linaro™

Arm Solutions at Lightspeed

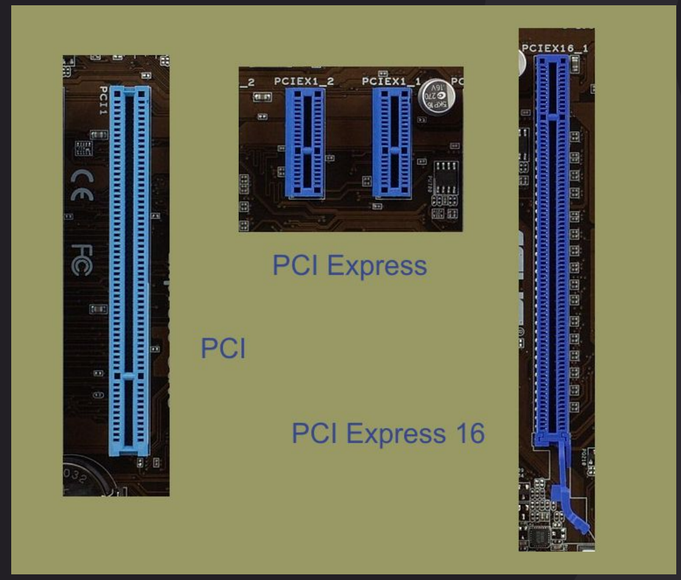
How (not) to write PCI(e) controller drivers in Linux Kernel

Manivannan Sadhasivam - Linaro

Why this talk?

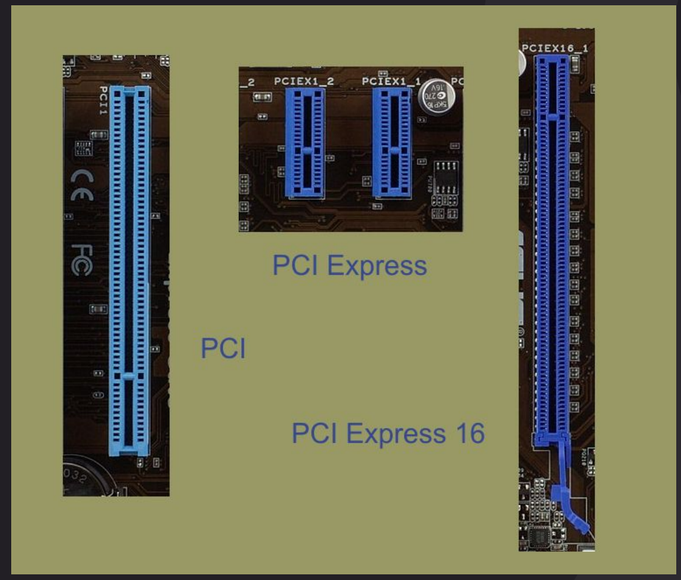
PCIe in a Nutshell

- Peripheral Component Interconnect



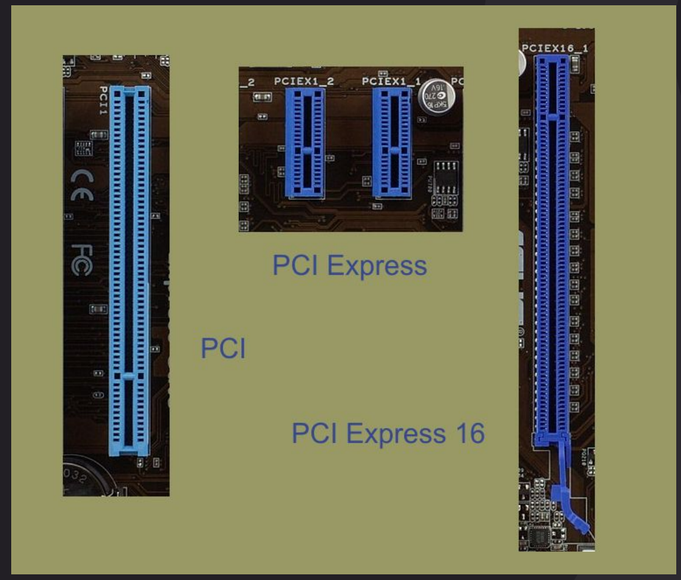
PCIe in a Nutshell

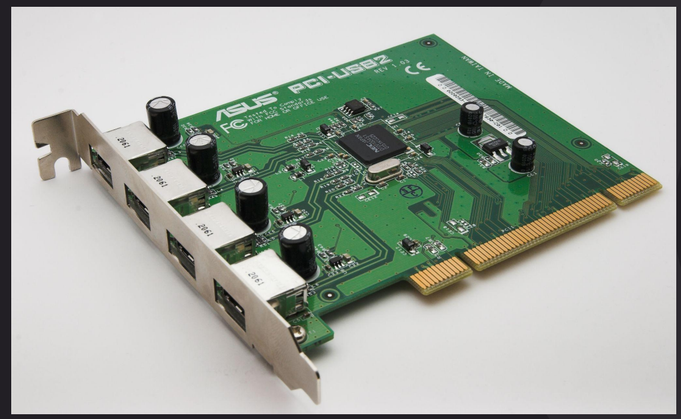
- Peripheral Component Interconnect
- PCIe - PCI Express (Software compatible with PCI)



