PCI and PCI Express Bus Architecture

Computer Science & Engineering Department
Arizona State University
Tempe, AZ 85287

Dr. Yann-Hang Lee
yhlee@asu.edu
(480) 727-7507
Buses in PC-XT and PC-AT

- **ISA (Industry Standard Architecture)**
  - IBM-PC and PC-XT: 8 bits at 4.77MHz, directly connect to 8088, 2-stage bus cycle (2.38Mbyte/sec bus bandwidth)
  - AT bus: extension slot + 8 bit ISA
    - 16 bits at 8.33MHz for 80286
Buses in PC(486)

- 16-bit ISA cannot support Window applications --- video data
- VESA LB (local bus) -- linked to 486 local bus, 33MHZ, 32 bits
Buses in PC (Pentium)

- **Backside Bus**
- **Frontside Bus**
- **PCI**
  - Direct access to system memory for connected devices
  - Uses a bridge to connect to the frontside bus and therefore to the CPU
- **ISA**
Advantages and Disadvantages of Buses

- **Versatility:**
  - New devices can be added easily
  - Peripherals can be moved between computer systems that use the same bus standard

- **Low Cost:**
  - A single set of wires is shared in multiple ways

- **Manage complexity by partitioning the design**

- **It creates a communication bottleneck**
  - The bandwidth of that bus can limit the maximum I/O throughput

- **The maximum bus speed is largely limited by:**
  - The length of the bus and the number of devices on the bus
  - The need to support a range of devices with varying latencies and data transfer rates
Control lines: Signal requests and acknowledgments

Data/address lines carry information between the source and the destination:

A bus transaction includes three parts:
- Arbitration – which master can use the bus
- Issuing the command (and address) – request
- Transferring the data – action

Master is the one who starts the bus transaction by:
- issuing the command (and address)

Slave is the one who responds to the command by:
- Sending data to the master if the master asks for data
- Receiving data from the master if the master wants to send data
Types of Buses

- **Processor-Memory Bus (design specific)**
  - Short and high speed
  - Only need to match the memory system
    - Maximize memory-to-processor bandwidth
  - Connects directly to the processor
  - Optimized for cache block transfers

- **I/O Bus (industry standard)**
  - Usually is lengthy and slower
  - Need to match a wide range of I/O devices
  - Connects to the processor-memory bus or backplane bus

- **Backplane Bus (standard or proprietary)**
  - Backplane: an interconnection structure within the chassis
  - Allow processors, memory, and I/O devices to coexist
  - Cost advantage: one bus for all components
Synchronous and Asynchronous Bus

- **Synchronous Bus:**
  - Includes a clock in the control lines
  - A fixed protocol for communication that is relative to the clock
  - Advantage: involves very little logic and can run very fast
  - Disadvantages:
    - Every device on the bus must run at the same clock rate
    - To avoid clock skew, they cannot be long if they are fast

- **Asynchronous Bus:**
  - It is not clocked
  - It can accommodate a wide range of devices
  - It can be lengthened without worrying about clock skew
  - It requires a handshaking protocol
Simple Synchronous Protocol

- All agents operate synchronously – all source / sink data at same rate
- A simple protocol to manage the source and target
- Even memory busses are more complex than this
  - memory (slave) may take time to respond
  - it needs to control data rate
Asynchronous Handshake

- A read transaction

- t0: Master has obtained control and asserts address, direction, data, waits a specified amount of time for slaves to decode target

- t1: Master asserts request line

- t2: Slave asserts ack, indicating ready to transmit data

- t3: Master releases req, data received

- t4: Slave releases ack
Multiple Bus Masters: the Need for Arbitration

- To obtain access to the bus

- Bus arbitration scheme:
  - A bus master wanting to use the bus asserts the bus request
  - A bus master cannot use the bus until its request is granted
  - A bus master must signal to the arbiter after it finishes using the bus

- Bus arbitration schemes usually try to balance two factors:
  - Bus priority
  - Fairness and starvation

- Bus arbitration schemes can be divided into four broad classes:
  - Daisy chain arbitration: single device with all request lines.
  - Centralized, parallel arbitration
  - Distributed arbitration by self-selection: each device wanting the bus places a code indicating its identity on the bus.
  - Distributed arbitration by collision detection: Ethernet uses this.
Increasing the Bus Bandwidth

- **Separate versus multiplexed address and data lines:**
  - Address and data can be transmitted in one bus cycle if separate address and data lines are available.
  - Cost: (a) more bus lines, (b) increased complexity.

- **Data bus width:**
  - By increasing the width of the data bus, transfers of multiple words require fewer bus cycles.
  - Example: SPARCstation 20’s memory bus is 128 bit wide.
  - Cost: more bus lines.

