PCI-SIG®
Architecture Overview

Richard Solomon
IC Design Engineer
LSI Logic
What’s all this PCI stuff anyway?

- Presentation will cover basic concepts and their evolution from PCI through PCI-X® to PCI Express®
  - Specs written assuming designers have these key background concepts
  - High level overview of PCI, PCI-X, PCI Express

- Day 1 will progress through:
  - PCI Express Protocols…
  - …PCI Express 2.0 5GT/sec Electricals…
  - …and will close with exciting new work happening in the area of I/O Virtualization

- Day 2 devoted to members like you sharing their experiences implementing PCI-SIG Technologies
PCI Background
PCI

- Revolutionary
  - Plug and Play jumperless configuration (BARs)
  - Unprecedented bandwidth
    - 32-bit / 33MHz – 133MB/sec
    - 64-bit / 66MHz – 533MB/sec
  - Designed from day 1 for bus-mastering adapters

- Evolutionary
  - System BIOS maps devices then operating systems boot and run without further knowledge of PCI
  - PCI-aware O/S could gain improved functionality
PCI-X

- Revolutionary
  - Unprecedented bandwidth
    - Up to 1066MB/sec with 64-bit / 133MHz
  - Registered bus protocol
    - Eased electrical timing requirements
  - Brought split transactions into PCI “world”

- Evolutionary
  - PCI compatible at hardware *AND* software levels
  - PCI-X 266/533 added as “mid-life” performance bump
    - 2133MB/sec at PCI-X 266 and 4266MB/sec at PCI-X 533
Revolutionary AND Evolutionary

- PCI Express (aka PCIe)
  - Revolutionary
    - Unprecedented bandwidth
      - x1: 250MB/sec in *EACH* direction
      - x16: 4000MB/sec in *EACH* direction
    - “Relaxed” electricals due to serial bus architecture
      - Point-to-point, low voltage, dual simplex with embedded clocking
  - Evolutionary
    - PCI compatible at software level
      - Configuration space
      - Power Management
      - Of course, PCIe-aware O/S can get more functionality
    - Transaction layer familiar to PCI/PCI-X designers
    - System topology matches PCI/PCI-X
PCI Concepts
PCI Concepts

- Address spaces
  - Memory – 64-bit
  - I/O – 32-bit (non-burstable since PCI-X)
  - Configuration (“Config”) – Bus/Device/Function
  - PCI Express ECN adds “Trusted Configuration Space”
    - Just a fourth address space from a bus perspective
    - Enables system trust mechanisms

- Key configuration space regs/concepts
  - Base Address Registers (BARs)
    - 64-bit vs 32-bit addressing
  - Linked list of capabilities
Address spaces – Memory & I/O

- Memory space mapped cleanly to CPU semantics
  - 32-bits of address space initially
  - 64-bits introduced via Dual-Address Cycles (DAC)
    - Extra clock of address time on PCI/PCI-X
    - 4DWORD header in PCI Express
  - Burstable

- I/O space mapped cleanly to CPU semantics
  - 32-bits of address space
    - Actually much larger than CPUs of the time
  - Non-burstable
    - Most PCI implementations didn’t support
    - PCI-X codified
    - Carries forward to PCI Express
Address spaces – Configuration

- Configuration space???
  - Allows control of devices’ address decodes without conflict
  - No conceptual mapping to CPU address space
    - Memory-based access mechanisms introduced with PCI-X and PCIe
  - Bus / Device / Function (aka BDF) form hierarchy-based address
    - “Functions” allow multiple, logically independent agents in one physical device.
      - E.g. combination SCSI + Ethernet device
      - 256 bytes or 4K bytes of configuration space per device
    - PCI/PCI-X bridges form hierarchy
    - PCIe switches form hierarchy
      - Look like PCI-PCI bridges to software
  - “Type 0” and “Type 1” configuration cycles
    - Type 0: to same bus segment
    - Type 1: to another bus segment
Configuration Space (cont’d)