PCIe in a Nutshell

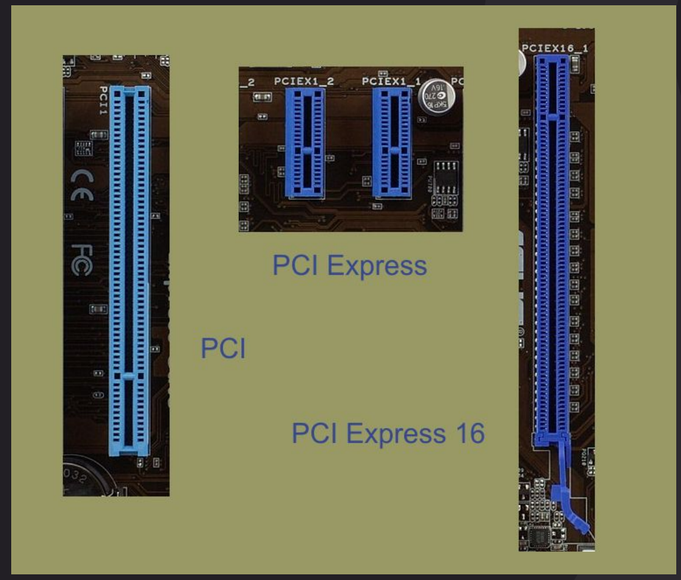
- Peripheral Component Interconnect
- PCIe - PCI Express (Software compatible with PCI)
- High speed expansion bus for PCs, Servers, Laptops, Mobiles
  - Marketed as Plug and Play (Hotplug)

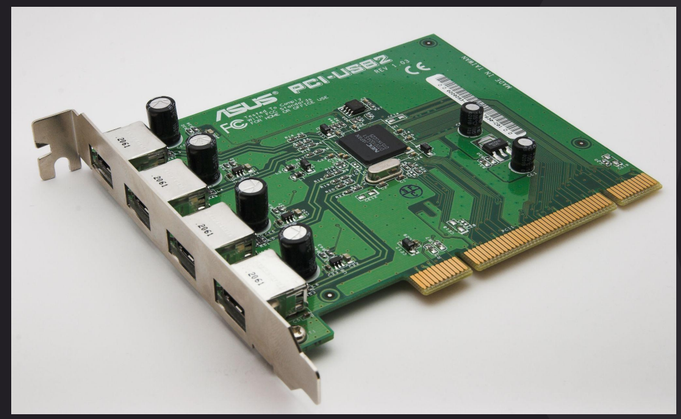




PCIe in a Nutshell

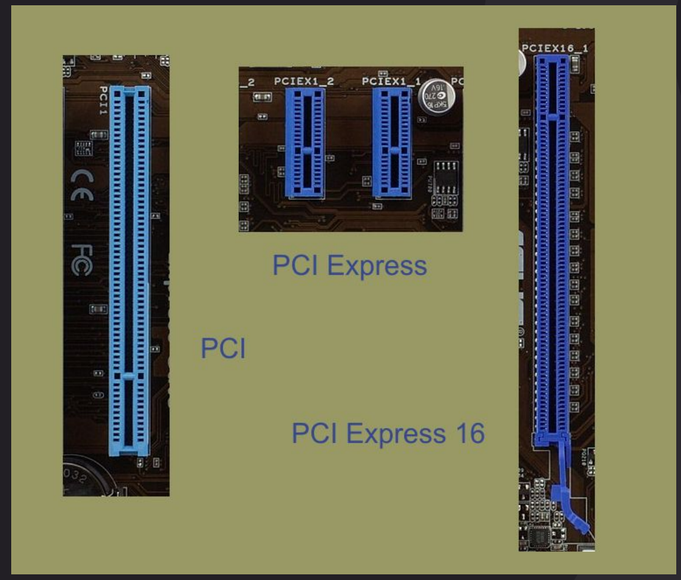
- Peripheral Component Interconnect
- PCIe - PCI Express (Software compatible with PCI)
- High speed expansion bus for PCs, Servers, Laptops, Mobiles
  - Marketed as Plug and Play (Hotplug)
- Specification developed by Intel
  - Later moved under PCI-SIG

