Notes

Introduction

PCIe® Hot-swap is not to be confused with PCIe Hot-plug. PCIe Hot-plug is derived from revision 1.0 of the Standard Hot Plug Controller specification for PCI[1]. This specification describes the methodology by which PCIe endpoint devices may be added to or removed from an operational system without compromising the operational state of the system. The specification defines a number of standard hardware registers and signals, such as Attention Button and Attention Indicator, that allow the development of a common Hot-plug device driver. PCIe Hot-plug is designed as a "no-unexpected" or "graceful" methodology, i.e., the user is not permitted to install or remove a PCIe endpoint device without first notifying the system software.

There is no standard for PCIe Hot-swap; it is system-dependant and usually implemented by the system vendor. PCIe Hot-swap allows an endpoint or one or more PCIe switches with one or more endpoints to be inserted or removed from a PCIe system gracefully or unexpectedly.

This application note discusses some of the issues and firmware considerations as they relate to implementing PCIe Hot-swap on standard PC based systems.

Hot-Plug

The Hot-Plug Specification describes the methodology by which an endpoint may be replaced in a running system without having to turn the system off.

In order to facilitate such a procedure, additional hardware is required to control power to the slot (refer to Figure 1), to relay user requests to the host operating system, and for the operating system to relay status information back to the user. The basic procedure for Hot-plug is as follows:

- User presses attention button to inform operating system that card is to be removed.
- Operating system places card driver in quiescent state, powers down slot, and informs user that card is safe to remove.
- User removes and optionally replaces card.

The procedure for installing a new endpoint is essentially the reverse of the removal procedure. Not only is this a lengthy procedure from the user point of view but the additional hardware required to support Hot-plug also makes it impractical in price sensitive PC markets.

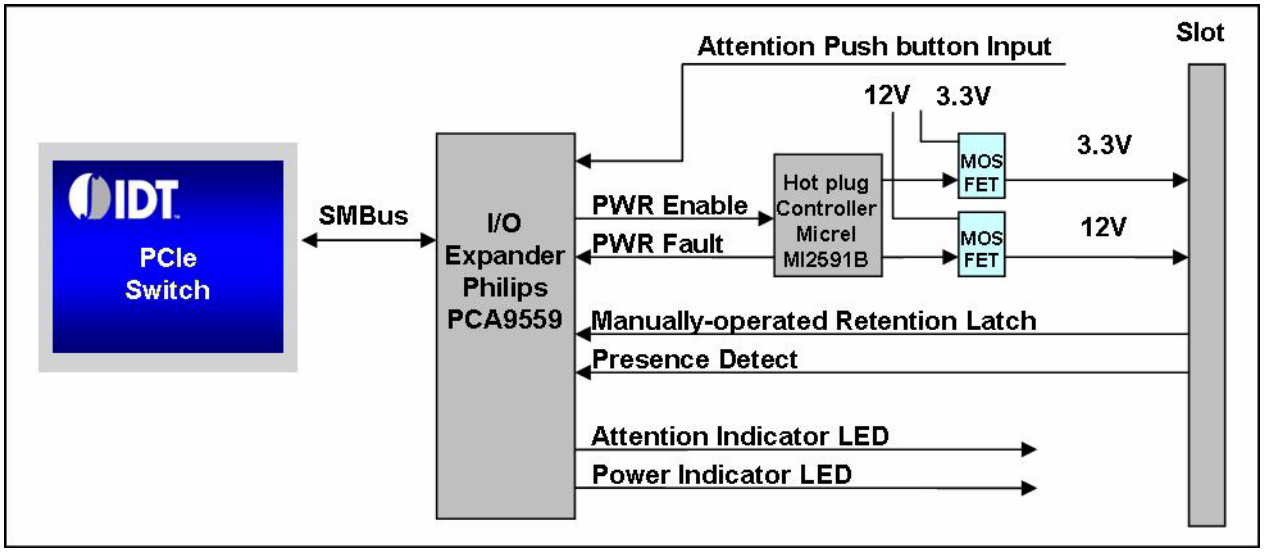

Figure 1 PCIe Hot-Plug Block Diagram

Hot-Swap

PCIe Hot-swap allows an endpoint or one or more PCIe switches with one or more endpoints to be inserted or removed from a PCIe system gracefully or unexpectedly without special consideration. In a Hot-swap environment, PCIe endpoint devices are connected to PCIe switches or to the root complex utilizing PCIe cables. The fact that these cables only carry data and that the PCIe transceivers are Hot-plug safe is what makes it possible for Hot-swap to be realized on standard "off the shelf" PC based architectures without the need for additional hardware.

Firmware Considerations

Although the principles of Hot-swap are fairly straight forward, there are a number of factors that need to be taken into consideration when implementing such a system on PC based architectures, the most important of which is the BIOS.

PC BIOS

It's important to note that it is the PC BIOS and not the operating system that performs the initial PCIe bus enumeration and device resource allocation. The operating system merely scans the PCIe devices when it boots and reserves the memory and I/O resources specified in the endpoint Base Address Registers (BARs).