- **Block transfers:**
  - Allow the bus to transfer multiple words in back-to-back bus cycles.
  - Only one address needs to be sent at the beginning.
  - The bus is not released until the last word is transferred.
  - Cost: (a) increased complexity, (b) decreased response time for request.
Increasing Bus Transaction Rate

- **Overlapped operations (pipelined)**
  - perform arbitration for next transaction during current transaction
  - initiate next address phase during current data phase

- **Bus parking**
  - master holds onto bus and performs multiple transactions as long as no other master makes request

- **Split-phase (or packet switched) bus**
  - completely separate address and data phases
  - arbitrate separately for each
  - address phase yield a tag which is matched with data phase

- "All of the above" in most modern processor-memory busses
PCI Bus

- **Release 2.1 -- 66MHz, 32-bit and 64-bit connectors.**
  - 3.3V or 5V based on PCI chip set’s buffer/drivers

![Diagram](image)

- **Agent, bus master (initiator) and slave (target)**

- **Bus transaction:**
  - bus masters issue requests ⇔ arbitration ⇔ bus grant
  - issues address and command and begins a cycle frame (transaction)
    - memory, I/O, configuration read/write commands
  - a target is selected (device select)
  - it is ready to complete the data transfer phase
PCI Bus Operation

- **Address phase**
  - At the same time, initiator identifies target device and the type of transaction
  - The initiator asserts the FRAME# signal
  - Every PCI target device latches the address and decodes it

- **Data Phase**
  - Number of data bytes to be transferred is determined by the number of Command/Byte Enable signals asserted by initiator
  - Both of initiator and target must be ready to complete data phase
  - IRDY# and TRDY# used

- **Transaction completion and return of bus to idle state**
  - By deasserting the FRAME# but asserting IRDY#
  - When the last data transfer has completed, the initiator returns the PCI bus to idle state by deasserting IRDY#
PCI Read/Write Transactions

- All signals sampled on rising edge
- Centralized Parallel Arbitration
  - overlapped with previous transaction
- All transfers are (unlimited) bursts

- Address phase starts by asserting FRAME#
- Next cycle “initiator” asserts cmd and address
- Data transfers happen when
  - IRDY# asserted by master when ready to transfer data
  - TRDY# asserted by target when ready to transfer data
  - transfer when both asserted on rising edge
- FRAME# deasserted when master intends to complete only one more data transfer
PCI Bus Signals

A typical PCI read transaction
PCI Lines (1)

- **CLK** – PCI input clock
  - All signals sampled on rising, allowed to vary from 0 to 33 MHz

- **RST#** -- asynchronous reset
  - PCI device must tri-state all I/Os during reset

- **TRDY#** –
  - When the target asserts this signal, it tells the initiator that it is ready to send or receive data

- **STOP#** –
  - Used by target to indicate that it needs to terminate the transaction

- **DEVSEL#** – Device select
  - When a target recognizes its address, it asserts DEVSEL# to claim the corresponding transaction

- **FRAME#** – Signals the start and end of a transaction

- **IRDY#** –
  - Assertion by initiator indicates that it is ready to send receive data
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, for example, C/BE#[0] is the byte enable for AD[7:0]

- **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)
Arbitration and Error Signals

- **REQ# – O**
  - Asserted by initiator to request bus ownership
  - Point-to-point connection to arbiter – each initiator has its own REQ# line

- **GNT# – I**
  - Asserted by system arbiter to grant bus ownership to the initiator
  - Point-to-point connection from arbiter – each initiator has its own GNT# line

- **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, such as address parity error
  - May invoke NMI (non-maskable interrupt, i.e., a restart) in some systems
Example – Basic Write

- A four-DWORD burst from an initiator to a target
- Addressing, handshaking, and data transfer phases
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 2\textsuperscript{nd}, 3\textsuperscript{rd}, and last data are transferred one cycle each (”1-1-1”)

- If the profile is 5-1-1-1
  - Medium decode – DEVSEL\# asserted on 2\textsuperscript{nd} 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
  - Total of 4 data phases, but required 8 clocks
    - Only 50% efficiency
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#
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
  - Wait states may be inserted dynamically by the initiator or target
    - Target deasserts TRDY# to signal it is not ready
    - Initiator deasserts IRDY# to signal it is not ready

- **Target termination**
  - 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

- 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%
- 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!
Basic Write Transaction

(PCI Local Bus Specification, Revision 2.2)
PCI Optimizations and Additional Features

- **Push bus efficiency toward 100% under common simple usage**
- **Bus parking**
  - retain bus grant for previous master until another makes request
  - granted master can start next transfer without arbitration
- **Arbitrary burst length**
  - initiator and target can exert flow control with xRDY
  - discount with STOP (abort or retry, by target), FRAME (by master) and GNT (by arbiter)
- **Delayed (pended, split-phase) transactions**
  - free the bus after request to slow device
- **Additional Features**
  - Interrupts: support for controlling I/O devices
  - Cache coherency: support for I/O and multiprocessors
  - Locks: support timesharing, I/O, and MPs
  - Configuration Address Space (plug and play)
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
Accessing the Address Spaces

- **Memory space (4GB)** accessed using a large variety of processor instructions (mov, add, or, shr, push, etc.) and virtual-to-physical address-translation.

