Acknowledgements

Thanks are due to Ravi Budruk, Mindshare, Inc. for much of the material on PCI Express Basics
Agenda

- PCI Express Background
- PCI Express Basics
- PCI Express Recent Developments
PCI Express Background
Revolutionary AND Evolutionary

- **PCI™** (1992/1993)
  - 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 2.1 (1995) doubled bandwidth with 66MHz mode
Revolutionary AND Evolutionary

- **PCI-X™ (1999)**
  - 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 2.0 (2003) doubled bandwidth
      - 2133MB/sec at PCI-X 266 and 4266MB/sec at PCI-X 533
Revolutionary AND Evolutionary

- PCI Express – aka PCIe® (2002)
  - Revolutionary
    - Unprecedented bandwidth
      - x1: up to 1GB/sec in *EACH* direction
      - x16: up to 16GB/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, etc.
      - Of course, PCIe-aware O/S can get more functionality
    - Transaction layer familiar to PCI/PCI-X designers
    - System topology matches PCI/PCI-X
    - PCIe 2.0 (2006) doubled per-lane bandwidth: 250MB/s to 500MB/s
    - PCIe 3.0 (2010) doubled again to 1GB/s/lane… PCIe 4.0 will double again to 2GB/s/lane!
PCI Concepts
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
    - 4 DWORD 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 in PCI-X and PCIe
  - Bus / Device / Function (aka BDF) form hierarchy-based address (PCIe 3.0 calls this “Routing ID”)
    - “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)
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**
Configuration Space – Capabilities List

- 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.

Feature-specific Configuration Registers

<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>Pointer to Next Capability</td>
<td>Capability ID</td>
<td></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>
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>Pointer to Next Capability</th>
<th>Version</th>
<th>Capability ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature-specific Configuration Registers</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Dword 0 | Dword 1 | Dword n |
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
What are MSI and MSI-X?

- Memory Write replaces previous interrupt semantics
  
  ✓ PCI and PCI-X devices stop asserting INTA/B/C/D and PCI Express devices stop sending Assert_INTx messages once MSI or MSI-X mode is enabled
  
  ✓ MSI uses one address with a variable data value indicating which “vector” is asserting
  
  ✓ MSI-X uses a table of independent address and data pairs for each “vector”

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

- PCI commands contained no length
  - Bus allowed disconnects and retries
  - Difficult data management for target device
    - Writes overflow buffers
    - Reads require pre-fetch
      - How much to pre-fetch? When to discard? Prevent stale data?

- PCI commands contained no initiator information
  - No way for target device to begin communication with the initiator
  - Peer-to-peer requires knowledge of system-assigned addresses
Split Transactions

- PCI-X commands added length and Routing ID of initiator
  - Writes: allow target device to allocate buffers
  - Reads: Pre-fetch now deterministic
- PCI-X retains “retry” & “disconnect”, adds “split”
- Telephone analogy
  - Retry: “I’m busy go away”
    - Delayed transactions are complicated
  - Split: “I’ll call you back”
    - Simple
    - More efficient
Benefits of Split Transactions

Bandwidth Usage with Conventional PCI Protocols

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

Bandwidth Usage with PCI-X Enhancements

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

Number of Load Exerciser Cards

Bandwidth MegaBytes/sec

Percent of Total Bandwidth

System Overhead

Transaction Overhead

Bandwidth

MegaBytes/sec

Percent of Total Bandwidth

100%

90%

80%

70%

60%

50%

40%

30%

20%

10%

0%

25 50 75 100 125 150 175 200 225 250 275

25 50 75 100 125 150 175 200 225 250 275

1 2 3 4 5

1 2
PCI Express Basics
PCI Express Features

- Dual Simplex point-to-point serial connection
  - Independent transmit and receive sides
- Scalable Link Widths
  - x1, x2, x4, x8, x12, x16, x32
- Scalable Link Speeds
  - 2.5, 5.0 and 8.0GT/s (16GT/s coming in 4.0)
- Packet based transaction protocol
PCI Express Terminology