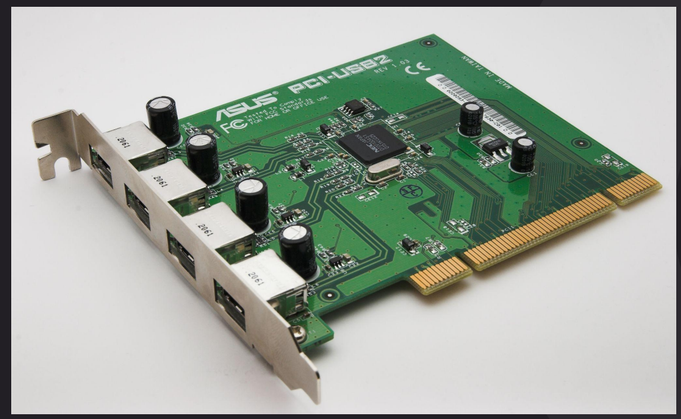




PCIe in a Nutshell

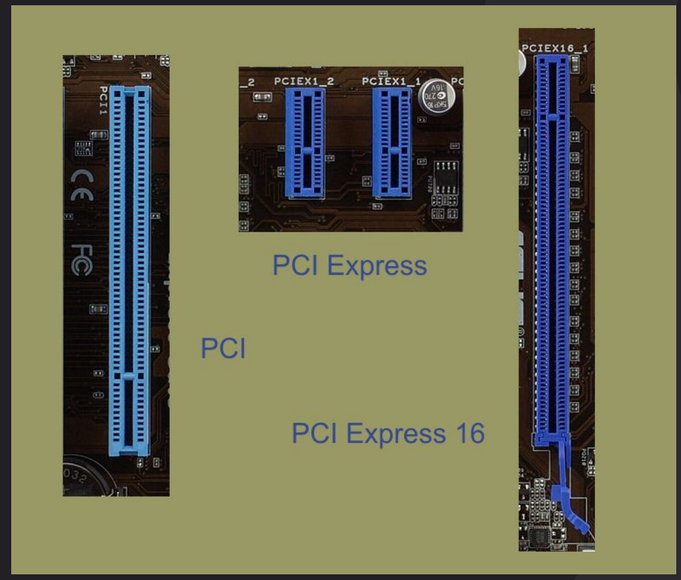
- Peripheral Component Interconnect
- PCIe - PCI Express (Software compatible with PCI)
- High speed expansion bus for PCs, Servers, Laptops, Mobiles
  - Marketed as Plug and Play (Hotplug)
- Specification developed by Intel
  - Later moved under PCI-SIG
- Works in Lanes (Tx/Rx differential pairs)

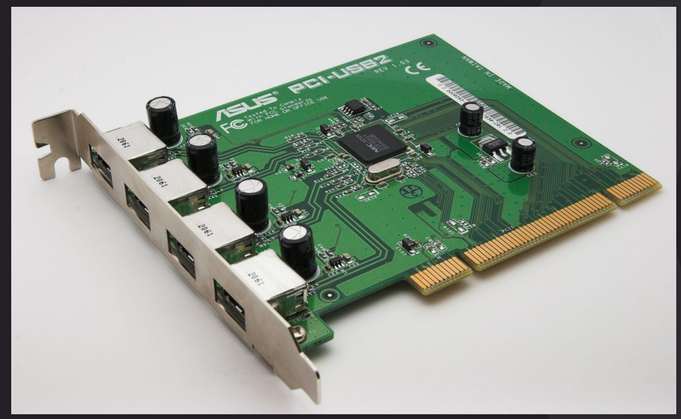




PCIe in a Nutshell

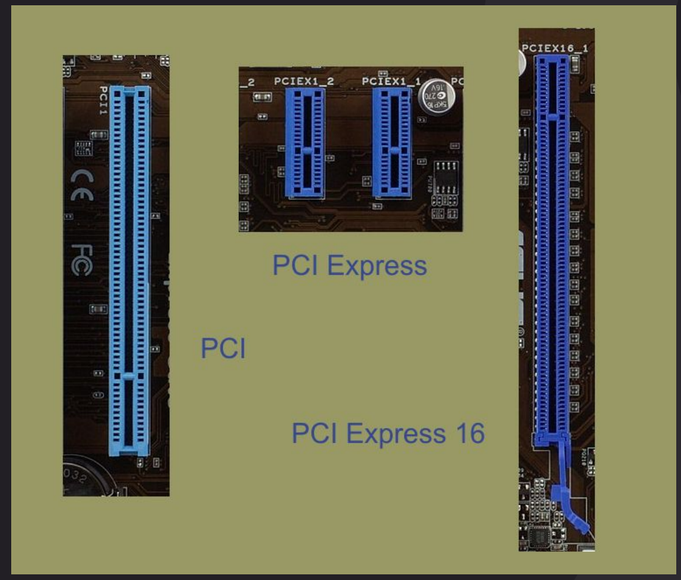
- Peripheral Component Interconnect
- PCIe - PCI Express (Software compatible with PCI)
- High speed expansion bus for PCs, Servers, Laptops, Mobiles
  - Marketed as Plug and Play (Hotplug)
- Specification developed by Intel
  - Later moved under PCI-SIG
- Works in Lanes (Tx/Rx differential pairs)
- Supports Power Management (PCI PM, ASPM)