- **PCI configuration space (16MB)** accessed only by using the processor’s special ‘in’ and ‘out’ instructions (without any translation of port-addresses).

- **I/O space (64KB)**

  - *i/o-ports 0xCF8-0xCFFF dedicated to accessing PCI Configuration Space*

  (www.cs.usfca.edu/~cruse/cs336s09/lesson19.ppt)
PCI Configuration Address Space

Contains 256 bytes of basic device information,
- addressable by 8-bit PCI bus, 5-bit device, and 3-bit function numbers for the device
- the first 64 bytes (00h – 3Fh) make up the standard configuration header, including PCI ID, i.e. vendor ID and device ID registers, to identify the device
- the remaining 192 bytes (40h – FFh) represent user-definable configuration space, such as the information specific to a PC card for use by its accompanying software driver

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
Memory and IO Spaces

- Memory 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

- IO 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
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

<table>
<thead>
<tr>
<th>C/BE[3::0]#</th>
<th>Command Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Interrupt Acknowledge</td>
</tr>
<tr>
<td>0001</td>
<td>Special Cycle</td>
</tr>
<tr>
<td>0010</td>
<td>I/O Read</td>
</tr>
<tr>
<td>0011</td>
<td>I/O Write</td>
</tr>
<tr>
<td>0100</td>
<td>Reserved</td>
</tr>
<tr>
<td>0101</td>
<td>Reserved</td>
</tr>
<tr>
<td>0110</td>
<td>Memory Read</td>
</tr>
<tr>
<td>0111</td>
<td>Memory Write</td>
</tr>
<tr>
<td>1000</td>
<td>Reserved</td>
</tr>
<tr>
<td>1001</td>
<td>Reserved</td>
</tr>
<tr>
<td>1010</td>
<td>Configuration Read</td>
</tr>
<tr>
<td>1011</td>
<td>Configuration Write</td>
</tr>
<tr>
<td>1100</td>
<td>Memory Read Multiple</td>
</tr>
<tr>
<td>1101</td>
<td>Dual Address Cycle</td>
</tr>
<tr>
<td>1110</td>
<td>Memory Read Line</td>
</tr>
<tr>
<td>1111</td>
<td>Memory Write and Invalidate</td>
</tr>
</tbody>
</table>
The Plug-and-Play Concept

- 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
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
Type 00h Configuration Space Header

(PCI Local Bus Specification, Revision 2.2)
Configuration Commands

- Two DWORD I/O locations are used to generate configuration transactions
  - 0CF8h references a read/write register, CONFIG_ADDRESS.
  - 0CFCh references a read/write register, CONFIG_DATA.

- Bus enumeration
  - attempting to read the Vendor- and Device ID register for each combination of bus number and device number, at the device's function #0
  - knows a device exists, and can then program the memory mapped and I/O port addresses for the device.
PCI Challenges

- **Limited Bandwidth**
  - PCI-X and Advanced Graphics Port (AGP) for higher frequency
  - Reduction of distance
- **Bandwidth shared between all devices**
- **Limited host pin-count**
- **Lack of support for real time data transfer**
- **Stringent routing rules**
- **Lack of scaling with frequency and voltage**
- **Absence of power management**
- **PCI-X** -- an enhancement of the 32-bit PCI Local Bus for a higher bandwidth demand.
  - a double-wide version of PCI, running at up to four times the clock speed
Inter-Networking Driving Demand

- Multimedia applications drive the need for fast, efficient processing of data over wired or wireless media.
- CPU performance doubles about every 18 months while PC Bus performance doubles about every 3 years.

![Graph showing relative bandwidth over time](image)
PCI Express Basics

- Serial, point-to-point, Low Voltage Differential Signaling
- 2.5GHz full duplex lanes (2.5Gb/s)
  - PCIe Gen 2 = 5Gb/s
- Scaleable links – x1, x4, x8, x16
- Packet based transaction protocol
- Software compatible but with higher speeds
- Built-in Quality of Service provisions
  - Virtual Channels
  - Traffic Classes
- Reliability, Availability and Serviceability
  - End-to-End CRC (Cyclic redundant checking)
  - Poison Packet
  - Native Hot Plug support
- Flow Control
- Advance error reporting
## PCI Express Performance