PCI Express Device A

PCI Express Device B

Signal

Wire

Link

Lane

PCIe Technology Seminar
Copyright © 2014, PCI-SIG, All Rights Reserved
## PCI Express Throughput

<table>
<thead>
<tr>
<th>Bandwidth (GB/s)</th>
<th>Link Width</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x1</td>
</tr>
<tr>
<td>PCIe 1.x</td>
<td>0.25</td>
</tr>
<tr>
<td>&quot;2.5 GT/s&quot;</td>
<td></td>
</tr>
<tr>
<td>PCIe 2.x</td>
<td>0.5</td>
</tr>
<tr>
<td>&quot;5 GT/s&quot;</td>
<td></td>
</tr>
<tr>
<td>PCIe 3.0</td>
<td>1</td>
</tr>
<tr>
<td>&quot;8 GT/s&quot;</td>
<td></td>
</tr>
<tr>
<td>PCIe 4.0</td>
<td>2</td>
</tr>
<tr>
<td>&quot;16GT/s&quot;</td>
<td></td>
</tr>
</tbody>
</table>

### Derivation of these numbers:

- 20% overhead due to 8b/10b encoding in 1.x and 2.x
- Note: ~1.5% overhead due to 128/130 encoding not reflected above in 3.x and 4.0
Additional Features

- Data Integrity and Error Handling
  - Link-level “LCRC”
  - Link-level “ACK/NAK”
  - End-to-end “ECRC”

- Credit-based Flow Control
  - No retry as in PCI

- MSI/MSI-X style interrupt handling
  - Also supports legacy PCI interrupt handling in-band

- Advanced power management
  - Active State PM
  - PCI compatible PM
Additional Features

- Evolutionary PCI-compatible software model
  - PCI configuration and enumeration software can be used to enumerate PCI Express hardware
  - PCI Express system will boot “PCI” OS
  - PCI Express supports “PCI” device drivers
  - New additional configuration address space requires OS and driver update
    - Advanced Error Reporting (AER)
    - PCI Express Link Controls
PCI Express Topology

Legend
- □: PCI Express Device Downstream Port
- □: PCI Express Device Upstream Port
Request are translated to one of four transaction types by the Transaction Layer:

1. **Memory Read or Memory Write.** Used to transfer data from or to a memory mapped location.
   - The protocol also supports a *locked memory read* transaction variant

2. **I/O Read or I/O Write.** Used to transfer data from or to an I/O location.
   - These transactions are restricted to supporting legacy endpoint devices

3. **Configuration Read or Configuration Write.** Used to discover device capabilities, program features, and check status in the 4KB PCI Express configuration space.

4. **Messages.** Handled like posted writes. Used for event signaling and general purpose messaging.
Three Methods For Packet Routing

- Each request or completion header is tagged as to its type, and each of the packet types is routed based on one of three schemes:
  - ✓ Address Routing
  - ✓ ID Routing
  - ✓ Implicit Routing
- Memory and IO requests use address routing
- Completions and Configuration cycles use ID routing
- Message requests have selectable routing based on a 3-bit code in the message routing sub-field of the header type field
**Programmed I/O Transaction**

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

**Completer:**
- Step 2: Endpoint (completer) receives MRd
- Step 3: Endpoint returns Completion with data (CplD)
DMA Transaction

**Completer:**
- Step 2: Root Complex (completer) receives MRd
- Step 3: Root Complex returns Completion with data (CplD)

**Requester:**
- Step 1: Endpoint (requester) initiates Memory Read Request (MRd)
- Step 4: Endpoint receives CplD
Peer-to-Peer Transaction

Requester:
- Step 1: Endpoint (requester) initiates Memory Read Request (MRd)
- Step 4: Endpoint receives CplD

Completer:
- Step 2: Endpoint (completer) receives MRd
- Step 3: Endpoint returns Completion with data (CplD)
TLP Origin and Destination

