PCle driver development for Exynos SoC

Korea Linux Forum 2013

Jingoo Han
Samsung Electronics
S/W engineer at Samsung Electronics since 2005

- Linux kernel development for Samsung Exynos ARM SoC
- Develop Linux drivers
  - PCIe driver, USB driver
  - Framebuffer driver, Display Port driver
- Developed Samsung smart phones and tablet PC.
Introduction

- Linux kernel Maintainer since 2011
  - maintain 3 drivers and 1 subsystem
  - Exynos PCIe driver
  - Exynos Display Port driver
  - Samsung Framebuffer driver
  - Backlight Subsystem
Contents

- PCI Overview
- PCI Standard
- Exynos PCIe
- PCI in Device Tree
- PCI domain structure
- Mainline upstream
- Conclusion
PCI Overview
PCI Express is a High speed serial bus. 
PCI Express connects a computer to its attached peripheral devices. 
PCI Express specification was developed by the PCI Special Interest Group (PCI-SIG). 
PCI Express is the most popular bus standard in a computer these days.
History of PCI Express

- **Conventional PCI (1992/1993)**
  - Replaced MCA, EISA, and VLB buses
  - Provided better reliability than such legacy buses
  - Provided auto configuration via PCI Configuration space

- **PCI-X (1999)**
  - PCI-X enhanced conventional PCI for higher bandwidth.
  - Provided compatibility with conventional PCI
  - Replaced by PCI Express
History of PCI Express

- **PCI Express 1.0 (2003)**
  - In 2003, PCI-SIG introduced PCIe 1.0 with a transfer rate of 2.5GHz.

- **PCI Express 2.0 (2007)**
  - PCIe 2.0 was announced in 2007.
  - Doubled the transfer rate to 5GHz.
  - Improved the point-to-point data transfer protocol.

- **PCI Express 3.0 (2010)**
  - PCIe 3.0 was released in 2010.
  - The transfer rate is 8GHz.
  - Upgrades encoding scheme in order to enhance bandwidth.
Advantages of PCIe

- PCIe provides high bus bandwidth and high performance.
- I/O pin count is lower thanks to serial signal and embedded clock.
- PCIe reduces latency and cost.
- Scalability is supported, according to requirements of PCIe devices.
- Reliability is enhanced, by detailed error detection and error reporting.
- PCIe supports hardware I/O virtualization.
PCI Standard
PCI Components

- **Root complex**
  - Connects the CPU and memory to the PCIe switch or Endpoint

- **Switch**
  - Is a logical assembly of multiple PCI-to-PCI Bridge devices
  - There are one upstream port and some downstream ports.

- **Endpoint**
  - Can be the Requester or Completer
Address Spaces

PCI has three physical address spaces for PCI bus transaction: configuration, memory, and I/O address spaces.

Configuration Address Space
- ‘Configuration Address Space’ has been defined to support PCI hardware configuration.
- ‘Configuration Address Space’ provides access to 256 bytes of special configuration registers per PCI device.
- PCI Express introduces ‘Extended Configuration Space’ which extends the Configuration Space to 4096 bytes per Function.
- ‘Configuration Read’ command and ‘Configuration Write’ command are used to transfer data between each ‘Configuration Address Space’ of PCI devices.
Address Spaces

- **Memory Address Space**
  - ‘Memory address space’ is used for burst transactions.
  - Two different address formats are used: 32-bit address format and 64-bit address format.
  - ‘Memory Read’ command and ‘Memory Write’ command are used to transfer data in ‘Memory Address Space’.

- **I/O Address Space**
  - ‘I/O address space’ is for compatibility with the x86 architecture's I/O port address space. Future revisions of PCI Express specification may deprecate the use of ‘I/O Address Space’.
  - ‘I/O Read’ command and ‘I/O Write’ command are used to transfer data in ‘I/O Address Space’.
PCI devices have a set of registers referred to as 'Configuration Space'.

This configuration space is used for auto configuration of inserted PCI cards.

Configuration Space registers are mapped to memory locations.

PCI framework S/W should access the configuration space.
Device Identification

- ‘Vendor ID’ identifies the manufacturer of the device.
- ‘Device ID’ identifies the particular device.
- ‘Revision ID’ specifies a device specific revision identifier.
- ‘Class Code’ identifies the generic function of the device.
- ‘Header Type’ defines the layout of header.

Standard registers of Configuration Space Header
Device Control & Status

- ‘Command’ register provides control over a device's ability.
- ‘Status’ register is used to record status information.

Base Address Registers

- ‘Base Address’ registers are mapped into Memory Address Space or I/O Address Space.
Capabilities List

- Certain capabilities are supported by adding a set of registers to a linked list called the ‘Capabilities List’.
- Capabilities Pointer register points to the first item in the list.
- Each capability in the list consists of an 8-bit ID field assigned by the PCI SIG, an 8 bit pointer in configuration space to the next capability.