- Processor
- Host/PCI Bridge
  - Bus = 0
  - Secondary = 1
  - Subord = 3
- PCI Bus 0
- PCI-to-PCI Bridge
  - Primary = 0
  - Secondary = 1
  - Subord = 3
- PCI Bus 1
- PCI-to-PCI Bridge
  - Primary = 1
  - Secondary = 2
  - Subord = 2
- PCI Bus 2
- PCI Bus 3
- Main Memory
- Host/PCI Bridge
  - Bus = 4
  - Subord = 5
- PCI Bus 4
- PCI Bus 5
- PCI-to-PCI Bridge
  - Primary = 4
  - Secondary = 5
  - Subord = 5
Using Configuration Space

- **Device Identification**
  - VendorID: PCI-SIG assigned
  - DeviceID: Vendor self-assigned
  - Subsystem VendorID: PCI-SIG
  - Subsystem DeviceID: Vendor

- **Address Decode controls**
  - Software reads/writes BARs to determine required size and maps appropriately
  - Memory, I/O, and bus-master enables

- **Other bus-oriented controls**
Memory BARs

- 00 - 32-bit decoder. Locate anywhere in lower 4GB
- 01 - locate below 1MB (reserved in 2.2 spec)
- 10 - 64-bit decoder. Locate anywhere in 2\textsuperscript{32} memory space (implies this register is 64-bits wide and consumes next dword of config space as well as this one).
- 11 - reserved

Memory space indicator

Note upper DWORD not present when 64-bit decode is not set
Using Configuration Space

- I/O BARs look similar to Memory
  - Bit 0 is “1” to indicate I/O
  - No upper DWORD
  - No other encoded bits

![Base Address Diagram](chart)

- Base Address
- Reserved
- I/O Space Indicator

PCI-SIG Developers Conference Europe 2006  Copyright © 2006, PCI-SIG, All Rights Reserved
Using Configuration Space

- Command Register (common fields)

```
15 11 10 9 8 7 6 5 4 3 2 1 0

Reserved

Interrupt Disable
SERR# Enable
Parity Error Response
Bus Master Enable
Memory Space Enable
IO Space Enable
```
Using Configuration Space

- Status Register (common fields)

```
0  7  15  14  13  12  11  10  9  8  7  6  5  4  3
  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
  |-------------------|-------------------|-------------------|
  | Reserved          | Capabilities List |
  | Master Data Parity Error |
  | Signalled Target-Abort |
  | Received Target-Abort |
  | Received Master-Abort |
  | Signalled System Error |
  | Detected Parity Error |
```
Using Configuration Space – Capabilities List

Bit 4

Capabilities List “Head”
Using Configuration Space – Capabilities List (cont’d)

- Linked list
  - Follow the list! Cannot assume fixed location of any given feature in any given device
  - Features defined in their related specs:
    - PCI-X
    - PCIe
    - PCI Power Management
    - Etc…

<table>
<thead>
<tr>
<th>31</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature-specific Configuration Registers</td>
<td>Pointer to Next Capability</td>
<td>Capability ID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dword 0</td>
<td>Dword 1</td>
<td>Dword n</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using Configuration Space – Extended Capabilities List

- PCI Express only
- Linked list
  - Follow the list! Cannot assume fixed location of any given feature in any given device
  - First entry in list is *always* at 100h
  - Features defined in PCI Express specification

<table>
<thead>
<tr>
<th>31</th>
<th>2019</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pointer to Next Capability</td>
<td>Version</td>
<td>Capability ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Feature-specific Configuration Registers

Dword 0
Dword 1
Dword n
What is “Trusted” Configuration Space?

