PCI Bus Architecture

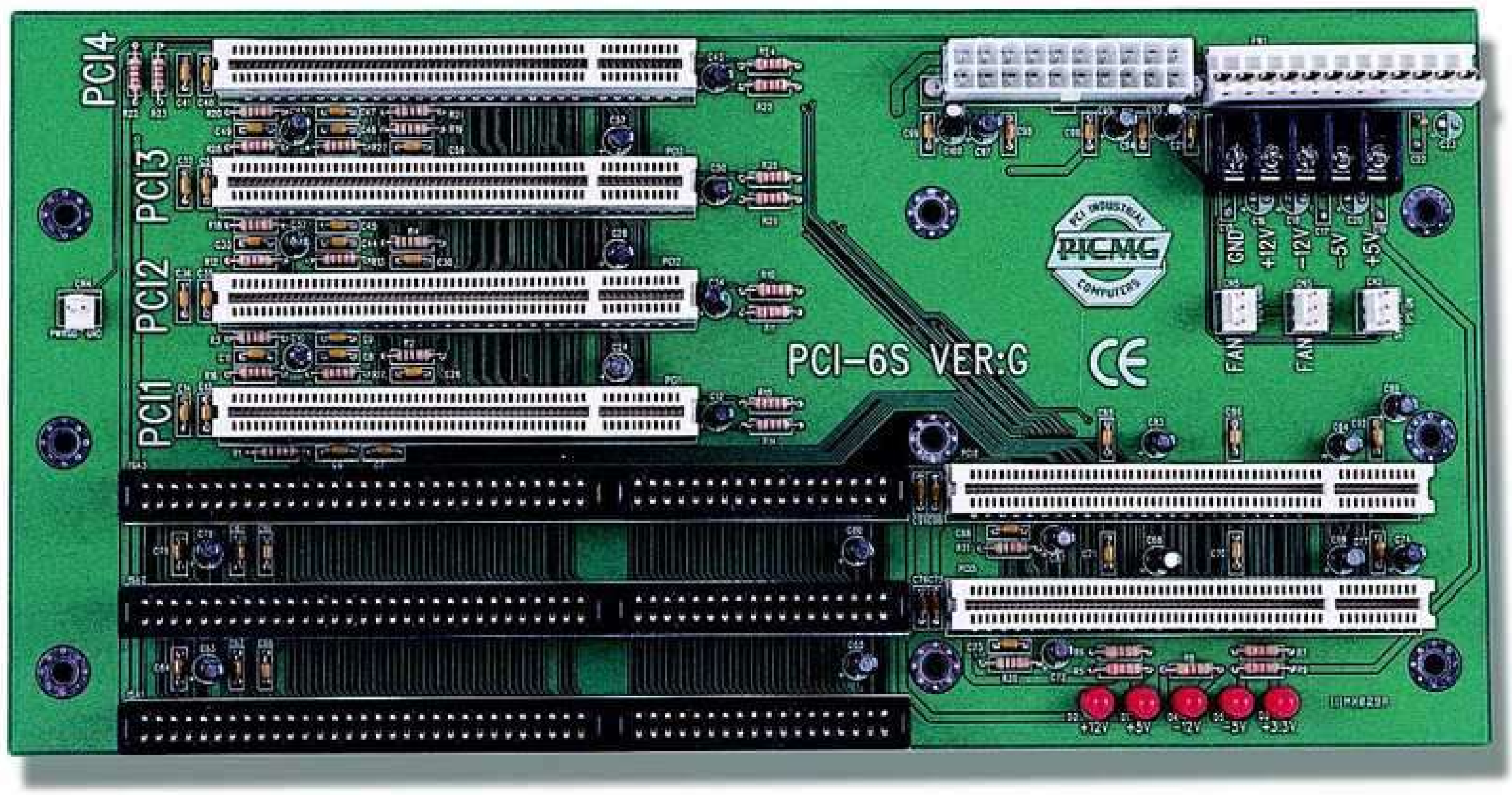
Introduction

- Introduction
- History of the Bus
- Performance
- Plug and Play
- How it works
- Other types of the PCI Bus
- Future of the PCI Bus
- Conclusion

