Chapter 14
PCI BUS Overview
PC Architecture for Technicians: Level-1

Systems Manufacturing Training and Employee Development
Copyright © 1996 Intel Corp.
OBJECTIVES: At the end of this section, the student will be able to do the following:

- Discuss Bus standards: ISA, EISA, VL BUS, & PCI
- Explain PCI Bus Architecture Fundamentals
- Discuss basic PCI bus cycles.
- Describe the Required PCI Bus signals.
- Discuss PCI Commands, Parity, Subtractive Decode, and PCI Arbitration
- Discuss the following as they relate PCI Configuration Access.
  - PCI Configuration Registers; PCI Configuration Mechanism Type -1; PCI ID Select Line usage.
The Peripheral Component Interconnect BUS
Why Local Bus?

Higher Performance, Higher Integration, and Low Cost

Basic PC
- CPU
  - Cache
- Memory Controller
- Expansion Bus I/F
- System Board
  - VGA
  - LAN
  - SCSI
  - Audio
  - Video
  - Modem
  - Add In
- ISA / EISA / Micro Channel

PCI PC
- CPU
  - Cache
- Memory Controller/Bridge
- Expansion Bus Bridge
  - LAN
  - SCSI
- High Bandwidth PCI Expansion
  - Video
  - Graphics
- Audio
- Modem
  - FAX
  - Add In
- ISA / EISA / Micro Channel

PCI Local Bus; 132 MBytes/Sec

LAN

Video

Modem

Add In
Introduction

- The ISA expansion bus in most of today’s PCs is based on a design that is over a decade old.
- Today’s multitasking operating systems and feature rich applications require not only a faster processor but better throughput to system peripherals.
  - High Resolution Graphics Video Adapter
  - Full Motion Video Adapter
  - SCSI Host Bus Adapter
  - Network Adapter
Introduction

Expansion Bus Transfer Rate Limitations

• **ISA** Bus Architecture
  - Transfers performed at 8/8.33 MHz clock speed.
  - 16 bits data bus
  - **8.33 MBytes/Sec Max.** (2 bytes in 2 clocks)

• **EISA** Bus Architecture
  - Clock speed at 8/8.33 MHz
  - 32 bits data bus
  - **33 MBytes/Sec Max.** (4 bytes in 1 clock)
Introduction

The PCI Bus Solution

- **PCI Local Bus**
  - Peripheral Component Interconnect
  - It is a high performance connection between motherboard components and expansion boards.
  - Transfer Rate of 132 MBytes per second maximum with 32-bit bus implementation
  - Transfer Rate of 264 MBytes per second with 64-bit bus implementation
  - 5 volts and 3.3 volts Operation
  - Clock Speed at 33 MHz or Less
    - 66 MHz implemented in Rev 2.1 of PCI Spec for 3.3V only.
Bus Standards
Introduction - Bus Standards

Introduction
The following pages describe the numerous bus standards implemented in today’s Personal Computers.

What is a Bus?
A bus is an electrical conduit in a computer that connects various components so they can communicate with each other. A bus standard is a set of rules that govern how the communications will take place….think of it as “grammar” for a computer. And different bus standards will have different grammatical rules that affect how quickly they can communicate.
PC Architecture Evolution

"Traditional PC"

- X86 Processor
- System I/O
  - ISA, EISA, MicroChannel
- Memory
- Graphics
- Disk
- Fax/Modem

FAST access to memory
SLOWER access to I/O

"PCI Local Bus" PC

- X86 Processor
- System I/O
  - ISA, EISA, MicroChannel Bus
- Memory
- Graphics
- Disk
- Fax/Modem

FAST access to memory
FAST access to I/O
Introduction - Bus Standards

**ISA**

When the PC was introduced in the early 1980’s, the Industry Standard Architecture (ISA) bus was used. As the PC’s popularity spread and “clones” appeared, the ISA bus was always used for compatibility. It is 16-bits wide at 8MHz, and is fully compatible with all PC software. An ISA system will accept ISA add-in cards. It is found in most PCs.
Introduction - Bus Standards

EISA
The Enhanced ISA, or EISA bus, is faster than ISA (32-bits @ 8MHz), and is fully compatible with ISA and all PC software. An EISA bus will give better overall performance to a computer system, and will accept ISA & EISA cards. It is typically used in servers, workstations and high-end PCs.

![Diagram showing CPU, Memory, EISA Bus, Keyboard, Disks, Other Elements, and EISA Slots with text overlay: EISA Bus: Improved overall performance]
Introduction - Bus Standards

VL / VESA Local Bus: In 1992, the VL Local Bus began to appear and was mainly used to increase the graphics performance. It is used in conjunction with an ISA bus. Since the VL Bus is attached directly to the CPU, both must run at the same speed. For electrical reasons, the VL Bus can only support 2 add-in cards at 33MHz, and none at 50MHz.
Introduction - Bus Standards

**PCI Bus:** Like the VL Bus, PCI will improve graphics performance, but will support other components as well (network, hard drive, etc.). Since it is not directly connected to the CPU, it sheds the electrical limitations of the VL bus and will accept multiple PCI-bus add-in cards.
PCI BUS ARCHITECTURE OVERVIEW
PCI Bus Terms

See the Glossary at end of chapter for more definitions.

- **Agent** - Each PCI device, whether a bus master (initiator) or a target is referred to as a PCI Agent.

- **Burst transfer** - The basic PCI bus transfer mechanism which consists of 1 address phase & multiple data phases.

- **Configuration cycle** - Bus cycles used for system initialization and configuration via the configuration address space.

- **PCI** - Peripheral Component Interconnect

- **Transaction** - An address phase plus one or more data phases.
Typical System Implementation

- “Bridges” isolate buses electrically and logically.