- Trusted Configuration Space (TCS) is an ECN for the PCIe 1.1 Base spec and will be included in PCIe 2.0 Base spec – does NOT apply to PCI or PCI-X
- TCS is a new PCIe address space
- TCS introduces 2 new PCIe Requests
  - Trusted Config Read & Trusted Config Write
  - System will control generation of these requests through means outside the PCIe spec
- TCS is optional for Endpoints, Switches, & Root Ports
  - Newly designed switches should include routing
Interrupts

- PCI introduced INTA#, INTB#, INTC#, INTD# - collectively referred to as INTx
  - Level sensitive
  - Decoupled device from CPU interrupt
  - System controlled INTx to CPU interrupt mapping
  - Configuration registers
    - report A/B/C/D
    - programmed with CPU interrupt number

- PCI Express mimics this via “virtual wire” messages
  - Assert_INTx and Deassert_INTx
MSI & MSI-X Explained
MSI & MSI-X Apply to ALL PCI-SIG Specifications

- Implementation of MSI *or* MSI-X is mandatory in both PCI Express and PCI-X
  - Note recent ECNs allowing MSI-X instead of MSI

- Implementation of either MSI or MSI-X is optional in Conventional PCI

- Subsequent slides apply to any bus implementation of MSI and MSI-X
  - Same structures in PCI, PCI-X, and PCI Express
Once Enabled, MSI or MSI-X Messages Replace INTx

- PCI and PCI-X devices stop asserting INTA, INTB, INTC, INTD once MSI or MSI-X mode is enabled.

- PCI Express devices stop sending Assert_INTx and Deassert_INTx TLPs once MSI or MSI-X mode is enabled.

- NOTE: *Boot devices* and any device intended for a non-MSI operating system generally must still support the appropriate INTx signaling!
MSI

- Message Signaled Interrupts (MSI) is an optional feature that enables a device function to request service by writing a system-specified data value to a system-specified address (using a PCI DWORD memory write transaction).

- System software initializes the message address and message data (referred to as the “vector”) during device configuration, allocating one or more vectors to each MSI-capable function.
MSI and MSI-X Explained

- **MSI-X**
  - MSI-X defines a separate optional extension to basic MSI functionality.
  - Many of the characteristics of MSI-X are identical to those of MSI.
  - MSI-X additional capabilities include,
    - a larger maximum number of vectors per function
    - the ability for software to control aliasing, when fewer vectors are allocated than requested
    - the ability for each vector to use an independent address and data value, specified by a table that resides in Memory Space.
Per-vector masking

- Per-vector masking is managed through a Mask and Pending bit pair per MSI vector or MSI-X Table entry.
- An MSI vector is masked when its associated Mask bit is set.
- An MSI-X vector is masked when its associated MSI-X Table entry Mask bit or the MSI-X Function Mask bit is set.
- While a vector is masked,
  - the function is prohibited from sending the associated message,
  - and the function must set the associated Pending bit whenever the function would otherwise send the message.
MSI and MSI-X Explained

- **MSI-X ECN**
  - A function is permitted to implement both MSI and MSI-X, but system software is prohibited from enabling both at the same time.
  - For the sake of software backward compatibility, MSI and MSI-X use separate and independent capability structures.
  - On functions that support both MSI and MSI-X, system software that supports only MSI can still enable and use MSI without any modification.
### MSI Capability Structure

Capability Structure for 64-bit Message Address and Per-vector Masking

<table>
<thead>
<tr>
<th>Message Control</th>
<th>Next Pointer</th>
<th>Capability ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Address</td>
<td>Capability Pointer</td>
<td>Capability Pointer + 04h</td>
</tr>
<tr>
<td>Message Upper Address (optional)</td>
<td>Capability Pointer + 08h</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Message Data</td>
<td>Capability Pointer + 0Ch</td>
</tr>
<tr>
<td>Mask Bits (optional)</td>
<td></td>
<td>Capability Pointer + 10h</td>
</tr>
<tr>
<td>Pending Bits (optional)</td>
<td></td>
<td>Capability Pointer + 14h</td>
</tr>
</tbody>
</table>
### MSI and MSI-X Explained

#### MSI-X Capability and Table Structures

<table>
<thead>
<tr>
<th>31</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Control</td>
<td>Next Pointer</td>
<td>Capability ID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table Offset</td>
<td></td>
<td>Table BIR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBA Offset</td>
<td></td>
<td>PBA BIR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Different from MSI, the MSI-X capability structure points to an MSI-X Table Structure and a MSI-X Pending Bit Array (PBA) structure, each residing in Memory Space.