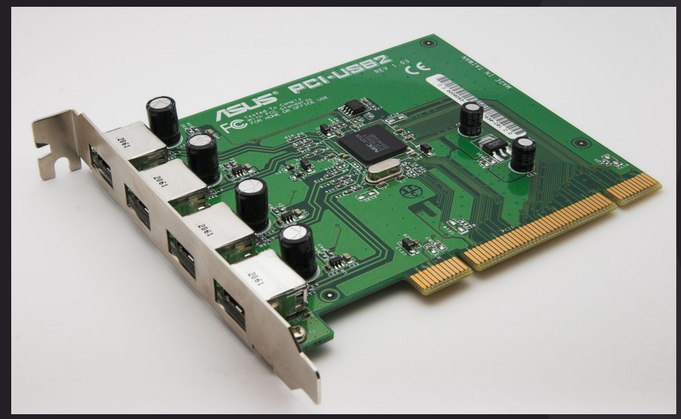




PCIe in a Nutshell

- Peripheral Component Interconnect
- PCIe - PCI Express (Software compatible with PCI)
- High speed expansion bus for PCs, Servers, Laptops, Mobiles
  - Marketed as Plug and Play (Hotplug)
- Specification developed by Intel
  - Later moved under PCI-SIG
- Works in Lanes (Tx/Rx differential pairs)
- Supports Power Management (PCI PM, ASPM)
- Supports I/O Virtualization (SR-IOV)