- Higher performance functions on PCI.
- Low performance functions on (E)ISA, MC/A.
Low, Mid-Range Desktop Platform

- Processor
- Bridge/ Memory Controller
- Cache
- DRAM
- Graphics
- Exp Bus Xface
- IDE
- ISA/EISA - MicroChannel
- Auto Configuration Ready
- Low System Cost
High-End Desktop Platform

- Scaleable PCI bus bandwidth - 64 bits addr & data
- Performance - linear burst; reads and writes

Diagram showing the components of a high-end desktop platform, including the processor, cache, memory controller, DRAM, bridge/motion controller, LAN, SCSI, external bus (Exp Bus Xface), ISA/EISA - MicroChannel, and audio and graphics cards.
PCI Mechanical

- PCI defines “shared” slots.
- The shared slot uses a different connector for the standard expansion card (e.g. ISA) and the PCI card.
  - Components are mounted on opposite side so both can use the same backplane area.
  - Cards are “mirrored” from ISA, EISA, and Micro Channel.
  - Existing chassis do not require modification to accommodate PCI cards.
Today’s desktop uses 5 V, but 3.3 V is coming

- Dual-voltage card works in both 5V and 3.3V systems

Connectors are MicroChannel-style, high density.

“Keys” prevent cards from being plugged into wrong systems.
Basic Electrical Consideration

- There are some unique electrical characteristics which differentiate the PCI bus from other busses on the market today.

- Most bus specifications today employ the technique of **Incident Wave Switching**.
  - This technique involves very strong output **drivers** that are capable of driving the output node to a valid logic level in one voltage step.
  - As the signal propagates down the trace, each device it passes detects the logic level and is switched.
  - Lines are terminated at the physical end of the trace to minimize reflections.
Basic Electrical Consideration

- The PCI Specification utilizes **Reflected Wave Switching**.

  - This technique employs weaker output drivers that are only capable of driving the output node to half the voltage of a valid logic level.
  
  - The transition to a valid logic level occurs when the signal travels to the end of the unterminated (open) bus and is reflected.

    - The wave is reflected in phase with the incident wave, effectively doubling the voltage on the bus.
    - The wave is absorbed by the low impedance driver.

- Generally speaking, each component is one load, each connector is one load, and the PCI card is one load.

  - An typical ChipSet can drive 10 PCI “loads”.
PCI BUS CYCLES
PCI Transfer

- The Initiator
- The Target
- The Agents
- The PCI Bus Clock
- The Address Phase
- Claiming the Transaction
- The Data Phase
- Transactions Duration
PCI Transfer - Details to Follow

All signals are qualified by the system clock.
PCI Transfer

- The PCI bus uses some multiplexed signals. Examples are:
  - **AD [31:0]** - Multiplexed Address and Data bus.
    - A typical bus cycle consists of one address phase and multiple data phases.
    - During the address phase, the signals carry the start Address.
    - During the data phase these signals carry the data objects being transferred between the initiator and target.
  - **C/BE [3:0]#** - Bus Commands & Byte Enables.
    - During the address phase, the signals represent Bus Commands
    - During the data phase these represent Byte Enable signals.
PCI Transfer

- **PCI Bus Clock** - This is a synchronous bus.
  - The frequency of the CLK in PCI may be anywhere from 0MHz to 33MHz.
    - 66MHz implemented in Rev 2.1 of PCI Bus Specification for 3.3V cards only.

- **Two participants in every PCI burst transfer.**
  - **The Initiator**
    - It is the device that initiates a transfer.
  - **The Target**
    - It is the device currently addressed by the initiator for the purpose of performing a data transfer.
PCI TRANSFER

- The Address Phase
  - Every transaction starts off with an **address phase** one PCI CLK period in duration.
  - During the address phase, the **initiator identifies the target device** (Address) and the **type of transaction** (Command - I/O or Memory Read / Write / Cfg).

- Claiming the Transaction
  - When a PCI agent determines that is the target of a transaction, it must **claim the transaction by asserting** a PCI signal intended for this purpose (DEVSEL#).
  - If the initiator doesn’t see this signal asserted within a predetermined amount of time, it will abort the transaction. (No system hang if target not ready)
PCI Transfer

The Data Phase

- It is the period during which a data object is transferred between the initiator and target.
  - Transfer when both the Initiator & Target are “READY”

- The amount of data to be transferred is determined by the Commands/Bytes Enable signals that are set active by the initiator during the data phase.

- Both the initiator and the target must indicate that they are ready to complete a data phase or the data phase will be extended by a wait state of one or more clock periods.
  - Wait states are inserted when either the Initiator or the Target are “NOT READY”
PCI Transfer

Transaction Duration

- The initiator identifies the overall duration of a burst transfer with a framing signal (FRAME#).
  - This PCI signal is asserted at the start of the address phase and remains active until the last data phase.
- The PCI Spec allows PCI masters to burst indefinitely as long as the target can provide or accept the data and there are no other requests for the bus.
- A Master Latency Timer (MLT) ensures that the current bus master will not hog the bus if the arbitrator indicates that another PCI master is requesting access to the bus.
  - The MLT also guarantees the current bus master a minimum amount of time before it must surrender the bus.
Basic Bus Cycle

- The most fundamental PCI transfer mechanism is the burst cycle.

- For a standard bus cycle, there must be a one-to-one correspondence between the number of addresses and the number of data transfers on the bus.
  - There must be one address for every data item.

- A burst cycle is different from a standard cycle in that the burst cycle contains one address phase and multiple data phases.
  - In a multiple-data phase transaction, it is the responsibility of the target to latch the start address and manage the address from data phase to data phase.
Basic Bus Cycle

- The motivation behind the concept of the burst cycle is the **Principle of Spatial Locality**.
  - “If a item is referenced, nearby items will tend to be referenced soon.”

- A simple illustration of this principle is the example of code fetching.
  - Most of the time these instructions will be located sequentially in memory.

- Burst cycles generally only apply to memory cycles.
  - Typically I/O devices are not capable of burst cycles and the Principle of Spatial Locality does not normally apply.
32-Bit Read Operation
All signals are qualified by the system clock.

4 bytes in 3 clks
44 MBytes/s
Basic Read Cycle

The Cycle begins with the assertion of the FRAME# signal in clock 1.

- This part of the cycle is termed the **ADDRESS PHASE**.
- An address is presented on the **AD[31:0]** bus
  - The address is latched by the target on the rising edge of clock 2.
- A valid **command** is placed on the **C/BE#** lines.
  - Example commands are:
    - Mem Read
    - I/O Write
    - Configuration Read
Basic Read Cycle

After the address phase, there is a **turnaround cycle** (i.e. dead cycle) (clock 2) required **to prevent bus contention** when one agent stops driving a signal and another agent begins.