Each structure is mapped by a Base Address register (BAR) belonging to the function. A BAR Indicator register (BIR) indicates which BAR, and maps Memory space.
## MSI and MSI-X Explained

### MSI-X Capability and Table Structures

<table>
<thead>
<tr>
<th>DWORD3</th>
<th>DWORD2</th>
<th>DWORD1</th>
<th>DWORD0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector Ctrl</td>
<td>Msg Data</td>
<td>Msg Upper Addr</td>
<td>Msg Address</td>
<td>entry 0  Base</td>
</tr>
<tr>
<td>Vector Ctrl</td>
<td>Msg Data</td>
<td>Msg Upper Addr</td>
<td>Msg Address</td>
<td>entry 1  Base +1*16</td>
</tr>
<tr>
<td>Vector Ctrl</td>
<td>Msg Data</td>
<td>Msg Upper Addr</td>
<td>Msg Address</td>
<td>entry 2  Base +2*16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>......</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>......</td>
</tr>
<tr>
<td>Vector Ctrl</td>
<td>Msg Data</td>
<td>Msg Upper Addr</td>
<td>Msg Address</td>
<td>entry (N-1) Base +(N-1)*16</td>
</tr>
</tbody>
</table>

**MSI-X Table Structure**
### MSI-X Capability and Table Structures

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>63 62 61 .............</td>
<td>QWORD0 Base</td>
</tr>
<tr>
<td>Pending Bits 0 through 63</td>
<td></td>
</tr>
<tr>
<td>Pending Bits 64 through 127</td>
<td>QWORD1 Base + 1*8</td>
</tr>
<tr>
<td>....</td>
<td>QWORD0 Base</td>
</tr>
<tr>
<td>Pending Bits ((N-1) div 64)*64 through N-1</td>
<td>QWORD((N-1) div 64) Base+((N-1) div 64)*8</td>
</tr>
</tbody>
</table>

**MSI-X PBA Structure**
Enabling and Sending Message Interrupts

- Both MSI and MSI-X are disabled following reset.
- System configuration software sets either the MSI Enable bit or the MSI-X Enable bit to enable either MSI or MSI-X, but never both simultaneously.
- Once MSI or MSI-X is enabled, and one or more vectors is unmasked, the function is permitted to send messages.
- To send a message, a function does a DWORD memory write to the appropriate message address with the appropriate message data.
PCI-X Explained
What is PCI-X?

- "PCI-X is high-performance backward compatible PCI"
  - PCI-X uses the same PCI architecture
  - PCI-X leverages the same base protocols as PCI
  - PCI-X leverages the same BIOS as PCI
  - PCI-X uses the same connector as PCI.
  - PCI-X and PCI products are interoperable
  - PCI-X uses same software driver models as PCI

- PCI-X is faster PCI
  - PCI-X 533 is up to 32 times faster than the original version of PCI
  - PCI-X protocol is more efficient than conventional PCI
What is PCI-X 2.0?