Example of Capabilities List

Standard registers of Configuration Space Header
Interrupts

- **INTx**
  - Conventional PCI introduced INTx.
  - There are 4 hard-wired Interrupt pins.
  - INTx is Level-triggered for robustness.

- **MSI** (Message Signaled interrupt)
  - MSI is in-band method of signaling an interrupt by writing a data to specific memory regions.
  - MSI provides 32 interrupts.
  - There is a slight performance enhancement.

- **MSI-X**
  - MSI-X provides up to 2048 interrupts.
Exynos PCIe
Exynos PCIe H/W

EXYNOS

- ARM-based SoC made by Samsung Electronics
- originates from the Greek words smart (exypnos) and green (prasinos)
- enhances the performance
- reduces the power consumption
Features

- There are two Root Complex controllers.
- PCI Express 2.0 is supported.
- There are 4 Lanes.
- MSI is available.
Exynos PCIe H/W

- **DesignWare PCIe Core IP**
  - DesignWare PCIe Core IP supports PCIe Express protocol layers such as Transaction layer, Data Link layer, and Physical Layer.
  - However, this PCIe Core IP does not support a full root complex.

- **Exynos Specific PCIe application controller**
  - Application controller is necessary to support full root complex functionality.
Exynos PCIe S/W

- **Common part**
  - controls DesignWare PCIe Core
  - can be shared by various PCIe drivers

- **Exynos specific part**
  - controls Exynos specific application controller
  - controls PHY controller
  - allocates resources
PCI in Device Tree
Device Tree

- Device tree usage is necessary, because Exynos supports Device Tree.
- Device tree is a data structure for describing hardware.
- Many aspect of the hardware can be described in a data structure that is passed to the operating system at boot time.
- The data structure itself is a simple tree of named nodes and properties.
- Nodes contain properties and child nodes.
- Properties are simple name-value pairs. The structure can hold any kind of data.

```plaintext
node {
    property = <value>;
    ....
}
```
SoC Specific Properties

- There are several SoC specific properties that describe the relationship between PCIe controller and other controllers.
- ‘Register’ property is necessary for accessing PCIe registers.
- ‘Interrupt’ property describes interrupt numbers that can be used for PCIe hardware.
- ‘clocks’ property and clock-names property describe clocks used for PCIe hardware.

Example of PCI in Device Tree

```
PCIe@290000 {
  reg = <0x290000 0x1000
        0x270000 0x1000
        0x271000 0x40>;
  interrupts = <0 20 0>, <0 21 0>, <0 22 0>;
  clocks = <&clock 28>, <&clock 27>;
  clock-names = "pcie", "pcie_bus";
}
```
PCI Address Translation

- The PCI address space is completely separate from the CPU address space, so address translation is necessary.
- Addresses for memory and I/O regions are usually same. It is called identical mapping.
- However, I/O regions should be different.
PCI Address Translation

- ‘Ranges’ property is used for PCI Address Translation.
- Each Red colored child address means PCI addresses.
- Each Blue colored child address means CPU addresses.
- Each child address in last columns means size of regions.
- ‘Ranges’ property should be added to ‘Device Tree’, in order to merge the PCI driver to the mainline kernel.

Example of PCI ‘ranges’ property in Device Tree
PCI domain structure
Multi-domain vs Single domain

- PCI domain is defined to be a set of PCI busses.

- There are two types of PCI domain structure.
  - Multi-domain structure
  - Single domain structure

- PCI domain structure should be chosen according to hardware architecture.
Multi-domain structure

- Multi-domain structure provides separate PCI domains.
- Each root complex has separate PCI Memory address regions.
- Multi domains structure is easy to implement.
Multi-domain

Multi-domain structure in Device Tree

- In multi PCI domains, there are separate nodes for each domain in Device Tree.
- Separate PCI Memory/IO/Configuration address regions are defined by each ‘ranges’ property.

Example of Device Tree for Multi-domain structure

```c
pcie@290000 {  
    ranges = <......>;
};
```

```c
pcie@2a0000 {  
    ranges = <......>;
};
```
### Single domain structure

- Single domain structure provides only one PCI domain.
- One PCI Memory address region can be shared.
- All ports are available under the single domain.
- Single domain structure can save memory regions.
- However, it is a little bit difficult to implement single domain structure.
Single domain structure in Device Tree

- In single PCI domain, there is only one node in Device Tree.
- PCI Memory and IO address regions are shared.
- However, Configuration regions are defined per port via ‘assigned-addresses’ properties.

Example of Device Tree for Single domain structure

```plaintext
pcie-controller@290000 {
    ranges = <.....>;
    pci@1,0 {
        assigned-addresses = <.....>
    };
    pci@2,0 {
        assigned-addresses = <.....>
    };
}
```
Exynos PCIe driver uses Multi-domain structure

- Exynos PCIe hardware was designed for multi-domain structure, not single domain structure.
- Each PCIe root complex controller cannot share PCIe memory regions.