- This cycle allows the master to stop driving the address on the bus, the output buffers on the master must be tristated by this time.
- AD bus ownership changes from initiator to target.
- A turnaround cycle must last one clock and is required on all signals that may be driven by more than one agent.
- The turnaround cycle is the clock between IRDY# asserted active and TRDY# asserted active.
Basic Read Cycle

- The data phase begins on clock 3 when IRDY# is driven active.
  - IRDY# is asserted during a read to indicate that the initiator is ready to accept data from the currently-addressed target.
- The target asserts the DEVSEL# signal during clock 3 to indicate that it has internally decoded the address and is responsible for the current bus cycle.
- When the target has output valid data, it asserts the TRDY# signal, telling the master that valid data is on the bus.
  - The earliest time the target can provide data is in clock 4.
Basic Read Cycle

The **data phase** (Cont.)

- Data transferred when both TRDY# & IRDY# are low.
- Wait states are inserted by holding the assertion of the TRDY# signal.
- The master can also insert wait states by driving the IRDY# signal inactive.
- FRAME# is deasserted on the cycle immediately previous to the cycle in which the final data item is latched by the master. (The next data transfer is the last).
- The cycle is concluded when both FRAME# and IRDY# are deasserted.
32-Bit Write Operation

Notice that no Turnaround Cycle is required

4 bytes in 2 clks
66 MBytes/s
Basic Write Cycle

- The write transaction begins with the assertion of FRAME# signal in clock 2.
  - This is very much like a Read Cycle.
  - For a write is important to notice that no turnaround cycle is required because the master will continue to drive the bus with valid data signals on the next data phase.
  - Notice that if IRDY# & TRDY# are both asserted for more than one clock cycle then new valid data is latched by the target on every clock cycle that both signals are asserted.
    - Continuous bursting with no wait states.
    - This applies to Read Cycles also.
PCI Signal Definition
PCI Signal Definition

**Required Pins**

- **Address & Data**
  - AD[31::00]
  - C/BE[3::0]#
  - PAR

- **Interface Control**
  - FRAME#
  - TRDY#
  - IRDY#
  - STOP#
  - DEVSEL#
  - IDSEL

- **Error Reporting**
  - PERR#
  - SERR#

- **Arbitration (masters only)**
  - REQ#
  - CNT#

- **System**
  - CLK
  - RST#

**Optional Pins**

- **64-Bit Extension**
  - AD[63::32]
  - C/BE[7::4]#
  - PAR64
  - REQ64#
  - ACK64#

- **Interface Control**
  - LOCK#
  - INTA#
  - NTB#
  - NTC#
  - INTD#

- **Interrupts**
  - SBO#
  - SDONE

- **Cache Support**
  - TDI
  - TDO
  - TCK
  - TMS
  - TRST#

**A minimal PCI master interface takes only 47 pins.**
<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Pin</th>
<th>Pin</th>
<th>Signal Name</th>
<th>Pin</th>
<th>Pin</th>
<th>Signal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>A1</td>
<td>B1</td>
<td>-12V</td>
<td>AD16</td>
<td>A32</td>
<td>AD17</td>
</tr>
<tr>
<td>+12V</td>
<td>A2</td>
<td>B2</td>
<td>No Connect</td>
<td>A33</td>
<td>B33</td>
<td>CBE2 #</td>
</tr>
<tr>
<td>No Connect</td>
<td>A3</td>
<td>B3</td>
<td>GND</td>
<td>A34</td>
<td>B34</td>
<td>GND</td>
</tr>
<tr>
<td>No Connect</td>
<td>A4</td>
<td>B4</td>
<td>No Connect</td>
<td>A35</td>
<td>B35</td>
<td>IRDY #</td>
</tr>
<tr>
<td>Vcc</td>
<td>A5</td>
<td>B5</td>
<td>Vcc</td>
<td>A36</td>
<td>B36</td>
<td>3.3V</td>
</tr>
<tr>
<td>PCIINT3 #</td>
<td>A6</td>
<td>B6</td>
<td>Vcc</td>
<td>A37</td>
<td>B37</td>
<td>DEVSEL #</td>
</tr>
<tr>
<td>PCIINT1 #</td>
<td>A7</td>
<td>B7</td>
<td>PCIINT2 #</td>
<td>A38</td>
<td>B38</td>
<td>GND</td>
</tr>
<tr>
<td>Vcc</td>
<td>A8</td>
<td>B8</td>
<td>PCIINT4 #</td>
<td>A39</td>
<td>B39</td>
<td>PLOCK #</td>
</tr>
<tr>
<td>Reserved</td>
<td>A9</td>
<td>B9</td>
<td>No Connect</td>
<td>A40</td>
<td>B40</td>
<td>PERR #</td>
</tr>
<tr>
<td>Vcc</td>
<td>A10</td>
<td>B10</td>
<td>Reserved</td>
<td>A41</td>
<td>B41</td>
<td>3.3V</td>
</tr>
<tr>
<td>Reserved</td>
<td>A11</td>
<td>B11</td>
<td>No Connect</td>
<td>A42</td>
<td>B42</td>
<td>SERR #</td>
</tr>
<tr>
<td>GND</td>
<td>A12</td>
<td>B12</td>
<td>GND</td>
<td>A43</td>
<td>B43</td>
<td>3.3V</td>
</tr>
<tr>
<td>GND</td>
<td>A13</td>
<td>B13</td>
<td>GND</td>
<td>A44</td>
<td>B44</td>
<td>CBE1 #</td>
</tr>
<tr>
<td>Reserved</td>
<td>A14</td>
<td>B14</td>
<td>Reserved</td>
<td>A45</td>
<td>B45</td>
<td>AD14</td>
</tr>
<tr>
<td>PCIINT3 #</td>
<td>A15</td>
<td>B15</td>
<td>GND</td>
<td>A46</td>
<td>B46</td>
<td>GND</td>
</tr>
<tr>
<td>Vcc</td>
<td>A16</td>
<td>B16</td>
<td>PCLKE</td>
<td>A47</td>
<td>B47</td>
<td>AD12</td>
</tr>
<tr>
<td>Reserved</td>
<td>A17</td>
<td>B17</td>
<td>GND</td>
<td>A48</td>
<td>B48</td>
<td>AD10</td>
</tr>
<tr>
<td>GND</td>
<td>A18</td>
<td>B18</td>
<td>REQA #</td>
<td>A49</td>
<td>B49</td>
<td>GND</td>
</tr>
<tr>
<td>Reserved</td>
<td>A19</td>
<td>B19</td>
<td>Vcc</td>
<td>A50</td>
<td>B50</td>
<td>KEY</td>
</tr>
<tr>
<td>AD30</td>
<td>A20</td>
<td>B20</td>
<td>AD31</td>
<td>A51</td>
<td>B51</td>
<td>KEY</td>
</tr>
<tr>
<td>3.3V</td>
<td>A21</td>
<td>B21</td>
<td>AD29</td>
<td>A52</td>
<td>B52</td>
<td>AD8</td>
</tr>
<tr>
<td>AD28</td>
<td>A22</td>
<td>B22</td>
<td>GND</td>
<td>A53</td>
<td>B53</td>
<td>AD7</td>
</tr>
<tr>
<td>AD26</td>
<td>A23</td>
<td>B23</td>
<td>AD27</td>
<td>A54</td>
<td>B54</td>
<td>3.3V</td>
</tr>
<tr>
<td>GND</td>
<td>A24</td>
<td>B24</td>
<td>AD25</td>
<td>A55</td>
<td>B55</td>
<td>AD5</td>
</tr>
<tr>
<td>AD24</td>
<td>A25</td>
<td>B25</td>
<td>3.3V</td>
<td>A56</td>
<td>B56</td>
<td>AD3</td>
</tr>
<tr>
<td>IDSEL</td>
<td>A26</td>
<td>B26</td>
<td>CBE3 #</td>
<td>A57</td>
<td>B57</td>
<td>GND</td>
</tr>
<tr>
<td>3.3V</td>
<td>A27</td>
<td>B27</td>
<td>AD23</td>
<td>A58</td>
<td>B58</td>
<td>AD1</td>
</tr>
<tr>
<td>AD22</td>
<td>A28</td>
<td>B28</td>
<td>GND</td>
<td>A59</td>
<td>B59</td>
<td>Vcc</td>
</tr>
<tr>
<td>AD20</td>
<td>A29</td>
<td>B29</td>
<td>AD21</td>
<td>A60</td>
<td>B60</td>
<td>SACK64 #</td>
</tr>
<tr>
<td>GND</td>
<td>A30</td>
<td>B30</td>
<td>AD19</td>
<td>A61</td>
<td>B61</td>
<td>Vcc</td>
</tr>
<tr>
<td>AD18</td>
<td>A31</td>
<td>B31</td>
<td>3.3V</td>
<td>A62</td>
<td>B62</td>
<td>Vcc</td>
</tr>
</tbody>
</table>