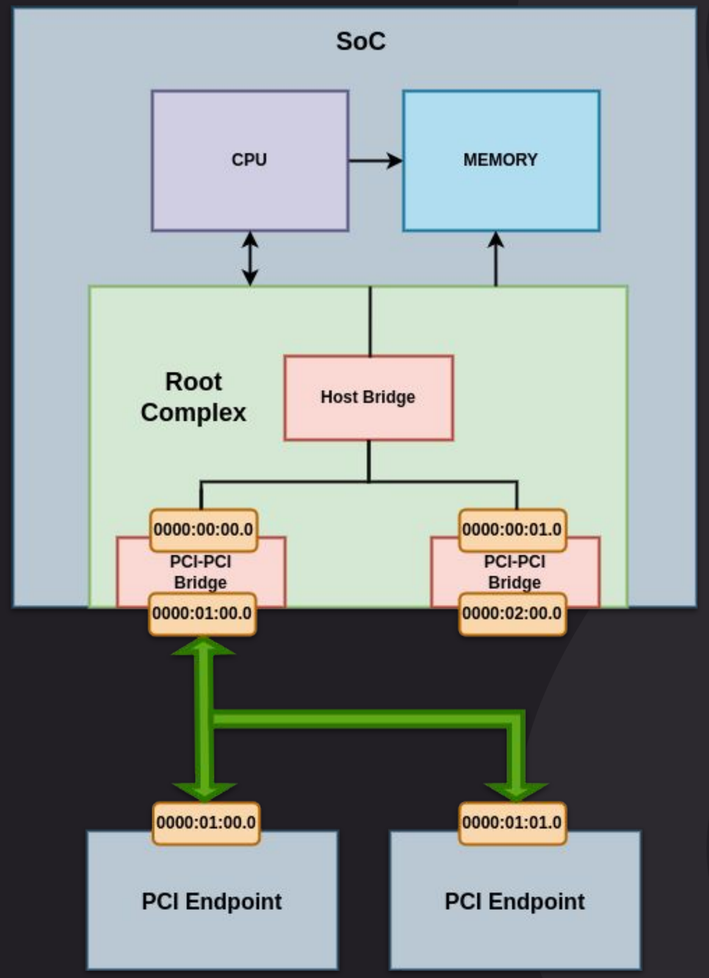




PCIe Architecture

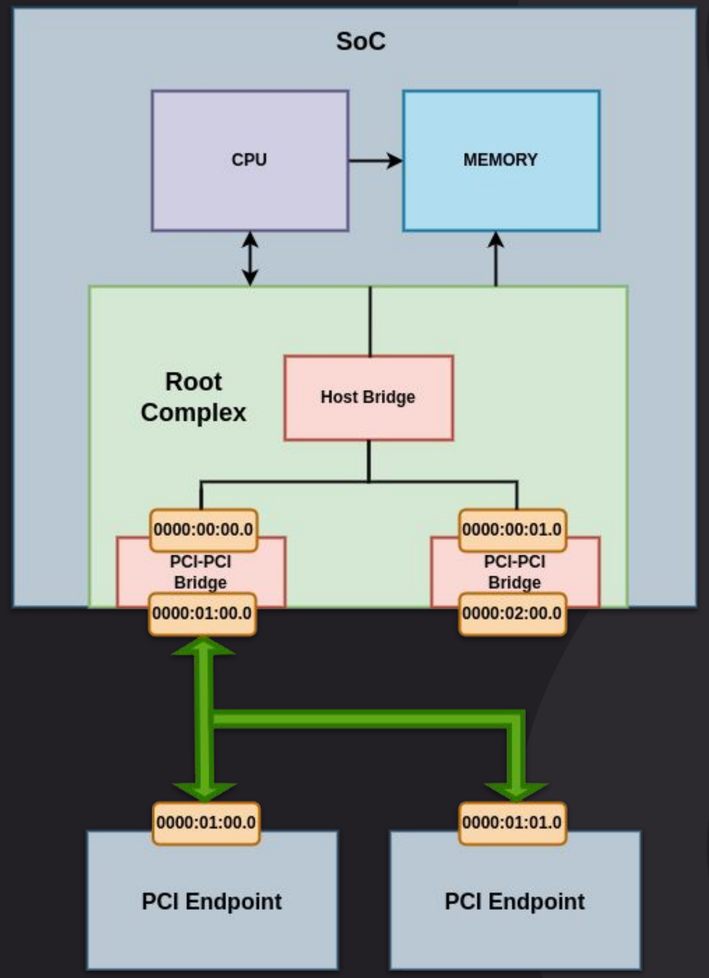
PCIe Architecture

- Point to Point topology
  - PCIe Root Complex - Host
  - PCIe Endpoint - Device



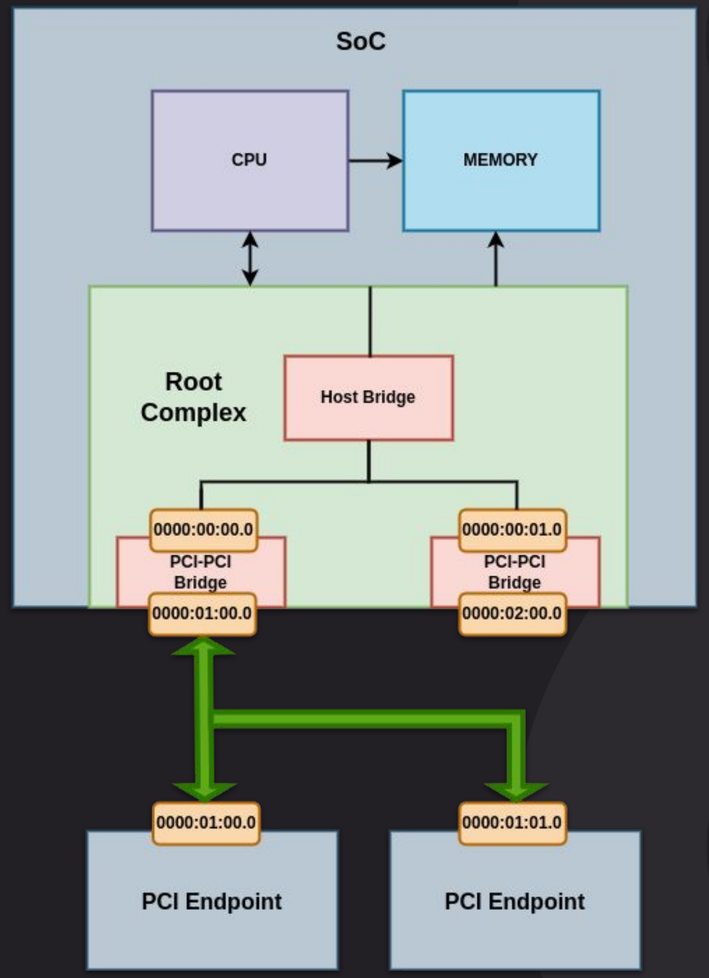
PCIe Architecture

- Point to Point topology
  - PCIe Root Complex - Host
  - PCIe Endpoint - Device
- Each PCIe device is identified by Bus Device Function (BDF) identifier
  - 8 bit Bus - 256 Busses
  - 5 bit Device - 32 Devices
  - 3 bit Function - 8 Functions



PCIe Architecture

- Point to Point topology
  - PCIe Root Complex - Host
  - PCIe Endpoint - Device
- Each PCIe device is identified by Bus Device Function (BDF) identifier
  - 8 bit Bus - 256 Busses
  - 5 bit Device - 32 Devices
  - 3 bit Function - 8 Functions
- PCIe Switches are often used for port expansion



PCIe in Linux Kernel

PCIe in Linux Kernel

- Linux kernel supported PCI from early v1.3 release and PCIe since v2.6

PCIe in Linux Kernel

- Linux kernel supported PCI from early v1.3 release and PCIe since v2.6
- Code organization:

PCIe in Linux Kernel

- Linux kernel supported PCI from early v1.3 release and PCIe since v2.6
- Code organization:
  - PCI Core drivers - drivers/pci/

PCIe in Linux Kernel

- Linux kernel supported PCI from early v1.3 release and PCIe since v2.6
- Code organization:
  - PCI Core drivers - drivers/pci/
  - PCIe Feature drivers - drivers/pci/pcie/