<table>
<thead>
<tr>
<th>Link Width</th>
<th>X1</th>
<th>X2</th>
<th>X4</th>
<th>X8</th>
<th>X12</th>
<th>X16</th>
<th>x32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth in Gbits/s (Tx and Rx)</td>
<td>5</td>
<td>10</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>160</td>
</tr>
<tr>
<td>Throughput in GB/s (Tx and Rx)</td>
<td>.5</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>Throughput in GB/s (per direction)</td>
<td>.25</td>
<td>.5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

**Raw:** Assuming 100% efficiency with no payload overhead.

- □ = PCI 32/66
- ▪ = PCI or PCI-X 64/66
- ◼ = PCI-X 64/133
PCIe Layers

- **Layered architecture**
- **Application Data transferred via packets**
  - Transaction Layer Packet (TLP)
- **PCIe core usually implement the lower three layers**
- **Protocol handling**
  - connection establishing
  - link control
  - flow control
  - power management
  - error detection and reporting

(http://www.cast-inc.com/company/events/shows/date06/cast_pcie-ocp_slides.pdf)
PCIe TLP Structure

Information in core section of TLP comes from Software Layer / Device Core

Created by Transaction Layer

Appended by Data Link Layer

Appended by Physical Layer

Example PCI Express Topology – Root & Switch

Transaction Types, Address Spaces

- Request are translated to one of four transaction types by the Transaction Layer:
  - Memory Read or Memory Write. Used to transfer data from or to a memory mapped location
    ➢ also supports a locked memory read transaction variant.
  - I/O Read or I/O Write. Used to transfer data from or to an I/O location
    ➢ restricted to supporting legacy endpoint devices.
  - Configuration Read or Configuration Write – Used to discover device capabilities, program features, and check status in the 4KB PCI Express configuration space.
  - Messages. Handled like posted writes. Used for event signaling and general purpose messaging.
## Transaction Layer Packet Types

<table>
<thead>
<tr>
<th>Description</th>
<th>Abbreviated Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Read Request</td>
<td>MRd</td>
</tr>
<tr>
<td>Memory Read Request – Locked Access</td>
<td>MRdLk</td>
</tr>
<tr>
<td>Memory Write Request</td>
<td>MWr</td>
</tr>
<tr>
<td>IO Read Request</td>
<td>IORd</td>
</tr>
<tr>
<td>IO Write Request</td>
<td>IOWr</td>
</tr>
<tr>
<td>Configuration Read Request Type 0 and Type 1</td>
<td>CfgRd0, CfgRd1</td>
</tr>
<tr>
<td>Configuration Write Request Type 0 and Type 1</td>
<td>CfgWr0, CfgWr1</td>
</tr>
<tr>
<td>Message Request without Data Payload</td>
<td>Msg</td>
</tr>
<tr>
<td>Message Request with Data Payload</td>
<td>MsgD</td>
</tr>
<tr>
<td>Completion without Data (used for IO, configuration write completions and read completion with error completion status)</td>
<td>Cpl</td>
</tr>
<tr>
<td>Completion with Data (used for memory, IO and configuration read completions)</td>
<td>CplD</td>
</tr>
<tr>
<td>Completion for Locked Memory Read without Data (used for error status)</td>
<td>CplLk</td>
</tr>
<tr>
<td>Completion for Locked Memory Read with Data</td>
<td>CplDLk</td>
</tr>
</tbody>
</table>


Real-Time Systems Lab, Computer Science and Engineering, ASU
Example TLP Request Header Formats

- Memory transaction (32 bits address)

- Configuration transaction

Real-Time Systems Lab, Computer Science and Engineering, ASU
Programmed I/O Transaction

Requester:
- Step 1: Root Complex (requester) initiates Memory Read Request (MRd)
- Step 4: Root Complex receives CplID

Completer:
- Step 2: Endpoint (completer) receives MRd
- Step 3: Endpoint returns Completion with data (CplID)

Incoming Requests

- Incoming requests perform local subsystem read or write
- Some incoming requests require sending completion TLP
- Completion TLP rules
  - Must form completion packets with respect to Max_Payload and Read Completion Boundary
  - Must correctly encode fields in completion TLP
  - Completion address in packet differs (I/O x Memory)
- Application must correctly report a request processing problem to the core

(http://www.cast-inc.com/company/events/shows/date06/cast_pcie-ocp_slides.pdf)
Outgoing Requests

- Outgoing Requests are generated by the application
- There is a set of rules for forming outgoing request TLP
  - Must be identified by unique Tag
  - Read requests restricted by Max_Read_Request_Size
  - Write requests restricted by Max_Payload
  - Must not cross 4kB address boundary
- Violations will result in request being discarded and error detected
- Completion request processing
  - Completions for multiple outstanding requests must be processed by Tag
  - Must have correct values in lower addresses to process multiple TLPs
  - Must process both Unsupported Request and Completer Abort responses