**5V Board**

**B49 M66EN**

On 3.3V Cards
PCI Signal Definition

PCI Signals - Required Pins (System)

- **RST# - PCI Reset Signal** (Asynchronous)
  - Forces all PCI configuration registers & output drivers to an initialized state.
  - All PCI signals in general should be driven to their benign states when RST# is activate.
    » In general this means they must be tri-stated.
    » Implemented in the platform support logic (i.e, the chipset [e.g. 82371FB PIIX])
PCI Signal Definition

PCI Signals - Required Pins (System)

- **CLK- PCI Clock Signal**
  - The PCI Clock Signal provides timing for all transactions, including bus arbitration's.
  - All inputs to PCI devices are sampled on the rising edge of the CLK signal (0-1 transition).
  - The frequency may be anywhere between 0MHz & 66MHz.
    - 66MHz implemented in Rev 2.1 of PCI Bus Specification for 3.3V cards only.
    - New Pin (M66EN) added to implement 66 MHz which is grounded by all 33 MHz 3.3V cards
    - If M66EN is a logic 0, the Max. CLK frequency is 33 MHz.
    - If M66EN is a logic 1, the Max. CLK frequency is 66 MHz.
PCI Signal Definition

**PCI Signals - Required Pins (Address/Data Bus)**

- **AD[31:0] - Multiplexed Address and Data bus.**
  - A typical bus cycle consists of one address phase and multiple data phases.
  - During the address phase, the signals carry the start Address.
  - During the data phase these signals carry the data objects being transferred between the initiator and target.

- **C/BE[3:0]#-Multiplexed Bus Command/Bytes Enables**
  - Defines the type of transaction.
  - Represent Bus Commands during the address phase.
  - Represent Byte Enable signals During the data phase.
PCI Signal Definition

C/BE[3:0]# - Represent Bus Commands during the address phase

C3 C2 C1 C0

0 0 0 0  Interrupt Acknowledge
0 0 0 1  Special Cycle
0 0 1 0  I/O Read
0 0 1 1  I/O Write
0 1 0 0  Reserved
0 1 0 1  Reserved
0 1 1 0  Memory Data Read (1-15 Bytes)
0 1 1 1  Memory Data Write (1-15 Bytes)
1 0 0 0  Reserved
1 0 0 1  Reserved
1 0 1 0  PCI Configuration Read (IDSEL)
1 0 1 1  PCI Configuration Write (IDSEL)
1 1 0 0  Memory Data Read (96< Bytes)
1 1 0 1  Dual Address Command (2 Cycles)
1 1 1 0  Memory Data Read Line (16-95 Bytes)
1 1 1 1  Memory Data Write and Invalidate (Burst Writeback Cycle - 32< Bytes)
PCI Bus Parity

Generating address and data parity is required for PCI.

Parity generation is delayed by one clock and error signalling is delayed by one more.
PCI Signal Definition

PCI Signals - Required Pins (Address/Data Bus)

- PAR (Even Parity) - Protects AD[31:0] & C/BE#[3:0]
  - The computed PAR but must be set (or cleared) so that the 37-bit field consisting of AD[31:0] & C/BE#[3:0] and PAR contains an EVEN number of one bits.
  - Driven by the initiator 1 clock after completion of the address phase and completion of each data phase of write transactions.
  - Driven by the target 1 clock after completion of each data phase of read transactions.
  - The computed Parity Bit is compared to the Parity Bit actually received on the PAR line to determine corruption.
    » A Parity Error is reported on the PERR# Signal.
PCI Signal Definition

PCI Signals - Required Pins (Interface Control)

• FRAME#
  - It indicates the start (when it is first asserted) and duration (the duration of the assertion) of an access.
  - FRAME# is deasserted when the initiator is ready to complete the final Data Phase.
    » The next Data Transfer is the final Data Phase.