Overview

- If you were installing a new sound, network, or video card in your computer in the ’90s, it was probably built to the old PCI standard, which described how to link these devices via physical wires to other parts of your computer.
- PCI was exciting for its time. PCI devices are easier to install than devices built to the previous ISA (Industry Standard Architecture). When an ISA device was plugged into the I/O bus, a computer could access it by communicating over the matching address on the bus. But it was hard to know in advance what devices would respond to a request, where the devices were located in I/O space, if the computer had the correct drivers to interact with the ISA card, or for devices to avoid conflicting with each other.

PCI vs ISA

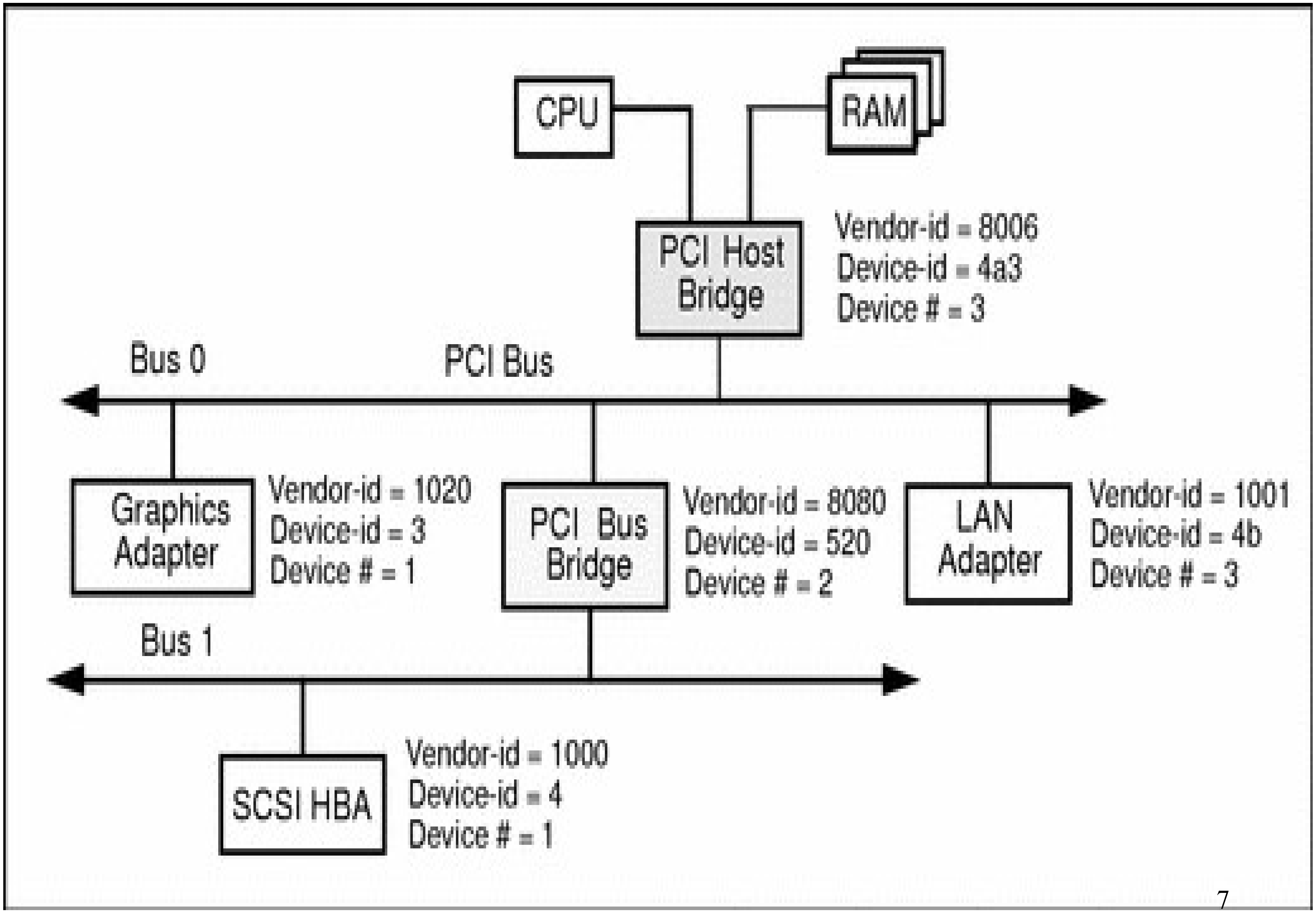

PCI4
PCI3
PCI2
PCI1
PCI-6S VER:G
PICMG
GND
+12V
-12V
-5V
+5V
FAN
FAN
+12V
+5V
-12V
-5V
+3.3V

Introduction of Config. Space

- PCI changed this by introducing the notion of "configuration space," a set of registers on the device area that allows the system to ask the card for information about itself, and respond accordingly.
- PCI also had greater bandwidth than its predecessors, and thus it quickly became ubiquitous. However, as time went on, PCI's limitations became more apparent:
  - The speed of a set of PCI devices was limited to that of the slowest device on the bus. Connecting one outdated peripheral would slow down all devices.
  - At the same time, demand for speed was at an all-time high, as gigabit ethernet became widespread, and PCI devices couldn't keep up.

Config Space

Device ID		Vendor ID	
Status		Command	
Class Code			Revision ID
BIST	Header Type	Lat. Timer	Cache Line S
Base Address Registers			
Cardbus CIS Pointer			
Subsystem ID		Subsystem Vendor ID	
Expansion ROM Base Address			
Reserved			Cap. Pointer
Reserved			
Max Lat.	Min Gnt.	Interrupt Pin	Interrupt Lin


CPU
RAM
PCI Host Bridge
Vendor-id = 8006
Device-id = 4a3
Device # = 3
Bus 0
PCI Bus
Graphics Adapter
Vendor-id = 1020
Device-id = 3
Device # = 1
PCI Bus Bridge
Vendor-id = 8080
Device-id = 520
Device # = 2
LAN Adapter
Vendor-id = 1001
Device-id = 4b
Device # = 3
Bus 1
SCSI HBA
Vendor-id = 1000
Device-id = 4
Device # = 1