In the past, it was only necessary to access 256 bytes of PCI configuration space. PC architectures utilized registers located at I/O ports 0xCF8/0xCFC to first select the device and configuration space address to be accessed and then to read or write data to the specified device. With the introduction of PCIe, the configuration space was extended to 4KB in size. Because only 8 bits were assigned to the PCI configuration space address register and to maintain backward compatibility, a new method of accessing the PCIe extended configuration space was required. The introduction of "Memory Mapped Configuration" (MMCFG) access satisfied this requirement. Instead of using I/O mapped address and data registers, MMCFG maps the actual configuration space for each endpoint into the systems address map.

The BIOS sets up the MMCFG space used to access the PCIe configuration space. On a standard PC, the MMCFG is typically a 256MB window that starts at physical address 0xE0000000. This allows access to 4KB of configuration space for each endpoint device up to the maximum number of devices supported by PCIe (8 functions x 32 devices x 256 buses x 4KB of configuration space = 256MB).

The exact location and the bus number range that may be accessed via the MMCFG window is communicated from the BIOS to the OS by means of the ACPI[2] MCFG[3] structure.

Notes

BIOS Limitations

In general, the PC BIOS was not written with Hot-swap or even Hot-plug in mind. It is assumed that, once booted, the hardware configuration will remain static. Because of this assumption, a number of complexities are introduced by the BIOS when implementing Hot-swap.

First, as can be seen in Figure 2, only one bus number is assigned to each root port in the system when it boots.

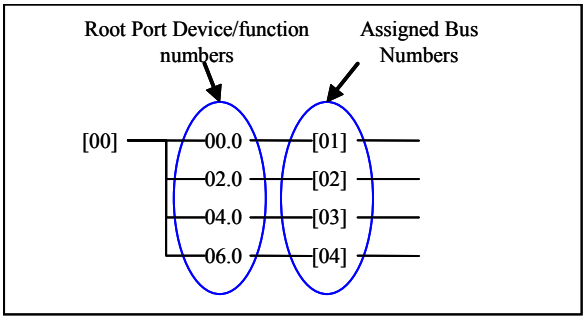

Figure 2 Bus Number Assignments After Boot

If a PCIe switch were to be connected to root port 02.0 after the system boots, there will be an insufficient number of bus numbers to assign to the newly connected switch. See Figure 3.

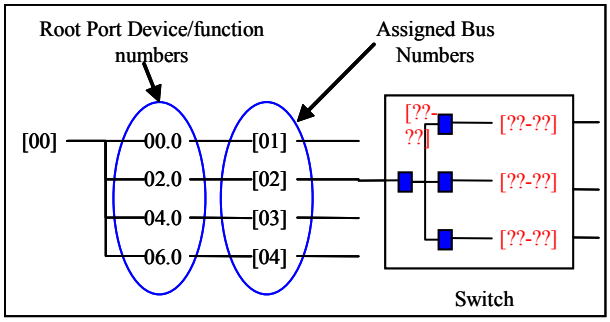

Figure 3 Insufficient Bus Numbers

Also, no memory or I/O resources are allocated to empty slots and no additional resources are allocated to populated slots.

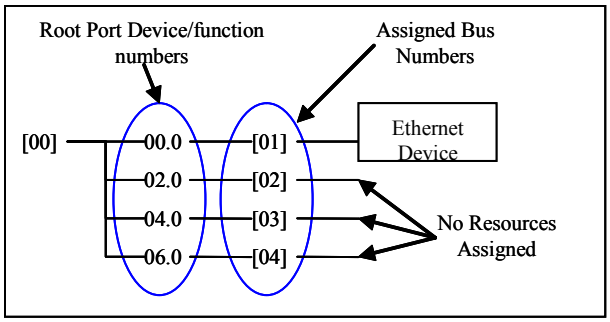

Figure 4 Insufficient Resources

Notes

This causes problems when a device is connected to a root port that was previously unpopulated or when a device is replaced with a device that requires more resources. Finally, the BIOS on some PCs limits the size of the MMCFG window based on the number of buses detected when the system booted[6]. This restricts access to devices on bus numbers higher than the maximum bus number discovered.

All of these limitations must be overcome in order for Hot-swap or even Hot-plug to work on PC based architectures. The ideal situation would be to prevent the BIOS from enumerating the PCIe buses and assigning memory and I/O resources to devices on specified root ports, and instead to allow the operating system to perform these tasks. In practice, this is not possible without modification to the BIOS, so alternate methods must be developed.

Bus Enumeration and Resource Allocation

When a new device is added to a running system, there are two methods by which the lack of bus numbers, memory, and I/O resources assigned by the BIOS can be addressed.

The first approach is for the operating system to stop all devices and place the device drivers for all devices sharing the same root port as the device being added into a quiescent state. Once in this state, the entire branch of the PCIe bus, starting at the root port, may be re-enumerated and have new memory and I/O resources allocated before restarting the device drivers. This approach has an advantage in the desktop PC environment where the user may want to connect/disconnect any number of devices at any time. The down side is that it may be undesirable to stop all devices on a branch of the PCIe bus just to add one new device. Furthermore, if for any reason a device cannot be stopped and its driver placed into a quiescent state, it is impossible to install the new device since the bus enumeration and resource allocation procedure is only possible when all devices on the bus have been stopped.