• TRDY# (Target Ready)
  - It is driven active when the target is ready to complete the current data phase (data transfer).
  - The data is transferred when the target is asserting TRDY# and the master is asserting IRDY#.
PCI Signal Definition

PCI Signals - Required Pins (Interface Control)

- **IRDY# (Initiator Ready)**
  - During a write, it indicates that the initiator is driving valid data onto the data bus.
  - During a read, it indicates that the initiator is ready to accept data from the currently-addressed target.

- **STOP#**
  - The target asserts STOP# to indicate that it wishes the initiator to prematurely end the transfer in progress.
  - This is a Target-Abort signal.
    - For example: Stop is asserted by the target if the target can’t support the Burst request or if it takes more than 8 PCI clocks to complete the data phase.
Target Device Select

- PCI can deal with targets with various address decode speeds.

- Extra address decode time (up to 2 clocks for medium & slow) allows varying price/performance designs.

- DEVSEL# timing is in Configuration Status Reg (Bit 10:9)
PCI Signal Definition

PCI Signals - Required Pins (Interface Control)

• DEVSEL# - (Device Select)
  - It is an output driven active by a target when the target has decoded its address.
  - It is an input to the current master and used for a Master Abort if the initiator does not detect DEVSEL# within 6 clocks.
  - It is an input to the Subtractive Decoder in the Expansion Bridge and used to claim the transaction if it does not detect DEVSEL# within 3 clocks (Fast, Med, Slow) after FRAME# is asserted.

    » DEVSEL# is asserted by the Expansion Bridge during Clock 5 for Subtractive Decode.
Subtractive Address Decode

- Cost effective mechanism to deal with a fragmented memory map. **Only 1 agent can implement Subtractive Decode.**
Target Device Select (cont.)

- **Subtractive decoding definition.**
  - A method of address decoding in which a device accepts all accesses not **positively decoded** by another agents.

- **Positive Decoding** - A method of address decoding in which a device responds to accesses only within an assigned address range.

- The PCI-Expansion bus is designed to claim many transactions not claimed by other devices on the bus.
  - When a transaction is not claimed by any other PCI device within a specified period of time, the PCI-Expansion bus bridge may assert DEVSEL# and pass the transaction through to the expansion bus (e.g. ISA Bus)
PCI Configuration Cycles

- Each PCI Device (IC & Slot) has one IDSEL line.
  - Typically upper address lines are connected to IDSEL signals.
PCI Signal Definition

- **PCI Signals - Required Pins** (Interface Control)
  - **IDSEL - Initialization Device Select**
    - IDSEL is used as a PCI device *chip select during configuration* read & write transactions.
      - Each PCI Device (IC & Slot) has one IDSEL line.
    - Assertion of a specific IDSEL during the address phase of a configuration access is used to select the physical package that is the target of the configuration access.
    - It is an Input for both Masters & Targets.
  - **NOTE**: More information on IDSEL in section on PCI Configuration Access.
Signal Description

PCI Signals - Required Pins (Error Reporting)

• PERR# (Parity Error)
  • Parity information is generated for all PCI devices that drive address or data information onto the AD bus.
  • A Parity Error is reported on the PERR# Signal if the computed Parity Bit does not compare to the Parity Bit actually received on the PAR line.

• SERR# (System Error)
  • This is considered a “last-recourse” for reporting serious errors and typically causes an NMI.
  • Is used to report address parity error, data parity error during a special cycle, problems other than parity detected by a PCI device.
Signal Description

PCI Signals - Required Pins (Arbitration Signals)

Hidden Arbitration (Bus Masters Only)

- Bus Arbitration can take place while another initiator is still in control of the bus.

- **REQ# (PCI Bus Request)** Output
  - When an initiator requires the use of the PCI bus, it asserts its device-specific REQ# line to the arbiter.

- **GNT# (PCI Bus Grant)** Input
  - When the arbiter has determined that the requesting initiator should be granted control of the PCI bus, it asserts the GNT# (Grant) line specific to the requesting initiator.
PCI Bus Arbitration

Bus arbitration is centralized.

- Number of masters limited only by number of REQ# / GNT# pairs supplied by the arbiter (e.g. 82437 TSC REQ#[3:0])
  - Arbitration method not specified (e.g. Rotating priority).
  - “Fairness” algorithm is implemented by designer.
PCI Bus Arbitration (cont.)

Arbitration cycles on the bus happen in parallel with the current bus cycle (if any). No arbitration overhead.
PCI Bus Arbitration (cont.)

- Arbitration cycles on the bus happen in parallel with the current bus cycle (if any). No arbitration overhead.
- Each bus master has it’s own request and grant signal, which are intercepted and used by the arbiter.
- The PCI specification allows PCI masters to burst indefinitely so long as the target can provide or accept the data and no other agent wants the bus.
- The PCI specification prevents slow targets from monopolizing the bus by requiring targets to retry if the transactions take longer than:
  - the “16 clock rule” to complete the first data phase
  - the “8 clock rule” to complete the current data phase.
PCI Bus Arbitration (cont.)

- Each master also has an associated **Master Latency Timer** (MLT) which determines the maximum number of PCI clocks that the master can be own the bus when another request is pending.
  - The MLT is decremented by one on each PCI clock.
    - Once the timer expires **AND the GNT signal is deasserted**, the master must initiate termination of the cycle.
  - The LT ensures that the current bus master will not hog the bus if the arbitrator indicates that another PCI master is requesting access to the bus.
  - The LT also guarantees the current bus master a minimum amount of time before it must surrender the bus.
PCI DEVICE CONFIGURATION
PCI Device Configuration

- Another important aspect of the PCI specification is the auto-configuration of PCI devices on the bus.