- Revision 2.0 includes everything from Revision 1.0
  - PCI-X 66
    - Same clock speed as fastest conventional PCI
    - Easier timing and more efficient bus utilization
  - PCI-X 133
    - Twice as fast as conventional PCI
    - Easier timing and more efficient bus utilization
- Revision 2.0 introduces 2 new speed grades
  - PCI-X 266
    - “Double data rate” clocking for transfer rates up to 266MHz
    - Up to 2.13 Gigabytes per second of bandwidth
  - PCI-X 533
    - “Quad data rate” clock for transfer rates of up to 533MHz
    - Up to 4.26 Gigabytes per second of bandwidth
- All PCI-X devices are fully backward-compatible to:
  - 33 MHz conventional PCI (66 MHz support is optional)
  - All lower PCI-X speeds
### PCI-X Modes and Speeds

<table>
<thead>
<tr>
<th>Mode</th>
<th>V&lt;sub&gt;I/O&lt;/sub&gt;</th>
<th>64-Bit Slots*</th>
<th>MB/s</th>
<th>32-Bit Slots*</th>
<th>MB/s</th>
<th>16-Bit</th>
<th>Error Prot</th>
<th>Conf Bytes</th>
<th>DIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI 33</td>
<td>5V/3.3V</td>
<td>266</td>
<td>133</td>
<td>N/A</td>
<td>par</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI 66</td>
<td>3.3V</td>
<td>533</td>
<td>266</td>
<td>N/A</td>
<td>par</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI-X 66</td>
<td>3.3V</td>
<td>533</td>
<td>266</td>
<td>N/A</td>
<td>par or ECC</td>
<td>256</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI-X 133</td>
<td>3.3V</td>
<td>800</td>
<td>400</td>
<td>N/A</td>
<td>par or ECC</td>
<td>256</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI-X 133 (operating at 100 MHz)</td>
<td>3.3V</td>
<td>1066</td>
<td>533</td>
<td>N/A</td>
<td>par or ECC</td>
<td>256</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI-X 266</td>
<td>1.5V</td>
<td>2133</td>
<td>1066</td>
<td>533</td>
<td>ECC 4K</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI-X 533</td>
<td>1.5V</td>
<td>4266</td>
<td>2133</td>
<td>1066</td>
<td>ECC 4K</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* For lower bus speeds, # slots / bus is implementation choice to share bandwidth
### Registered Bus Protocol

- **PCI @ 33MHz**
  - 30 ns period
  - 7 ns setup time

- **PCI @ 66MHz**
  - 15 ns period
  - 3 ns setup time

- **PCI-X registered protocol allocates a full clock period for logic decision**
  - @ 66MHz - 15ns
  - @ 133MHz - 7.5ns
PCI-X protocol always takes 2 clocks to “turn around” a control event

- Register-to-register design allows maximum flight time
PCI 2.x/3.0 vs. PCI-X Mode 1

- Same bus and control signals
- Evolutionary protocol changes
- Clock frequency up to 133 MHz

New “Attribute” phase for enhanced features
PCI-X 66/133 (Mode 1) vs. PCI-X 266/533 (Mode 2)

- Same bus and control signals
- PCI-X 266 moves 2x the data
- PCI-X 533 moves 4x the data
- Clock frequency up to 133 MHz

PCI-X 66/133 (Mode 1)

PCI-X 533 (Mode 2)

4 transfers per clock cycle

source-synchronous data strobes share C/BE pins
### Requester Attributes for Burst Transactions

<table>
<thead>
<tr>
<th>R</th>
<th>NS</th>
<th>RO</th>
<th>Tag</th>
<th>Requester Bus Number</th>
<th>Requester Device Number</th>
<th>Requester Function Number</th>
<th>Lower Byte Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Upper Byte Count</th>
<th>C/BE[3-0]#</th>
<th>AD[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>35 32</td>
<td>31 30 29 28 24 23</td>
<td>16 15 11 10 08 07</td>
</tr>
</tbody>
</table>

### Requester Attributes for DWORD Transactions

<table>
<thead>
<tr>
<th>R</th>
<th>NS</th>
<th>RO</th>
<th>Tag</th>
<th>Requester Bus Number</th>
<th>Requester Device Number</th>
<th>Requester Function Number</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte Enables</th>
<th>C/BE[3-0]#</th>
<th>AD[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>35 32</td>
<td>31 30 29 28 24 23</td>
<td>16 15 11 10 08 07</td>
</tr>
</tbody>
</table>