![Exynos PCIe domain structure diagram]

- **Domain 0**
  - Bus 0
  - Device 0
  - PCIe 0

- **Domain 1**
  - Bus 0
  - Device 0
  - PCIe 1

- **CPU Address**
  - PCIe0 MEM (512MB)
  - PCIe1 MEM (512MB)

- **CPU Address Values**
  - 0x4000_0000
  - 0x6000_0000
  - 0x8000_0000
  - 0x8000_0000

35
Mainline upstream
Mainline upstream

- **Benefit of mainline upstream**
  - Mainline upstream increases the quality & robustness of the driver.
  - Mainline upstream reduces customer support for long term.
  - It is easy to back-port new patches from the latest mainline kernel.

- **Requirement of mainline upstream**
  - It is not easy to upstream the driver to mainline kernel.
  - It is necessary to keep up with discussions on mailing-list.
  - It is necessary to discuss actively with other Maintainers and contributors.
Mainline upstream for Exynos PCIe

- **New subdirectory for PCI drivers**
  - Previously, PCIe driver was added to the machine directory.
  - New subdirectory for PCIe drivers is created since 3.11 kernel.
  - Exynos PCIe driver was added to ‘drivers/pci/host/’ directory.

<table>
<thead>
<tr>
<th>arch directory</th>
<th>drivers directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch/arm/mach-*pci.c</td>
<td>drivers/pcie/host/pci-*c</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
</tbody>
</table>

- **Advantages with new subdirectory**
  - It is possible to avoid duplication by sharing common functions.
  - It reduces machine dependency. So, PCIe driver can be easily reused for different architectures.
Exynos PCIe upstream

- 1st patchset was submitted to mailing-list on March in 2013
- There were a lot of review comments from Maintainers and Developers from mailing-list.
- The patch was modified and submitted 10 times for about 4 months.
- Finally, 10th patchset was merged to 3.11 kernel.
Exynos PCIe in 3.11 kernel

- v3.11 kernel
  - PCI maintainer and Samsung SoC maintainer gave ACK.
  - Exynos PCIe driver was applied to ARM-SOC tree.
  - Exynos PCIe driver was merged to 3.11 kernel via ARM-SOC tree.
Exynos PCIe in 3.12 kernel

- v3.12 kernel
  - Exynos PCIe driver was split into two parts such as common part and Exynos specific part.
  - Common layer can be re-used for other vendors’ SoCs using DesignWare PCIe Core IP.
    - Re-usability
    - Stability & Robustness
Exynos PCIe in 3.13 kernel

- v3.13 kernel
  - MSI will be supported.
  - MAINTAINER of Exynos PCIe driver will be added.

Some minor patches will be merged via PCI tree.
PCI host merge strategy

- Three PCI drivers were merged to mainline kernel via ARM-SOC tree.
- New drivers will be merged to 3.13 kernel via PCI tree.
- According to the situation, PCI driver can be merged via one of these two trees.

Mainline upstream for PCI host

Mainline kernel (Linus Torvalds)

ARM-SOC tree (Arnd Bergmann, Olof Johansson)

PCI maintainer’s git (Bjorn Helgaas)

Marvell

Samsung

Nvidia

Freescale

Renesas

Armada

Exynos

Tegra

i.MX6

R-Car
PCI host maintainers

- PCI host maintainership was announced by PCI maintainer.
- Each PCI driver should be reviewed by each maintainer.
  - Samsung Exynos: Jingoo Han
  - Marvell Armada: Jason Cooper, Andrew Lunn, Gregory Clement
  - Nvidia Tegra: Thierry Reding
  - Freescale i.MX6: Shawn Guo
Conclusion
How to validate Exynos PCIe

- The following features are tested.
  - Network support
  - Storage support

- Network support
  - With PCIe LAN card, network works properly between Exynos and network.

```
How to validate Exynos PCIe

- The following features are tested.
  - Network support
  - Storage support

- Network support
  - With PCIe LAN card, network works properly between Exynos and network.
```
How to validate Exynos PCIe

- Storage support
  - PCIe SATA card and SATA disk are tested.
  - File read/write works properly between Exynos and SATA disk.
Future work

- **Power optimization**
  - Power optimization should be considered.

- **I/O Virtualization**
  - I/O Virtualization allows multiple operating systems within a single computer to natively share PCIe devices.
  - SR-IOV feature will be supported.

- **PCIe Switch**
  - PCIe Switch functionality will be tested with PCIe Switch devices.
Conclusion

- Samsung Exynos ARM SoC can support PCI Express.

- Mainline Linux kernel can support Exynos PCIe driver since v3.11 kernel.
Reference

- PCI Express Base Specification Revision 3.0 (PCI-SIG, 2010)
- http://devicetree.org/Device_Tree_Usage