- With ISA today, when a one adds a new I/O card to the PC, one must be aware of the I/O requirements of the card to be added.
  - Typically this involves setting jumpers and adjusting base addresses and interrupts. (e.g. I/O = 340H, IRQ5)

- The PCI specification requires that all devices must have a set of configuration registers that are examined upon power up.
  - Jumpers and switches cannot be used for PCI configuration.
PCI Device Configuration

- Upon power up, the configuration software (probably the BIOS) detects the presence of all PCI devices on the bus.
  - Once detected, the software assesses the configuration space of each device and assigns memory and I/O regions that are guaranteed not to be used by other PCI devices.
    - All the configuration will be performed automatically.
- The visible results is that the end user merely powers down the PC, inserts the new card, and then powers up the PC.
  - Plug-n-Play
PCI Configuration Space

PCI Configuration Registers

- PCI devices implement a 256 byte space for Configuration Registers which provide a software interface for board and system set-up.
  - Organized as 64 DWord Registers
    - \[64 \text{Dwords} \times 4 \text{bytes/DWord} = 256 \text{ bytes}\]
  - The first 64 byte locations are predefined by the PCI specification.
    - 64 byte Header region -> 40H bytes (00H to 3FH)
  - The remaining 192 locations are device-specific.
These registers contain data items including:

- **Vendor ID**
  - 8086H = Intel; 1002H = ATI
- **Device ID**
  - 122EH = 82371FB PIIX Fn 0; 4158H = MACH32 Video
- **Status Register**
- **Command Register**
- **Class Code**
  - 0601H = ISA Bridge; 0101H = IDE controller
- **Revision ID**
  - Depends on stepping of device (e.g. - 01H).
## PCI Configuration Space

### Format of a PCI Device’s Configuration Header

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Device ID</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
</tr>
<tr>
<td>29</td>
<td>Vendor ID</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
</tr>
<tr>
<td>27</td>
<td>Status</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
</tr>
<tr>
<td>25</td>
<td>Command</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
</tr>
<tr>
<td>23</td>
<td>Class Code</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td>21</td>
<td>Revision ID</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
</tr>
<tr>
<td>19</td>
<td>BIST</td>
</tr>
<tr>
<td>18</td>
<td>Header Type</td>
</tr>
<tr>
<td>17</td>
<td>Latency Timer</td>
</tr>
<tr>
<td>16</td>
<td>Cache Line Size</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>Interrupt Pin</td>
</tr>
<tr>
<td>11</td>
<td>Max_Lat</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>Min_Gnt</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>Interrupt Line</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>Expansion ROM Base Address</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### Base Address Registers

- **3F**
- **3E**
- **3D**
- **3C**
PCI Configuration Space

- Functional Device’s Header Region
  - **Vendor ID** - Offset 00H & 01H (R/O register)
    - Identifies the manufacturer of the device. (e.g. Intel)
  - **Device ID** - Offset 02H & 03H (R/O register)
    - Assigned by the device manufacturer and identifies the type of device (e.g. TSC, PIIX, MACH32 Video)
  - **Command Register** - Offset 04H & 05H (R/W register)
    - Provides basic control over the device’s ability to respond to PCI accesses. Used by BIOS.
  - **Status Register** - Offset 06H & 07H
    - Tracks the status of PCI bus-related events. Used by BIOS.
PCI Configuration Space

• **Revision ID** - Offset 08H (R/O register)
  - Indicates Stepping of Device. Used by BIOS.

• **Base Class Code** - Offset 0BH (R/O register)
  - Base Class identifies the basic function.
  - (e.g. Mass Storage; Display Controller)
  - **Examples of defined Class Codes are:**
    - 01H - Mass Storage; 02H - Network Controller:
    - 03H - Display Controller; 04H - Multimedia Device.

• **Sub-Class Code** - Offset 0AH (R/O register)
  - Sub-Class is more specific.
  - (e.g. IDE controller; VGA controller)
PCI Configuration Example

- The PCI Configuration Address register is a 32-bit register located at CF8H, the PCI Configuration Data register is a 32-bit register located at CFCh.
  - These registers are accessible by DWORD, WORD or BYTE accesses

- HOW TO ACCESS I/O CONFIGURATION SPACE USING CONFIGURATION MECHANISM #1
  - 1. Using a DWORD write command, output the desired I/O configuration address to I/O port CF8H
  - 2. Using a DWORD read or write command, read or write data from the I/O port CFCh
PCI Configuration Example

- **CONFIG SPACE ENABLE FLAG (Bit 31):** Always 1 to indicate I/O access is to configuration space.
- **RESERVED (Bits 30-24):** Always 00h
- **BUS NUMBER (Bits 23-16):** Always 00h unless a bridge card is installed in a PCI slot
- **DEVICE NUMBER (Bits 15-10):** Used to indicate a specific PCI device. (e.g. The TSC, PIIX and PCI slots)
- **FUNCTION NUMBER (Bits 10-8):** PIIX is multi-function device.
- **REGISTER NUMBER (Bits 7-0):** Defines one of 64 DWORD locations for a specific PCI device. Note that Bits 1 and 0 must always be 0H (Zero Hex) for DWORD access.
Using Mechanism - 1, Type 0
Accessing a PCI config. port is a 2 step process:

- **Point & Shoot**
  - 1. Write the BUS#, DEV#, FCN# & REG# to **CFG ADDR PORT**
    - **Point:** dport (0CF8) = CONE  BUS#  DEV#  FCN#  REG
    - e.g. - dport (0CF8) = 8000xxxx; dport (0CF8) = 80003808
  - 2. Perform an I/O read or write to the **CFG DATA PORT**.
    - **Shoot:** dport (0CfC)
    - e.g. - dport (0CFC) [read], dport (0CFC) = xxxxxxxx [write]
PCI Configuration Example

0CF8h is the Configuration Address Port (Point)

- Must be accessed using a full 32-bit write.
- Bit 31 Cfg. Space Enable (CONE) must be set to a 1
- Point to a Register in a specific Device with CF8H.