**RO** -- Relax ordering

**NS** -- No Snoop

**R** -- Reserved
## Transaction Attributes

### Split Completion Address

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUS CMD</td>
<td>Requester command</td>
</tr>
<tr>
<td>C/BE[3-0]#</td>
<td>Completion and byte enable signal</td>
</tr>
<tr>
<td>AD[31:0]</td>
<td>Address [31:0]</td>
</tr>
<tr>
<td>Tag</td>
<td>Tag field</td>
</tr>
<tr>
<td>Upper Byte Count</td>
<td>Completer bus number</td>
</tr>
<tr>
<td>Lower Address [6:0]</td>
<td>Requester bus number</td>
</tr>
<tr>
<td>Requester Device Number</td>
<td>Requester device number</td>
</tr>
<tr>
<td>Requester Function Number</td>
<td>Requester function number</td>
</tr>
<tr>
<td>Lower Address [6:0]</td>
<td>Completer function number</td>
</tr>
</tbody>
</table>

RO -- Relaxed ordering

### Completer Attributes

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper Byte Count</td>
<td>Completer bus number</td>
</tr>
<tr>
<td>C/BE[3:0]#</td>
<td>Completer command</td>
</tr>
<tr>
<td>AD[31:0]</td>
<td>Address [31:0]</td>
</tr>
<tr>
<td>BCM SCM SCE SCM</td>
<td>Completer device number</td>
</tr>
<tr>
<td>R</td>
<td>Completer function number</td>
</tr>
<tr>
<td>Lower Byte Count</td>
<td>Completer function number</td>
</tr>
</tbody>
</table>

SCM -- Split Completion Message
SCE -- Split Completion Error
BCM -- Byte Count Modified
R -- Reserved
Split Transactions

- Bus efficiency of Read almost as good as Write
- Split Completion routed back to requester across bridges using initiator’s number and bus number
- Split Transaction components
  - Step 1. Requester requests bus and arbiter grants bus
  - Step 2. Requester initiates transaction
  - Step 3. Target (completer) communicates intent with new target termination, Split Response
  - Step 4. Completer executes transaction internally
  - Step 5. Completer requests bus and arbiter grants bus
  - Step 6. Completer initiates Split Completion
Split Transactions

PCI Bus

Requester A

Completer B

PCI Bus

Requester Attributes

Address, Memory Read

Split Response

(Requester's Attribute)

Split Completion

Immediate Response

Completer Attributes

DATA

Initiator

Target

Completer Attributes

Initiator

Target

Split Transaction Requester

Split Transaction Completer
The PCI-X protocol is more efficient than traditional PCI.

Bandwidth Usage with Conventional PCI Protocols

Bandwidth Usage with PCI-X Protocols, included in PCI-X 2.0

Idle Time -- Unused BW
System Overhead -- Scheduling
Transaction Overhead -- Addressing and Routing
Transaction Data Payload -- Actual user data

System Overhead -- Scheduling
Transaction Overhead -- Addressing and Routing
Transaction Data Payload -- Actual user data
Idle Time -- Unused BW
PCI-X I/O Signaling Voltages

- PCI-X 66 and 133
  - 3.3V signaling
  - Card-edge connector keyed for “3.3V” or “Universal” signaling
- PCI-X 266 and 533 use combination of 3.3V I/O and new 1.5V I/O
  - Control signals use 3.3V I/O
  - Data and strobe signals use 1.5V I/O
    - Faster signaling rates
    - Point-to-point and electrically terminated for improved noise immunity
    - New interface low-power state to manage interface power
    - I/O buffer change only
      - Same system supply voltages
      - Automatic selection by devices at power-up
  - Card-edge connector keyed for “3.3V signaling”
PCI-X 2.0 Offers Improved RAS Features