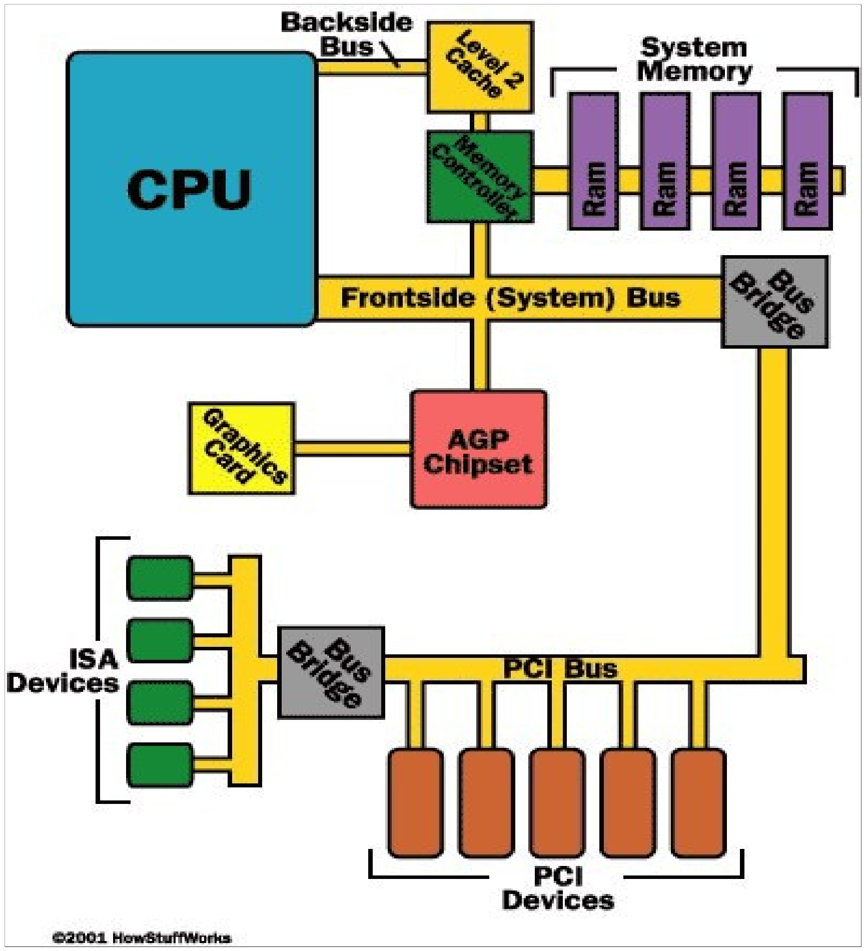
Introduction

- A computer bus is used to transfer data from one location or device on the motherboard to the central processing unit where all calculations take place.
- Two different parts of a Bus
  - Address bus-transfers information about where the data should go
  - Data bus-transfers the actual data

History

- PCI(Peripheral Component Interconnect) bus is based on ISA (Industry Standard Architecture) Bus and VL (VESA Local) Bus.
- Introduced by Intel in 1992
- Revised twice into version 2.1 which is the 64-bit standard that it is today.
- Great feature of PCI Bus was that it was invented as an industry standard
- PCI provides direct access to system memory for the devices that are connected to the bus which is then connected through a bridge that connects to the front side bus.
- This configuration allowed for higher performance without slowing down the processor

History



•The PCI Bus was originally 33Mhz and then changed to 66Mhz.

•PCI Bus became big with the release of Windows 95 with "Plug and Play" technology

•"Plug and Play" utilized the PCI bus concept.

PCI System Bus Performance

- What makes the PCI bus one of the fastest I/O bus used today?
- Three features make this possible:
  - Burst Mode: allows multiple sets of data to be sent (Kozierok, 2001a)
  - Full Bus Mastering: the ability of devices on the PCI bus to perform transfers directly (Kozierok, 2001c)
  - High Bandwidth Options: allows for increased speed of the PCI (Kozierok, 2001a)

How PCI Compares to Other Buses

Bus Type	**Bus Width**	**Bus Speed**	**MB/sec**	**Advantages**	**Disadvantages**
ISA	16 bits	8MHz	16 MBps	low cost, compatibility, widely used	low speed, Jumpers & DIP switches. becoming obsolete
PCI	64 bits	133 MHz	1 GBps	very high speed, Plug & Play, dominant board-level bus	incompatible with older systems, can cost more
CompactPCI	64 bits	33MHz	132 MBps	designed for industrial use, hot swapping/Plug & Play, ideal for embedded systems	lower speed than PCI, need adapter for PC use, incompatible with older systems

Table 1: How PCI compares to other buses (Tyson, 2004a; Quatech, 2004c)

Plug and Play

- Requirements for full implementation:
  - Plug and Play BIOS
  - Extended System Configuration Data (ESCD)
  - Plug and Play operating system

- Tasks it automates:
  - Interrupt Requests (IRQ)
  - Direct Memory Access (DMA)
  - Memory Addresses
  - Input/Output (I/O) Configuration

(Tyson, 2004b)

How PCI Works: Installing A New Device

- Once a new device has been inserted into a PCI slot on the motherboard
  1. Operating System Basic Input/Output System (BIOS) initiates Plug and Play (PnP) BIOS.
  2. PnP BIOS scans the PCI bus for any new hardware connected to the bus. If new hardware is found, it will ask for identification.
  3. The device will respond with its identification and send its device ID to the BIOS through the bus.
  4. PnP checks the Extended System Configuration Data (ESCD) to make sure the configuration data already exists for the card. (If the card is new, then there will be no data for it.)