CONE BUS#  DEV#  FCN#  REG

dport (0CF8) = 80003800 (ITP example)

- 32 BIT Port (0CF8) contains I/O Addr CF8, CF9, CFA, & CFB

```
dport(0cf8)=80003800H
1000 0000 0000 0000 0011 1000 0000 0000
```

<table>
<thead>
<tr>
<th>31-30</th>
<th>24-23</th>
<th>16-15</th>
<th>11-10</th>
<th>8-7</th>
<th>2-1-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reserv</td>
<td>Bus Number</td>
<td>Dev #</td>
<td>FCN</td>
<td>Register #</td>
</tr>
</tbody>
</table>

HOST DAT A BUS
PCI Configuration Example

0CFCh is the Configuration Data Port (Shoot)

- 32 BIT Port (0CFC) contains I/O Addr CFC, CFD, CFE, & CFF
- Read or Write the Data at the address pointed to at CF8H.
- NOTE: Any address output to CF8h is always on a 4 byte (DWORD) boundary.
  - ITP uses Dport for 4 bytes, Wport for 2 bytes, Port for 1 Byte

- You can read or write any DWORD, WORD, or BYTE in the four byte range by using the correct offset as follows:
  - DWORD @ CFCh
  - WORD @ CFCh or CFEh
  - BYTE @ CFCh, CFDh, CFEh or CFFh
PCI Configuration Example

Examples of Accessing PCI Configuration Space using an ITP:

- Read Example:
  - dport (0CF8H) = 80000060H
  - dport (0CFCH)

- Write Example:
  - dport (0CF8H) = 80000060H
  - dport (0CFCH) = 02020000H
PCI Configuration Cycles

- Typically upper address lines are connected to IDSEL signals.
- Newer Devices use AD11-31, Older Devices used AD16-31.

Each PCI Device (IC & Slot) has one IDSEL line.
PCI Configuration Cycles

- IDSEL is used as a PCI device chip select during configuration read & write transactions.
  - Assertion of a specific IDSEL during the address phase of a configuration access is used to select the physical package that is the target of the configuration access.

- Each PCI Device (IC & Slot) has one IDSEL line.

Example implementation follows:

- The IDSEL associated with device 0 is connected to AD11
- The IDSEL associated with device 1 is connected to AD12 etc.
- The IDSEL associated with device 7 is connected to AD18.
- The IDSEL associated with device 20 is connected to AD31.
  - Note: The Host Bridge is always Device #0 & does not pass it Cfg cycles to the PCI bus, thus AD11 is never asserted.
PCI Config. Cycles: Mechanism-1, Type 0

IDSEL: 1 line set high by Bridge Device based on decoding of the target’s physical Device number contained in Config Address bits 15:11 (DEV#) (Newer Devices use AD11-31)

[e.g. dport(0cf8)=80003800H: Dev #7 -> AD18]

```
1000 0000 0000 0000 0011 1000 0000 0000
```

PCI Config. Cycles: Mechanism-1, Type 0

<table>
<thead>
<tr>
<th>HOST</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUS</td>
<td>BUS</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Reserved</td>
<td>Bus Number</td>
</tr>
<tr>
<td>Dev #</td>
<td>FCN</td>
</tr>
<tr>
<td>R egister #</td>
<td>R</td>
</tr>
</tbody>
</table>

"IDSEL" Alias - Only 1 bit set!

```
31 AD18 11-10 8-7 2-1-0
```

PC Architecture For Technicians  Level-1
Technical Excellence Development Series
I/O Cfg Addr -> Device Number Mapping

Newer Devices use AD11-31

**Lower 16 bits of Host Data Bus**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Dev #</th>
<th>IDSEL</th>
<th>15,14,13,12</th>
<th>11</th>
<th>10, 9, 8</th>
<th>7,6,5,4</th>
<th>3,2,1,0</th>
</tr>
</thead>
<tbody>
<tr>
<td>00XX</td>
<td>0</td>
<td>AD11</td>
<td>0000</td>
<td>0</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>08XX</td>
<td>1</td>
<td>AD12</td>
<td>0000</td>
<td>1</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>10XX</td>
<td>2</td>
<td>AD13</td>
<td>0001</td>
<td>0</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>18XX</td>
<td>3</td>
<td>AD14</td>
<td>0001</td>
<td>1</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>20XX</td>
<td>4</td>
<td>AD15</td>
<td>0010</td>
<td>0</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>28XX</td>
<td>5</td>
<td>AD16</td>
<td>0010</td>
<td>1</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>30XX</td>
<td>6</td>
<td>AD17</td>
<td>0011</td>
<td>0</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>38XX</td>
<td>7</td>
<td>AD18</td>
<td>0011</td>
<td>1</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>40XX</td>
<td>8</td>
<td>AD19</td>
<td>0100</td>
<td>0</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>48XX</td>
<td>9</td>
<td>AD20</td>
<td>0100</td>
<td>1</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>50XX</td>
<td>A</td>
<td>AD21</td>
<td>0101</td>
<td>0</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>58XX</td>
<td>B</td>
<td>AD22</td>
<td>0101</td>
<td>1</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>60XX</td>
<td>C</td>
<td>AD23</td>
<td>0110</td>
<td>0</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>68XX</td>
<td>D</td>
<td>AD24</td>
<td>0110</td>
<td>1</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>70XX</td>
<td>E</td>
<td>AD25</td>
<td>0111</td>
<td>0</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
<tr>
<td>78XX</td>
<td>F</td>
<td>AD26</td>
<td>0111</td>
<td>1</td>
<td>000</td>
<td>XXXX</td>
<td>XX00</td>
</tr>
</tbody>
</table>

PC Architecture For Technicians  Level-1
Technical Excellence Development Series
### PCI Config. Cycles: Mechanism-1, Type 0

**IDSEL:** 1 line set high by Bridge Device based on decoding of the target’s physical Device number contained in Config Address bits 15:11 (DEV#) *(Older Devices use AD16-31)*

<table>
<thead>
<tr>
<th>Bus Number</th>
<th>Dev #</th>
<th>Register #</th>
<th>FCN</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 0000 0000 0000 0011 1001 0000 0000</td>
<td><strong>IDSEL</strong> Alias - Only 1 bit set!</td>
<td>R</td>
<td>FCN</td>
<td>Register #</td>
</tr>
</tbody>
</table>

[e.g. dport(0cf8)=80003900H: Dev #7 -> AD23]

---

**PCI A/D BUS**

<table>
<thead>
<tr>
<th>31</th>
<th>AD23</th>
</tr>
</thead>
</table>

**HOST DATA BUS**

<table>
<thead>
<tr>
<th>31-30</th>
<th>24-23</th>
<th>16-15</th>
<th>11-10</th>
<th>8-7</th>
<th>2-1-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reserved</td>
<td>Bus Number</td>
<td>Dev #</td>
<td>FCN</td>
<td>Register #</td>
</tr>
</tbody>
</table>