PCIe in Linux Kernel

- Linux kernel supported PCI from early v1.3 release and PCIe since v2.6
- Code organization:
  - PCI Core drivers - drivers/pci/
  - PCIe Feature drivers - drivers/pci/pcie/
  - PCI Controller drivers - drivers/pci/controllers/

PCIe in Linux Kernel

- Linux kernel supported PCI from early v1.3 release and PCIe since v2.6
- Code organization:
  - PCI Core drivers - drivers/pci/
  - PCIe Feature drivers - drivers/pci/pcie/
  - PCI Controller drivers - drivers/pci/controllers/
  - PCI Endpoint core - drivers/pci/endpoint/

PCIe in Linux Kernel

- Linux kernel supported PCI from early v1.3 release and PCIe since v2.6
- Code organization:
  - PCI Core drivers - drivers/pci/
  - PCIe Feature drivers - drivers/pci/pcie/
  - PCI Controller drivers - drivers/pci/controllers/
  - PCI Endpoint core - drivers/pci/endpoint/
  - PCI Client drivers - All over the place (Ethernet, WLAN, NVMe etc...)

PCIe in Linux Kernel

- Linux kernel supported PCI from early v1.3 release and PCIe since v2.6
- Code organization:
  - PCI Core drivers - drivers/pci/
  - PCIe Feature drivers - drivers/pci/pcie/
  - PCI Controller drivers - drivers/pci/controllers/
  - PCI Endpoint core - drivers/pci/endpoint/
  - PCI Client drivers - All over the place (Ethernet, WLAN, NVMe etc...)

PCI Controller Drivers

PCI Controller Drivers

- What they do?

PCI Controller Drivers

- What they do?
  - Initialize resources (clocks, regulators, PHY, power domain etc…)
    - devm_pci_alloc_host_bridge()

PCI Controller Drivers

- What they do?
  - Initialize resources (clocks, regulators, PHY, power domain etc…)
    - devm_pci_alloc_host_bridge()
  - Setup Root Port(s)

PCI Controller Drivers

- What they do?
  - Initialize resources (clocks, regulators, PHY, power domain etc…)
    - devm_pci_alloc_host_bridge()
  - Setup Root Port(s)
  - Setup MSI/MSI-X/INTx

PCI Controller Drivers

- What they do?
  - Initialize resources (clocks, regulators, PHY, power domain etc…)
    - devm_pci_alloc_host_bridge()
  - Setup Root Port(s)
  - Setup MSI/MSI-X/INTx
  - Setup Address Translation Unit (ATU) for MEM/IO region

PCI Controller Drivers

- What they do?
  - Initialize resources (clocks, regulators, PHY, power domain etc…)
    - devm_pci_alloc_host_bridge()
  - Setup Root Port(s)
  - Setup MSI/MSI-X/INTx
  - Setup Address Translation Unit (ATU) for MEM/IO region
  - Setup DMA

PCI Controller Drivers

- What they do?
  - Initialize resources (clocks, regulators, PHY, power domain etc…)
    - devm_pci_alloc_host_bridge()
  - Setup Root Port(s)
  - Setup MSI/MSI-X/INTx
  - Setup Address Translation Unit (ATU) for MEM/IO region
  - Setup DMA
  - Start LTSSM

PCI Controller Drivers

- What they do?
  - Initialize resources (clocks, regulators, PHY, power domain etc…)
    - devm_pci_alloc_host_bridge()
  - Setup Root Port(s)
  - Setup MSI/MSI-X/INTx
  - Setup Address Translation Unit (ATU) for MEM/IO region
  - Setup DMA
  - Start LTSSM
  - Enumerate endpoint devices
    - pci_host_probe()

4 Common Mistakes & Solutions

Mistakes and Solutions

Mistake 1

Mistakes and Solutions

Mistake 1

- Root ports not described in firmware
    - Devicetree mostly

Mistakes and Solutions

Mistake 1

- Root ports not described in firmware
  - Devicetree mostly
- Controller node ends up hosting root port properties

Mistakes and Solutions

Mistake 1

- Root ports not described in firmware
  - Devicetree mostly
- Controller node ends up hosting root port properties

```
pcie0: pcie@600000 {
        compatible = "qcom,pcie-sm8250";

        [...]

        iommu-map = <0x0   &apps_smmu 0x1c00 0x1>,
                    <0x100 &apps_smmu 0x1c01 0x1>;

        [...]

        phys = <&pciephy_0>;
        phy-names = "pciephy";

        [...]

        reset-gpios = <&tlmm 79 GPIO_ACTIVE_LOW>;
        wake-gpios = <&tlmm 81 GPIO_ACTIVE_HIGH>;

        [...]
};
```