New Device Cont…

5. PnP will assign an Interrupt Request Line, Direct Memory Access, memory address and Input/Output settings to the card, then stores the information in the ESCD.
6. When the Windows software loads, it will check the PCI bus and the ESCD to see if there is new hardware. Windows will alert the user that new hardware has been found if there is new hardware installed and will also identify the hardware.
7. Windows will determine the device and attempt to install its driver. The operating system may ask the user to insert a disk containing the driver or direct it to where the driver is located. In the event that Windows is unable to determine what the device is, it will provide a dialog window so the user can identify the hardware and load its driver.

How a Device Works

- Example: PCI-based sound card

1. The sound card will convert the analog signal to a digital signal.
2. The digital audio data carried across the PCI bus to the bus controller, which determines which device on the PCI device has the priority to send data to the central processing unit (CPU) and whether the data will go directly to the CPU or to the system memory.
3. If the sound card is in recording mode, the bus controller will assign a high priority to the data coming from the sound card. It will send the sound cards data over the bus bridge to the system bus.
4. The system bus will save the data in system memory. When the recording is complete, then it will be up to the user to save the data from the sound card on either the hard drive, or will remain in memory for additional processing.

Other Types of PCI

- Original PCI
- PCI 2.3
- PCI-X
  - PCI-X 2.0 (second revision)
- PCI Express

Figure: PCI-X 2.0 card

(http://ch272.thinkquest.hostcenter.ch/cgi-local/community.pl?action=pc_theme&lang=ge)

Future of PCI: Requirements

- Support multiple market segments
- Backwards compatible
- Scalable performance
- Advanced features including QoS, power management, and data integrity

