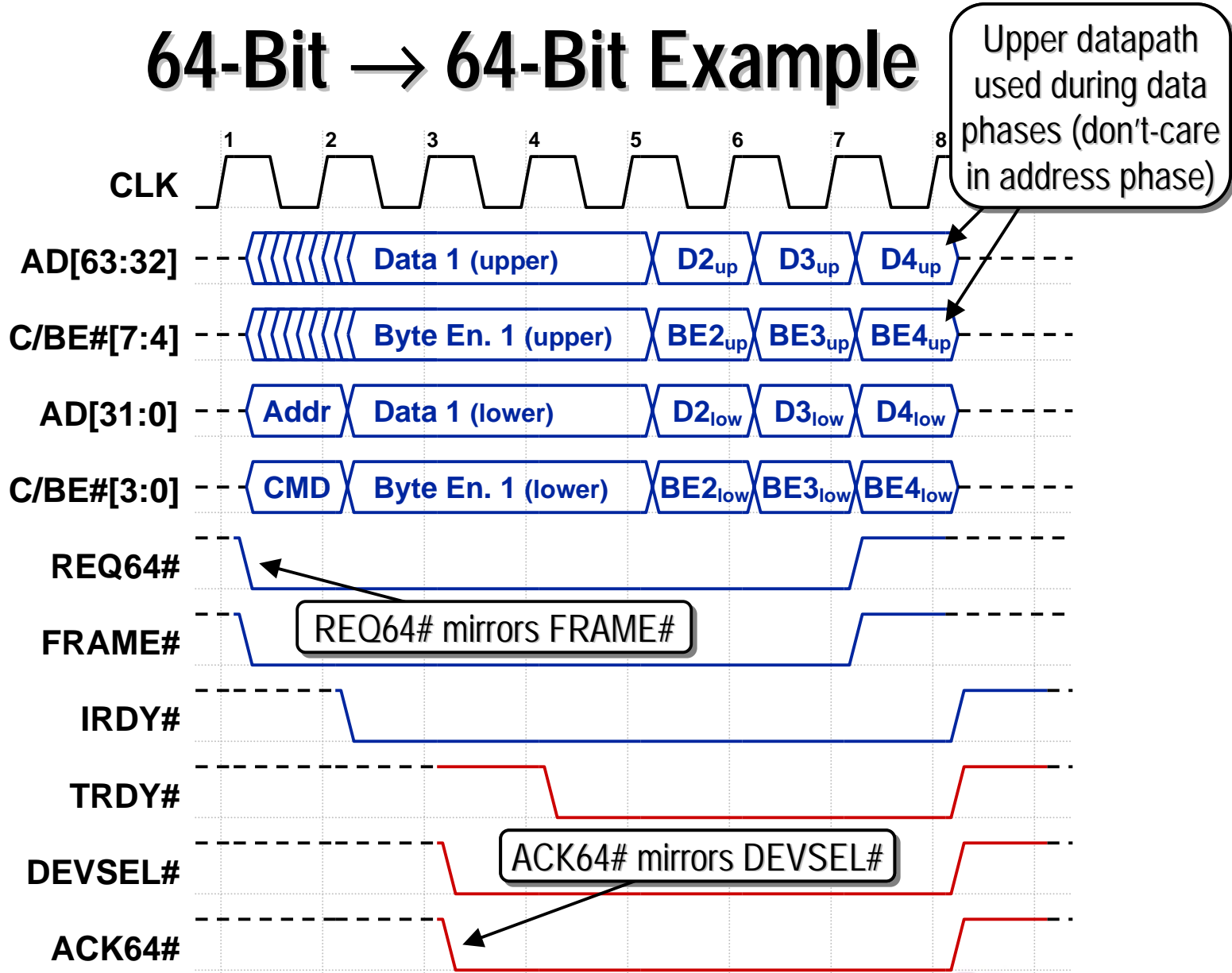
Additional 64-Bit Pins

- ◆ **REQ64#**
 - *Mirrors FRAME# – indicates that the initiator requests a 64-bit transaction*
- ◆ **ACK64#**
 - *Mirrors DEVSEL# – indicates that the target is able to fulfill the transaction request using the 64-bit datapath*
 - *By not asserting ACK64#, the target tells the initiator that it is a 32-bit agent*

64-Bit Initiator to 64-Bit Target

- ◆ Works the same as a standard 32-bit transaction, with the following additions:
 - *The initiator asserts REQ64# to mirror FRAME#*
 - *The target, in response, asserts ACK64# to mirror DEVSEL#*
- ◆ Data is transferred on **AD[31:0]** *and* **AD[63:32]**
 - *C/BE#[7:4] and PAR64 are also used*
- ◆ The starting address must be QUADWORD aligned (i.e., divisible by 8: **AD[2] = 0**)

64-Bit → 64-Bit Example



66MHz PCI Overview

- ◆ Pushes PCI bandwidth as high as 528MB/sec
- ◆ Most often used with the 64-bit extension, although it is legal to have 32-bit 66MHz PCI
- ◆ The signaling protocol is the same as with 33MHz PCI

66MHz PCI Overview

- ◆ 66MHz PCI can only use 3.3V signaling
- ◆ A device that can operate at 66MHz has the 66MHz Capable bit set in the Status Register
- ◆ The loading allowance is cut in half (5 loads), so only one or two add-in slots are possible
- ◆ For open systems, 66MHz add-in cards must also work on a 33MHz PCI bus