ACPI-Based Support For Intel Thunderbolt Hot-Plug
In the Linux* Kernel

Rafael J. Wysocki

Intel Open Source Technology Center

March 25, 2014
Outline

1. Technology Overview
   - Intel Thunderbolt
   - ACPI Events Handling

2. Support In The PCI Subsystem
   - ACPI-Based PCI Hot-Plug (ACPIPHP)
   - PCI Device Addition/Removal And Concurrency

3. Resources
Device interconnect technology

1. DisplayPort forwarding
   - Transparent with respect to software.

2. PCI Express with hot-plug
   - PCIe hierarchy branch (multiple devices) on a single link.
   - Daisy chains of devices.
   - Out-of-band event (hot-add, hot-remove) signaling.
Thunderbolt Signaling Arrangements

Apple systems
- Events signaled through hardware interrupts.
- Handled by the OS (interrupt handler).
- OS responsible for PCIe bus re-configuration.

Systems from other vendors
- Events signaled through SMI (System Management Interrupts).
- Handled by the BIOS in SMM (System Management Mode).
- BIOS responsible for PCIe bus re-configuration.
- OS notified via ACPI bus check notifications.
BIOS-Based Thunderbolt Event Handling Control Flow

- Link Event (plug, unplug)
- SMI
- SMI handler (SMM)
- ACPI GPE
- Re-configure PCIe Bus
ACPI GPE (Simplest Case)

General Purpose Event (GPE)

- Pair of bits: enable and status (same position, different registers).
- One enable and one status register per block.
- Two “fixed” blocks, GPE block 0 and GPE block 1.
- Status bit set triggers an ACPI SCI (System Control Interrupt).
- Status bit may be set if the corresponding enable bit is set.

GPE number: 0...0xFF

Bit position within GPE blocks 0 and 1.
GPE Handling Control Flow

1. ACPI Interrupt
2. Interrupt Handler
3. GPE Status Set
4. Determine GPE Number
5. GPE Handler Present?
   - YES: Execute GPE Handler
   - NO: Queue up GPE Work
   - _Lxx / _Exx Present?
     - YES: Execute _Lxx / _Exx
     - NO: Notify (Object, Device Wake)
   - Wakeup GPE?
     - YES: Notify (Object, Device Wake)
     - NO: GPE Work Function
ACPI Namespace

Kernel data structure
- Hierarchy (tree) of objects representing various system parts.
- Supposed to reflect system topology.
- Contains data and code (methods).
- Covers things known to the BIOS only.

ACPI namespace management
- Created during system startup.
- May be dynamically extended and (theoretically) shrunk later.
- Encoded in data structures called *definition blocks* in ACPI tables.
ACPI Namespace Continued

ACPI Machine Language (AML)

Byte code compiled from sources in ACPI Source Language (ASL) and used for encoding definition blocks.
ACPI GPE Handling

ACPI interrupt (e.g. SCI) handler determines the GPE number

1. Execute _Lxx/_Exx (AML methods associated with GPEs) if present.
2. Execute Notify (Object, Device Wake) if possible.

In a typical system
_Lxx/_Exx determine target objects and execute Notify () for them.

Notify (Object, Notification Value)
AML operator causing callbacks supplied by the OS (kernel) to be run.
ACPI Device Representation

Devices are represented by ACPI namespace nodes

Objects below those nodes may represent

- Data (names, addresses etc.).
- Control methods (pieces of AML code).
- Dependent (child) devices.

Each namespace object (except for the root) has exactly one parent.

Each namespace object can be unambiguously identified by the path to it from the namespace root.
Device Objects In ACPI Namespace (Example)
ACPI Notification Handlers

**Notify()** executes callbacks supplied by the OS

```c
notify_handler(ACPI object handle, event code, data pointer)
```

```c
acpi_install_notify_handler(ACPI object handle, handler type, notify_handler, data pointer)
```

Problem: Ensure that relevant notify handlers are present

- Before 3.12: Objects corresponding to “removable” PCI devices.
- 3.12 – 3.14: All objects corresponding to PCI devices (almost).
- 3.15-rc: Use the global notify handler.
Identification Of Notification Targets

Target device identification problem
Find the PCI device (struct pci_dev), if present, represented by the given object in the ACPI namespace.