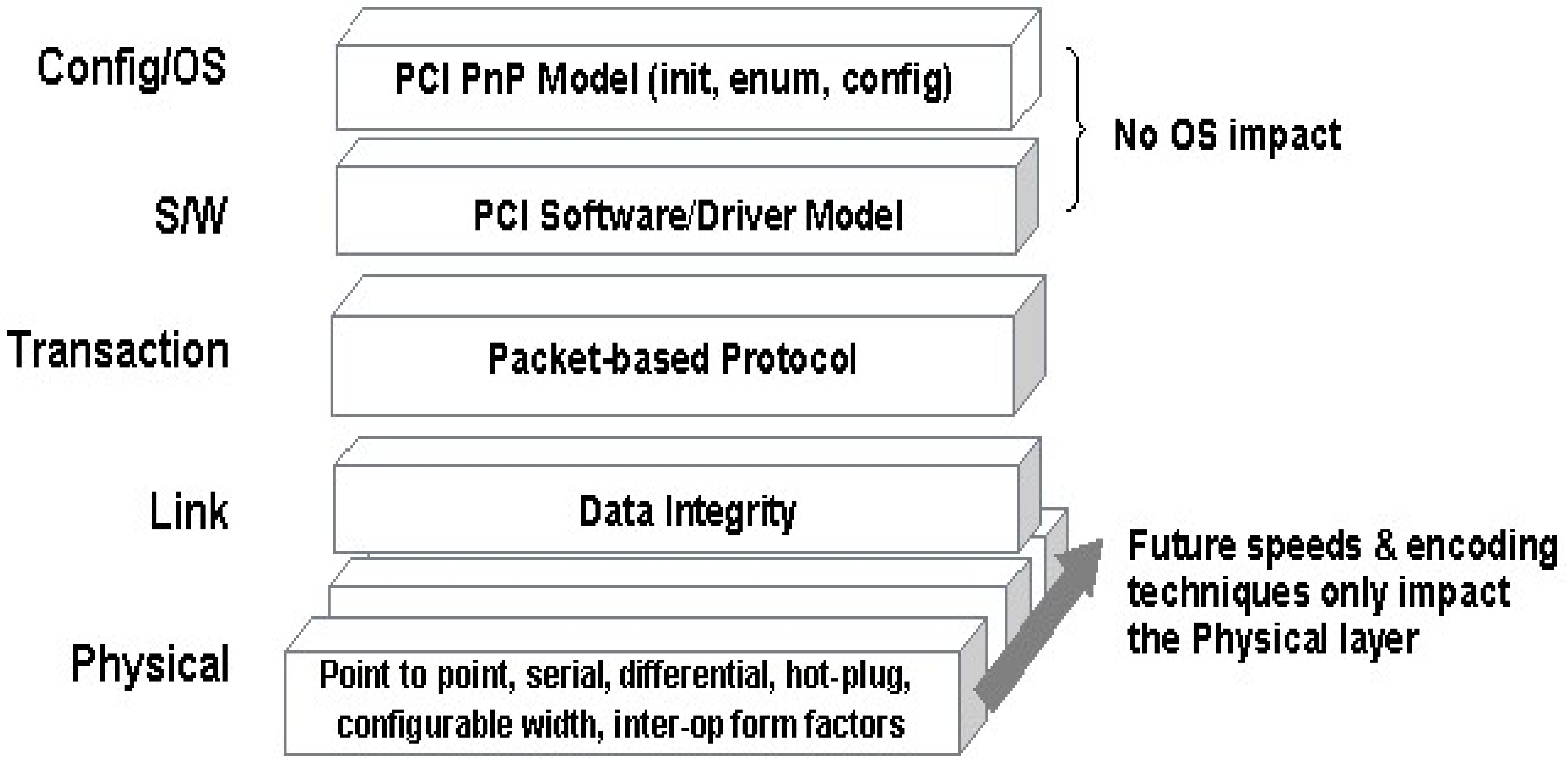
PCIe

- To take on the limitations of PCI, PCIe needed to tackle issues with bus sharing and bus contention.
- When multiple devices were on the same PCI bus, PCI was forced to "clock down" and match the speed of the slowest device on the bus.
- PCIe uses some electrical cleverness to address this. By encoding data using the 8b/10b line code, PCIe is able to encode both data and clock information in a single signal, removing the need for an external clock and greatly increasing potential bandwidth. (Newer versions of PCIe have continued to improve on this; PCIe 3.0 and onward encode with 128b/130b and enable even faster transfer rates.)

PCIe as Network

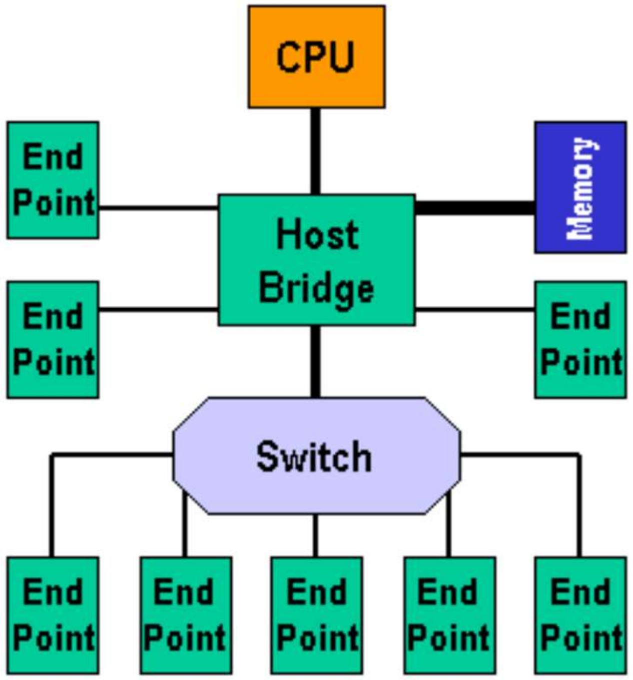
- Additionally, PCIe needed a way to handle high-speed data streaming. If a device didn’t have enough buffering to contain all the data it wanted—as is the case with, say, high-definition video—then naturally you’d like the device to continuously stream the data. However, while that device was busily streaming a huge amount of data, it would put a stranglehold on the bus, preventing all the other devices from doing anything. PCIe resolves this issue by allowing for packet fragmentation, breaking up the data stream into smaller packets that can be transmitted via the transaction layer protocol that underlies the PCIe fabric.
- Thus, it’s easier to think of PCIe as a network, rather than a physical bus. Each device has an address, and the spec describes functionality for flow control, error detection, and retransmissions, none of which existed in PCI.