Mistakes and Solutions

Solution

Mistakes and Solutions

Solution

- Move Root Port properties to Root Port

Mistakes and Solutions

Solution

- Move Root Port properties to Root Port

```
pcie0: pcie@600000 {
        compatible = "qcom,pcie-sm8250";

        [...]

        pcieport0: pcie@0 {
                device_type = "pci";
                reg = <0x0 0x0 0x0 0x0 0x0>;
                bus-range = <0x01 0xff>;

                #address-cells = <3>;
                #size-cells = <2>;
                ranges;

                iommu-map = <0x0   &apps_smmu 0x1c00 0x1>,
                            <0x100 &apps_smmu 0x1c01 0x1>;
                phys = <&pciephy_0>;
                phy-names = "pciephy";
                reset-gpios = <&tlmm 79 GPIO_ACTIVE_LOW>;
                wake-gpios = <&tlmm 81 GPIO_ACTIVE_HIGH>;
        };
};
```

Mistakes and Solutions

Mistake 2

Mistakes and Solutions

Mistake 2

- Slot/Endpoint power supplies defined in host bridge node

Mistakes and Solutions

Mistake 2

- Slot/Endpoint power supplies defined in host bridge node

```
pcie0: pcie@600000 {
        compatible = "qcom,pcie-sm8250";

        [...]
        vpcie12v-supply = <&pcie0_12v>;
        vpcie3v3-supply = <&pcie0_3p3v>;
        vpcie3v3aux-supply = <&pcie0_3p3v_aux>;
};
```

Mistakes and Solutions

Solution 1

Mistakes and Solutions

Solution 1

- Define slot supplies in Root Port node

Mistakes and Solutions

Solution 1

- Define slot supplies in Root Port node

```
pcie0: pcie@600000 {
        compatible = "qcom,pcie-sm8250";

        [...]

        pcieport0: pcie@0 {
                device_type = "pci";
                reg = <0x0 0x0 0x0 0x0 0x0>;
                bus-range = <0x01 0xff>;

                [...]

                vpcie12v-supply = <&pcie0_12v>;
                vpcie3v3-supply = <&pcie0_3p3v>;
                vpcie3v3aux-supply = <&pcie0_3p3v_aux>;
};
```

Mistakes and Solutions

Solution 1

- Define slot supplies in Root Port node
  - Rely on PWRCTRL drivers (CONFIG_PCI_PWRCTL*)
    - Works for standard supplies

```
pcie0: pcie@600000 {
        compatible = "qcom,pcie-sm8250";

        [...]

        pcieport0: pcie@0 {
                device_type = "pci";
                reg = <0x0 0x0 0x0 0x0 0x0>;
                bus-range = <0x01 0xff>;

                [...]

                vpcie12v-supply = <&pcie0_12v>;
                vpcie3v3-supply = <&pcie0_3p3v>;
                vpcie3v3aux-supply = <&pcie0_3p3v_aux>;
};
```

Mistakes and Solutions

Solution 2

Mistakes and Solutions

Solution 2

- Define Endpoint supplies in Endpoint node

Mistakes and Solutions

Solution 2

- Define Endpoint supplies in Endpoint node

```
pcie0: pcie@600000 {
        compatible = "qcom,pcie-sm8250";

        [...]

        pcieport0: pcie@0 {
                device_type = "pci";
                reg = <0x0 0x0 0x0 0x0 0x0>;
                bus-range = <0x01 0xff>;

                [...]

        wifi@0 {
                compatible = "pci17cb,1101";
                reg = <0x10000 0x0 0x0 0x0 0x0>;

                vddrfacmn-supply = <&vreg_pmu_rfa_cmn>;
                vddaon-supply = <&vreg_pmu_aon_0p59>;
                vddwlcx-supply = <&vreg_pmu_wlcx_0p8>;
                vddwlmx-supply = <&vreg_pmu_wlmx_0p85>;
                vddrfa0p8-supply = <&vreg_pmu_rfa_0p8>;
                vddrfa1p2-supply = <&vreg_pmu_rfa_1p2>;
                vddrfa1p7-supply = <&vreg_pmu_rfa_1p7>;
                vddpcie0p9-supply = <&vreg_pmu_pcie_0p9>;
                vddpcie1p8-supply = <&vreg_pmu_pcie_1p8>;
        };
};
```

Mistakes and Solutions

Solution 2

- Define Endpoint supplies in Endpoint node
  - Rely on PWRSEQ drivers (CONFIG_PCI_PWRCTL_PWRSEQ/CONFIG_POWER_SEQUENCING*)
    - Works for complex supplies

```
pcie0: pcie@600000 {
        compatible = "qcom,pcie-sm8250";

        [...]

        pcieport0: pcie@0 {
                device_type = "pci";
                reg = <0x0 0x0 0x0 0x0 0x0>;
                bus-range = <0x01 0xff>;

                [...]

        wifi@0 {
                compatible = "pci17cb,1101";
                reg = <0x10000 0x0 0x0 0x0 0x0>;

                vddrfacmn-supply = <&vreg_pmu_rfa_cmn>;
                vddaon-supply = <&vreg_pmu_aon_0p59>;
                vddwlcx-supply = <&vreg_pmu_wlcx_0p8>;
                vddwlmx-supply = <&vreg_pmu_wlmx_0p85>;
                vddrfa0p8-supply = <&vreg_pmu_rfa_0p8>;
                vddrfa1p2-supply = <&vreg_pmu_rfa_1p2>;
                vddrfa1p7-supply = <&vreg_pmu_rfa_1p7>;
                vddpcie0p9-supply = <&vreg_pmu_pcie_0p9>;
                vddpcie1p8-supply = <&vreg_pmu_pcie_1p8>;
        };
};
```