The second approach, depicted in Figure 5, is for the operating system to assign additional bus numbers, memory, and I/O resources to each port on the PCIe bus. Effectively, the operating system must re-enumerate the PCIe buses.

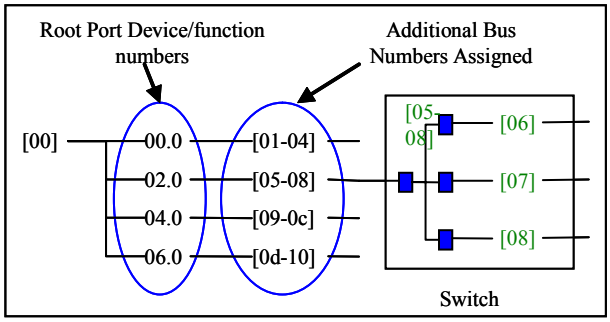

Figure 5 Additional Bus Number Assignment

When a new device, such as the switch shown in Figure 5, is connected, the operating system will be able to assign the pre-allocated resources to the new device. This approach has the obvious advantage of not having to stop all devices on the branch of the bus to which the new device is being connected. The downside is that the system configuration must be known ahead of time so the operating system can be configured to reserve the required resources to support the maximum system configuration. For this reason, this approach is better suited for systems where the maximum system configuration is known ahead of time.

Notes

MMCFG Access

Being able to re-enumerate PCIe buses when new devices are connected to the system is one thing. Being able to access these devices after the buses have been re-enumerated is something else altogether.

Some operating system APIs will restrict access to the MMCFG window based on the bus number being accessed. For example, assume that when the BIOS enumerated the PCIe buses it detected 5 buses, bus #0-4. The operating system then re-enumerated the PCIe buses adding an additional 4 bus numbers, bus #5-8. When an attempt is made to access devices on bus #5-8, because these bus numbers are outside the bus number range specified initially by the BIOS, the operating system may revert to the legacy mode of PCI configuration space access (I/O ports 0xCF8/0xCFC). Using this method to access the PCIe configuration space will limit configuration space access to the first 256 bytes of configuration address space. One operating system known to behave in this manner is Linux (2.6.18 Kernel). In this case, it is important that the operating system knows the highest PCIe bus number in the system in order for the configuration space access API functions to access the extended configuration space.

It is also possible for the BIOS to limit the size of the MMCFG window based on the number of PCIe buses discovered during the initial enumeration process. It is the responsibility of the operating system to resize the MMCFG window to match the actual system configuration as devices are added. Such behavior has been observed on AMD K8 based PC's loaded with the Phoenix AwardBIOS.

Device Removal and Insertion

Since there is no attention button for the user to press to inform the host operating system that a device is to be removed or inserted, automatic device removal/insertion detection must be performed. Detecting the insertion or removal of a PCIe device is accomplished in one of three ways.

1. By checking the Device Link Layer Link Active (DLLLA) bit in the PCIe Link Status register of the down-stream ports [4].
2. By reading a vendor-specific link status register of the down-stream ports[5].
3. By attempting to read the device/vendor IDs of the device connected to the down-stream ports[4].

Since not all devices support DLLLA status reporting, if after checking the Link Capabilities it is determined that the down-stream port does not support DLLLA reporting and if no vendor specific link status register is provided, method #3 should be used to detect the presence/absence of an endpoint device.

Interrupts

Interrupts are another area where care should be taken when re-enumerating PCIe buses. In some cases, operating systems associate specific CPU interrupt sources with PCIe root port bus numbers. If the PCIe root port bus numbers change, as they would in the case of bus re-enumeration, the operating system will no longer be able to associate the correct interrupt with the re-enumerated bus number. Again, with Linux, this is indeed the case. Therefore, after re-enumerating a PCIe root port it is necessary to update the operating system's PCI interrupt routing table in order to keep it in sync with the updated bus numbers.

Conclusion

Even with the limitations of the PC BIOS, it is possible to implement a system that utilizes PCIe Hot-swap on standard PC architectures. Implementing this same methodology on an embedded system, such as the MPC8548e, is also possible. In fact, it should be simpler due to the lack of a BIOS.

IDT has developed and can make available the source code for a Linux device driver that demonstrates the principles outlined in this document for use on platforms utilizing IDT PCIe switches.

References

[1] PCI Hot-Plug Specification http://www.pcisig.com

[2] Advanced Configuration and Power Interface Specification http://www.acpi.info

[3] PCI Firmware Specification, Revision 3.0 http://www.pcisig.com

Notes

[4] PCI Express® Base Specification Revision 2.0 http://www.pcisig.com

[5] Intel® E7520 Memory Controller Hub (MCH) http://www.intel.com

[6] BIOS and Kernel Developer's Guide for AMD Athlon™ 64 and AMD Opteron™ Processors http://www.amd.com

Revision History

September 16, 2008: Initial publication.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Rev.1.0 Mar 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.