PCI Express Solution


Config/OS
PCI PnP Model (init, enum, config)
S/W
PCI Software/Driver Model
No OS impact
Transaction
Packet-based Protocol
Link
Data Integrity
Physical
Point to point, serial, differential, hot-plug, configurable width, inter-op form factors
Future speeds & encoding techniques only impact the Physical layer

Advanced Switching with PCI express

- Signals take place at link level
- Allows for QoS and fan out capabilities
- Utilizes system bandwidth



Conclusion

- Due to the need for growing data transfer rates among IO devices, the original PCI Architecture has become outdated
- A new model of PCI, called PCI Express will replace the dated architecture giving it life for another decade

Lab Exercise

$ lspci | cut -d: -f1-2

```
0000:00:00.0 Host bridge
0000:00:00.1 RAM memory
0000:00:00.2 RAM memory
0000:00:02.0 USB Controller
0000:00:04.0 Multimedia audio controller
0000:00:06.0 Bridge
0000:00:07.0 ISA bridge
0000:00:09.0 USB Controller
0000:00:09.1 USB Controller
0000:00:09.2 USB Controller
0000:00:0c.0 CardBus bridge
0000:00:0f.0 IDE interface
0000:00:10.0 Ethernet controller
0000:00:12.0 Network controller
0000:00:13.0 FireWire (IEEE 1394)
0000:00:14.0 VGA compatible controller
```

$ cat /proc/bus/pci/devices | cut -f1

```
0000
0001
0002
0010
0020
0030
```

Cont.

```
$ tree /sys/bus/pci/devices/
/sys/bus/pci/devices/
|-- 0000:00:00.0 -> ../../../devices/pci0000:00/0000:00:00.0
|-- 0000:00:00.1 -> ../../../devices/pci0000:00/0000:00:00.1
|-- 0000:00:00.2 -> ../../../devices/pci0000:00/0000:00:00.2
|-- 0000:00:02.0 -> ../../../devices/pci0000:00/0000:00:02.0
|-- 0000:00:04.0 -> ../../../devices/pci0000:00/0000:00:04.0
|-- 0000:00:06.0 -> ../../../devices/pci0000:00/0000:00:06.0
|-- 0000:00:07.0 -> ../../../devices/pci0000:00/0000:00:07.0
|-- 0000:00:09.0 -> ../../../devices/pci0000:00/0000:00:09.0
|-- 0000:00:09.1 -> ../../../devices/pci0000:00/0000:00:09.1
|-- 0000:00:09.2 -> ../../../devices/pci0000:00/0000:00:09.2
|-- 0000:00:0c.0 -> ../../../devices/pci0000:00/0000:00:0c.0
|-- 0000:00:0f.0 -> ../../../devices/pci0000:00/0000:00:0f.0
|-- 0000:00:10.0 -> ../../../devices/pci0000:00/0000:00:10.0
|-- 0000:00:12.0 -> ../../../devices/pci0000:00/0000:00:12.0
|-- 0000:00:13.0 -> ../../../devices/pci0000:00/0000:00:13.0
`-- 0000:00:14.0 -> ../../../devices/pci0000:00/0000:00:14.0
```

End of Presentation

Any Questions?