Mistakes and Solutions

Mistake 3

Mistakes and Solutions

Mistake 3

- Mapping each Outbound translation using ATU

Mistakes and Solutions

Mistake 3

- Mapping each Outbound translation using ATU

```
static void __iomem *dw_pcie_other_conf_map_bus(struct pci_bus *bus,
                                                unsigned int devfn, int where)
{
        struct dw_pcie_rp *pp = bus->sysdata;
        struct dw_pcie *pci = to_dw_pcie_from_pp(pp);
        struct dw_pcie_ob_atu_cfg atu = { 0 };
        int type, ret;
        u32 busdev;

        [...]

        ret = dw_pcie_prog_outbound_atu(pci, &atu);
        if (ret)
                return NULL;

        return pp->va_cfg0_base + where;
}

[...]

static struct pci_ops dw_child_pcie_ops = {
        .map_bus = dw_pcie_other_conf_map_bus,

        [...]
};
```

Mistakes and Solutions

Mistake 3

- Mapping each Outbound translation using ATU
  - Inefficient and adds latency

```
static void __iomem *dw_pcie_other_conf_map_bus(struct pci_bus *bus,
                                                unsigned int devfn, int where)
{
        struct dw_pcie_rp *pp = bus->sysdata;
        struct dw_pcie *pci = to_dw_pcie_from_pp(pp);
        struct dw_pcie_ob_atu_cfg atu = { 0 };
        int type, ret;
        u32 busdev;

        [...]

        ret = dw_pcie_prog_outbound_atu(pci, &atu);
        if (ret)
                return NULL;

        return pp->va_cfg0_base + where;
}

[...]

static struct pci_ops dw_child_pcie_ops = {
        .map_bus = dw_pcie_other_conf_map_bus,

        [...]
};
```

Mistakes and Solutions

Solution 1

Mistakes and Solutions

Solution 1

- Configure ECAM mode in bootloader

Mistakes and Solutions

Solution 1

- Configure ECAM mode in bootloader
- Pros
  - Ability to reuse existing controller drivers
    - pci-host-generic.c

Mistakes and Solutions

Solution 1

- Configure ECAM mode in bootloader
- Pros
  - Ability to reuse existing controller drivers
    - pci-host-generic.c
- Downside
  - Requires firmware change
  - Cannot reset the host bridge

Mistakes and Solutions

Solution 2

Mistakes and Solutions

Solution 2

- Configure ECAM mode in controller driver
  - Reference

Mistakes and Solutions

Solution 2

- Configure ECAM mode in controller driver
  - Reference
- Pros
  - Control lies with Linux Kernel
  - Can reset the host bridge

Mistakes and Solutions

Solution 2

- Configure ECAM mode in controller driver
  - Reference
- Pros
  - Control lies with Linux Kernel
  - Can reset the host bridge
- Downside
  - Requires a separate controller driver

Mistakes and Solutions

Mistake 4

Mistakes and Solutions

Mistake 4

- shutdown() callback often not defined

Mistakes and Solutions

Mistake 4

- shutdown() callback often not defined
    - Developers get confused with remove()

Mistakes and Solutions

Mistake 4

- shutdown() callback often not defined
  - Developers get confused with remove()
  - Messes up endpoint state machine during warm reboot

Mistakes and Solutions

Solution

Mistakes and Solutions

Solution

- Define shutdown() callback

Mistakes and Solutions

Solution

- Define shutdown() callback
    - Put the endpoint into D3Cold

Mistakes and Solutions

Solution

- Define shutdown() callback
  - Put the endpoint into D3Cold
  - Send PME_Turn_Off

Mistakes and Solutions

Solution

- Define shutdown() callback
  - Put the endpoint into D3Cold
  - Send PME_Turn_Off
  - Wait for PME_TO_Ack from endpoint

Mistakes and Solutions

Solution

- Define shutdown() callback
  - Put the endpoint into D3Cold
  - Send PME_Turn_Off
  - Wait for PME_TO_Ack from endpoint
  - Assert PERST#

Mistakes and Solutions

Solution

- Define shutdown() callback
  - Put the endpoint into D3Cold
  - Send PME_Turn_Off
  - Wait for PME_TO_Ack from endpoint
  - Assert PERST#
  - Remove power
    - Let the PWRCTRL driver take care of it!

linaro™ Arm Solutions at Lightspeed

linaro.org

Thank you

manivannan.sadhasivam@linaro.org
IRC: mani_s