- Parity protection
  - Provides full compatibility with conventional PCI and PCI-X 1.0

- ECC protection new in PCI-X 2.0
  - Covers both header and payload
  - Provides automatic single bit error recovery
  - Detects all double bit errors
  - Detects all errors in single nibble
  - Detects phase errors (e.g. missed strobe or extra strobe)
  - Adds no additional latency over parity
  - Required for Mode 2; optional for Mode 1

- Enhanced data-error recovery options
  - Available both for Mode 1 and Mode 2
Compatible with Conventional PCI

- No OS or driver change required
  - New configuration registers default to functional values
  - Optional performance tuning registers
  - Other configuration registers unchanged
  - No device programming model changes required

- Optional improved error handling
  - Enables smart device and new driver to recover from PERR# event
PCI Express Overview

**Scalable/Extensible I/O**
- Scalable in performance and feature set
- Suitable for over 10-year horizon
- High-end and mainstream applications

**Cost Effective**
- PCI cost structure at system level
- Low power, no sidebands
- Commodity ingredients: FR-4 PCBs, simple connectors, low manufacturing costs

**Multiple Market Segments/Applications**
- Mobile, desktop, server and communication devices
- Chip-to-chip, board-to-board, modules, docking, cables

**Compatibility & Smooth Migration**
- Preserves investments in PCI ecosystem
- Path to future enhancements and proliferations

Serial, point-to-point interconnect of choice for all platform applications
PCIe Architecture Features

- **PCI Compatibility**
  - Configuration and PCI software driver model
  - PCI power management software compatible

- **Performance**
  - Scalable frequency (2.5-5GT/s)
  - Scalable width (x1, x4, x8, x16)
  - Low latency and highest utilization (BW/pin)

- **Physical Interface**
  - Point-to-point, dual-simplex
  - Differential low voltage signaling
  - Embedded clocking
  - Supports connectors, modules, cables

- **Protocol**
  - Fully packetized split-transaction
  - Credit-based flow control
  - Hierarchical topology support
  - Virtual channel mechanism

- **Advanced Capabilities**
  - CRC-based data integrity, hot plug, error logging

- **Enhanced Configuration Space**
  - Extensions and bridges into other architectures
PCIe Protocol Overview

- **PCI-X Address/Attribute phases:**
  - Evolved into the PCIe Packet Header:

```
<table>
<thead>
<tr>
<th>Byte 0</th>
<th>7 6 5 4 3 2 1 0</th>
<th>7 6 5 4 3 2 1 0</th>
<th>7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>Fmt</td>
<td>Type</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 4</td>
<td>Requester ID</td>
<td>Tag</td>
<td>Last DW</td>
</tr>
<tr>
<td>Byte 8</td>
<td></td>
<td></td>
<td>BE</td>
</tr>
<tr>
<td>Byte 12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
The packet bytes get converted to 8b/10b and serialized
PCIe Protocol Overview

- Framing varies depending on link width
  - x1

![Diagram showing PCIe protocol layers and framing symbols.]

- Reserved bits and Sequence Number added by Data Link Layer
- STP Framing Symbol added by Physical Layer
- TLP generated by Transaction Layer
- LCRC added by Data Link Layer
- END Framing Symbol added by Physical Layer
PCIe Protocol Overview

- Framing varies depending on link width
  - x4

![Diagram showing PCIe lane layout and framing symbols]

- STP/END Framing Symbols
- Physical Layer
- Sequence Number/LCRC
- Data Link Layer
- TLP - Transaction Layer
PCIe Architecture Specifications

Form Factors:
- Card
- Mini Card
- Express Module
- Wireless Card (WIP)
- Cable (WIP)

- Base
  - Electrical
  - Protocol
  - Configuration
- Bridge
- I/O Virtualization (WIP)

- Layered, scalable architecture
- Performance matched to applications
- Innovative form factors