Before 3.12: Use the data pointer argument of notify handlers
Racy: That pointer may become invalid while the handler is running.

3.12 – 3.14: Attach context data to namespace objects
Use acpi_attach_data()/acpi_get_data() and proper locking.

In 3.15-rc context data are attached to struct ACPI_device objects.
ACPI Bus Check Notifications Handling

Bus Check Notification (code 0)

Bus re-enumeration is required (starting at the target object).

- Walk the bus from the target device down.
- Remove devices that are not present any more.
- Add devices that has just appeared.

Before 3.12 that only really worked for PCI devices with corresponding ACPI namespace objects (insufficient for Thunderbolt hot-plug).

3.12 and later check the entire PCI tree branch below the notify target.
Preservation Of BIOS-Provided Device Configuration

- Hot-added devices may be already configured by the BIOS.
- Changing their configuration may lead to problems.

BIOS-Provided configuration settings should be used as long as they work:
- 3.10 and earlier kernels always tried to configure hot-added devices.
- 3.11 and later do their best to use the BIOS-provided settings.

The change in 3.11 was made as a fix related to docking stations holding PCIe devices, but it also is necessary for supporting Thunderbolt hot-plug.
Before 3.12 PCI hot-plug notifications (bus check, device check, eject requests) might be handled during system suspend or resume.

That lead to problems when device drivers’ `.remove()` callbacks were executed before their system resume callbacks (or between them).

3.12 and later kernels use locking to prevent hot-plug notifications from being handled during system suspend/resume.
PCI Device Remove And Bus Rescan sysfs Interface

```
/sys/devices/pci0000:00/.../remove
```
Remove struct `pci_dev` representing the device (as though the device was removed).

```
/sys/devices/pci0000:00/.../rescan
```
Re-scan the PCI bus starting from this device, possibly adding `struct pci_dev` objects that are missing.

This has to be synchronized with hot-plug notification handling, but that synchronization was not there in 3.13 and earlier kernels.
PCI Device Remove And Bus Rescan Synchronization

In 3.14 the locking used internally by the remove/rescan sysfs interface was extended to all PCI device addition and removal code paths.

In 3.14 and later kernels all code paths involving PCI device addition or removal are (supposed to be) mutually exclusive.

Among other things this allows different PCI device hot-plug frameworks to co-exist in the same system.
Surprise Device Removal And PCI Drivers

With Thunderbolt PCIe devices may physically go away before their drivers’ .remove() callbacks are executed.

That may happen at any time.

PCI device drivers should be prepared for that

- Lack of device response need not be an error.
- Handle the lack of device response gracefully.
- Do not wait for device response forever (or poll it forever).
- Clean up everything even if the device doesn’t respond.
- During the clean-up do not touch hardware that is gone.
Summary

- Intel Thunderbolt is supported on systems where the BIOS participates in the handling of link events.
- ACPI bus check notifications are then used to notify the kernel about PCIe bus configuration changes.
- The kernel code responsible for handling those notifications had to be modified substantially to cover Thunderbolt hot-plug properly.
- The PCI subsystem’s locking was hardened in the process.
- Thunderbolt hot-plug support implies that device drivers have to be prepared to handle surprise device removal gracefully.

Questions?


*Advanced Configuration and Power Interface Specification, Revision 5.0a* (http://www.acpi.info/DOWNLOADS/ACPI_5_Errata%20A.pdf).

Documentation And Source Code

- https://acpica.org
- Documentation/acpi/
- include/acpi/
- include/linux/acpi.h
- include/linux/device.h
- include/linux/pci*.h
- drivers/acpi/
- drivers/pci/
- drivers/pci/hotplug/acpiphp*
- drivers/pci/pci-acpi.c
Thanks!

Thank you for attention!