PCI Express Device A

Device Core

PCI Express Core

Logic Interface

Transaction Layer

Data Link Layer

Physical Layer

PCI Express Device B

Device Core

PCI Express Core

Logic Interface

Transaction Layer

Data Link Layer

Physical Layer

Link

TLP Transmitted

TLP Received
TLP Structure

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

Bit transmit direction

Created by Transaction Layer

Appended by Data Link Layer

Appended by Physical Layer
DLLP Origin and Destination

PCI Express Device A

- Device Core
  - PCI Express Core
  - Logic Interface
    - TX
    - RX
    - Transaction Layer
    - Data Link Layer
      - Physical Layer

PCI Express Device B

- Device Core
  - PCI Express Core
  - Logic Interface
    - TX
    - RX
    - Transaction Layer
    - Data Link Layer
      - Physical Layer

DLLP Transmitted

DLLP Received

Link
### DLLP Structure

- **Bit transmit direction**

```
Start  DLLP  CRC  End
1B     4B    2B    1B
```

- **Data Link Layer** appended by Physical Layer

- **ACK / NAK Packets**
- **Flow Control Packets**
- **Power Management Packets**
- **Vendor Defined Packets**
Ordered-Set Origin and Destination

PCI Express Device A

PCI Express Core

Logic Interface

Transaction Layer

Data Link Layer

Physical Layer

TX

RX

PCI Express Device B

PCI Express Core

Logic Interface

Transaction Layer

Data Link Layer

Physical Layer

TX

RX

Ordered-Set Transmitted

Ordered-Set Received

Link
Ordered-Set Structure

- **Training Sequence One (TS1)**
  - ✓ 16 character set: 1 COM, 15 TS1 data characters

- **Training Sequence Two (TS2)**
  - ✓ 16 character set: 1 COM, 15 TS2 data characters

- **SKIP**
  - ✓ 4 character set: 1 COM followed by 3 SKP identifiers

- **Fast Training Sequence (FTS)**
  - ✓ 4 characters: 1 COM followed by 3 FTS identifiers

- **Electrical Idle (IDLE)**
  - ✓ 4 characters: 1 COM followed by 3 IDL identifiers

- **Electrical Idle Exit (EIEOS) (new to 2.0 spec)**
  - ✓ 16 characters
Credit-based *flow control* is point-to-point based, not end-to-end

Receiver sends Flow Control Packets (FCP) which are a type of DLLP (Data Link Layer Packet) to provide the transmitter with credits so that it can transmit packets to the receiver.
ACK/NAK Protocol Overview

From Transaction Layer

Transmit Device A

Data Link Layer

Tx

TLP
Sequence | TLP | LCRC

Replay Buffer

De-mux

Mux

Rx

DLLP
ACK/NAK

To Transaction Layer

Receiver Device B

Data Link Layer

Tx

TLP
Sequence | TLP | LCRC

DLLP
ACK/NAK

De-mux

Mux

Rx

Tx

DLLP
ACK/NAK

Error Check

Rx

TLP
Sequence | TLP | LCRC
PCI Express
Recent Developments
New Specifications

- M.2 Specification, Revision 1.0

- PCI Express Base 4.0, Draft 0.3
  https://www.pcisig.com/members/downloads/PCI_Express_Base_4.0_Rev0.3_February19-2014.pdf

- OCuLink, Draft 0.7

- SFF-8639, Draft 0.7
Newer ECNs
(See session 4 after lunch for details)

- NOP DLLP

- Readiness Notifications (RN)

- M-PCIe

- L1 PM Substates
Upcoming Events

- Compliance Workshop #91 (Taipei)
  - Tomorrow 😊
- Compliance Workshop #92 (California)
  - December 2-5, 2014
- PCI-SIG Developers Conference Israel
  - March 2-3, 2015
- PCI-SIG Developers Conference
  - June 23-24, 2015
- PCI-SIG Developers Conference APAC
  - [TBD] October 2015?
Present a DevCon Member Implementation Session

- Watch for e-mailed Call For Papers
- Send in an abstract!
  - 160 word summary
  - Ok to attach more detail (even a presentation)
  - No confidential material!
  - Not a datasheet or catalog or other marketing!
- Get selected
- Meet milestones and deadlines
- Practice, practice, practice the presentation
- Present at DevCon
Thank you for attending the PCIe Technology Seminar.

For more information please go to www.pcisig.com