*IDSEL* set high for clock cycles by Bridge Device based on the target’s physical Device number contained in Config Address bits 15:11 (DEV#). *(Older Devices use AD16-31)*
## I/O Cfg Addr -> Device Number Mapping

### Older Devices used AD16-31

#### Lower 16 bits of Host Data Bus

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Dev #</th>
<th>IDSEL</th>
<th>Device #</th>
<th>FCN</th>
<th>Register #</th>
</tr>
</thead>
<tbody>
<tr>
<td>00XX</td>
<td>0</td>
<td>AD16</td>
<td>0000</td>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>08XX</td>
<td>1</td>
<td>AD17</td>
<td>0000</td>
<td>1</td>
<td>000</td>
</tr>
<tr>
<td>10XX</td>
<td>2</td>
<td>AD18</td>
<td>0001</td>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>18XX</td>
<td>3</td>
<td>AD19</td>
<td>0001</td>
<td>1</td>
<td>000</td>
</tr>
<tr>
<td>20XX</td>
<td>4</td>
<td>AD20</td>
<td>0010</td>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>28XX</td>
<td>5</td>
<td>AD21</td>
<td>0010</td>
<td>1</td>
<td>000</td>
</tr>
<tr>
<td>30XX</td>
<td>6</td>
<td>AD22</td>
<td>0011</td>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>39XX</td>
<td>7</td>
<td>AD23</td>
<td>0011</td>
<td>1</td>
<td>001</td>
</tr>
<tr>
<td>40XX</td>
<td>8</td>
<td>AD24</td>
<td>0100</td>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>48XX</td>
<td>9</td>
<td>AD25</td>
<td>0100</td>
<td>1</td>
<td>000</td>
</tr>
<tr>
<td>50XX</td>
<td>A</td>
<td>AD26</td>
<td>0101</td>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>58XX</td>
<td>B</td>
<td>AD27</td>
<td>0101</td>
<td>1</td>
<td>000</td>
</tr>
<tr>
<td>60XX</td>
<td>C</td>
<td>AD28</td>
<td>0110</td>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>68XX</td>
<td>D</td>
<td>AD29</td>
<td>0110</td>
<td>1</td>
<td>000</td>
</tr>
<tr>
<td>70XX</td>
<td>E</td>
<td>AD30</td>
<td>0111</td>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>78XX</td>
<td>F</td>
<td>AD31</td>
<td>0111</td>
<td>1</td>
<td>000</td>
</tr>
</tbody>
</table>

Older Devices used AD16-31.
Where to get more information

- **PCI Local Bus Specification (PCI SIG)**
  - PCI Special Interest Group (CC:Mail PCI_SIG)
- **PCI System Architecture (Shanley & Anderson)**
  - Mindshare (ISBN 0-201-40993-3)
- **The Indispensable PC Hardware Book (Messmer)**
  - Addison-Wesley (ISBN 0-201-87697-3)
- **PCI Hardware and Software (Solari & Willse)**
  - Annabooks (ISBN 0-929392-28-0)
PCI Bus Glossary

○ Agent
  • Each PCI device, whether a bus master (initiator) or a target is referred to as a PCI Agent.
  • A logical entity that operates on a computer bus. The term applies collectively to functions of a bus master or a bus slave, or to a combination of both.

○ Bridge
  • The logic that connects one computer to another, allowing an agent on one bus to access an agent on the other.
  • Examples would be the bridge between the PCI bus and a standard expansion bus (e.g. ISA bus)
PCI Bus Glossary

- **Burst transfer**
  - The basic bus transfer mechanism of PCI. A burst is comprised of an address phase and one or more data phases.

- **Bus Commands**
  - Signals used to indicate to a target the type of transaction the master is requesting.
  - During the address phase of a PCI transaction, the initiator broadcasts a command on the C/BE bus.
    - Memory Read/Write
    - I/O Read/Write
    - Configuration Read/Write
PCI Bus Glossary

- **Bus device** - A bus device can be either a bus master or target:
  - **MASTER**
    - An agent which has an ability to obtain control of the interface and perform memory or I/O reads and writes to system resources.
    - Drives the address phase, and transaction boundary (FRAME#). The master initiates a transaction, drives the data handshaking signal (IRDY#) with the target.
  - **TARGET** - Claims the transaction by asserting DEVSEL# and handshakes the transaction (TRDY#) with the initiator.
PCI Bus Glossary

- **Configuration address space**
  - A set of 64 registers (DWORDS) used for configuration, initialization, and catastrophic error handling. This address space consists of two regions: a header region and a device-dependent region.

- **Configuration cycle**
  - Bus cycles used for system initialization and configuration via the configuration address space.

- **Latency timer**
  - A mechanism for ensuring that a bus master does not extend the access latency of other masters beyond a specified value.
PCI Bus Glossary

- **PCI** - Peripheral Component Interconnect.

- **Phase** - One or more clocks in which a single unit of information is transferred, consisting of:
  - An *address phase* (a single address transfer in one clock for a single address cycle and two clocks for a dual address cycle).
  - A *data phase* (one transfer state plus zero or more wait states).

- **Positive Decoding**
  - A method of address decoding in which a device responds to accesses only within an assigned address range. See also subtractive decoding.
PCI Bus Glossary

Subtractive decoding

- The PCI-Expansion bus is designed to claim many transactions not claimed by other devices on the bus.
- A method of address decoding in which a device accepts all accesses not positively decoded by another agent. See also positive decoding.

Target

- An agent that responds (with a positive acknowledgement by asserting DEVSEL#) to a bus transaction initiated by a master.

Transaction

- An address phase plus one or more data phases.
PCI Bus Glossary

- **Turnaround cycle**
  - A bus cycle used to prevent contention when one agent stops driving a signal and another agent begins. A turnaround cycle must last one clock and is required on all signals that may be driven by more than one agent.

- **Wait State**
  - A bus clock in which no transfer occurs.
SUMMARY

WE HAVE DISCUSSED THE FOLLOWING:

- Bus standards: ISA, EISA, VL BUS, & PCI
- PCI Bus Architecture Fundamentals
- Basic PCI bus cycles.
- The Required PCI Bus signals.
- PCI Commands, Parity, Subtractive Decode, and PCI Arbitration
- PCI Configuration Access
  - PCI Configuration Registers
  - PCI Configuration Mechanism Type -1
  - PCI ID Select Line usage