Bangalore Linux Kernel Meetup - 2023

Saga of PCIe on ARM

Manivannan Sadhasivam | Linaro

Who am I?

- Senior Linux Kernel Engineer - Qualcomm Landing team
  - Working from Erode, Tamil Nadu
- Open Source contributor since 2016
  - Primarily focussed on Linux Kernel (> 650 patches in mainline)
- Linux Kernel Maintainership
  - PCI Endpoint Subsystem - Reviewer
  - Designware PCIe controller drivers
  - Designware eDMA drivers
  - Qualcomm MHI bus and NAND driver
  - ARM Bitmain, RDA Micro, Actions Semi SoCs

Disclaimer

This presentation is not a deep dive into PCIe on ARM

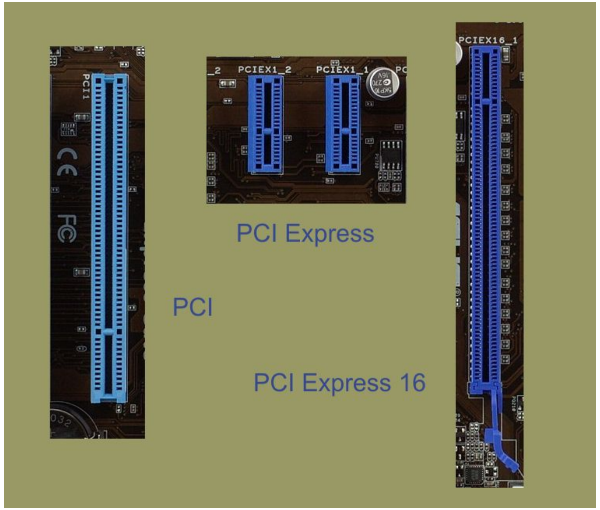
but rather breaking common misconceptions...

Agenda

- PCIe in a Nutshell
- PCIe in Linux Kernel
- PCIe support on x86 (Intel/AMD)
- PCIe support on ARM
- Key Takeaways

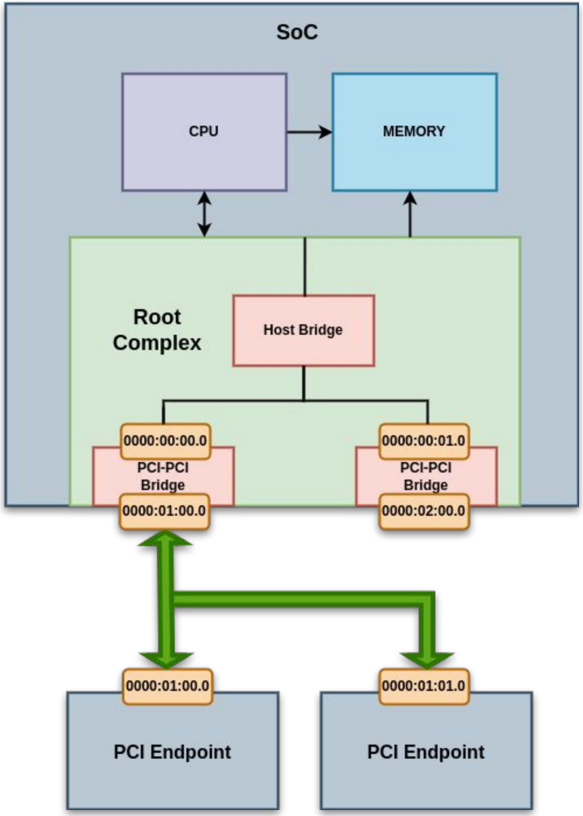
PCIe in a Nutshell

- **P**eripheral **C**omponent **I**nterconnect
- PCIe - **PCI E**xpress (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

- Point to Point topology
  - PCIe Root Complex - **Host**
  - PCIe Endpoint - **Device**
- Each PCIe device is identified by **B**us **D**evice **F**unction (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

- Linux kernel supported PCI from early v1.3 release and PCIe since v2.6
- Code organization:
  - PCI core - **drivers/pci/** (Common for both PCI and PCIe)
  - PCIe core - **drivers/pci/pcie/**
  - PCI/PCIe RC/EP controller drivers - **drivers/pci/controllers/**
  - PCI/PCIe Endpoint core - **drivers/pci/endpoint/**
  - PCI/PCIe drivers for devices - All over the place (Ethernet, WLAN, NVMe etc...)
- So connecting a PCIe device to a Linux machine should just **WORK?**

PCIe support on x86 (Intel/AMD)

- Most of the PCIe devices when connected to a x86 machine will just work
  - Is that because, Intel developed PCI in the early days? **Heh NO!!!**
- Then why?
- One of the reasons is **BIOS/ACPI**
  - In x86 machines, BIOS enumerates all the PCIe devices attached to the system during early boot.
    - Linux just queries the ACPI namespace to get the list of devices attached to the system instead of doing enumeration
  - All the resource allocations (I/O, MEM, IRQ) are handled by BIOS
  - It even configures the devices for power management (ASPM)

PCIe support on x86 (Intel/AMD)

- But it’s not just BIOS/ACPI, it is also about PCIe controllers
  - In x86 machines, the chip vendor will often integrate their own **in-house PCIe controllers** in the chip
  - So there are no integration issues between PCIe and CPU
- In x86, the issues with PCIe mostly come from hot pluggable PCIe devices as they are not controlled by BIOS
  - They may even take down the entire system if buggy*
- Since most of the heavy lifting is done by BIOS/ACPI, there is no need for a dedicated device driver in kernel for PCIe controllers

PCIe support on ARM

- Most of the ARM based SoCs have issues with PCIe
  - Is that because they all are ARM SoCs? **Heh NO!!!**
  - ARM only licenses CPU IPs to chip vendors
- Then why?
- One of the reasons is **NO BIOS/ACPI**
  - Most* of the ARM based SoCs are targeted for mobile and embedded use cases, so there is no BIOS/ACPI as in the PC world
  - All the resource allocations (I/O, MEM, IRQ) are handled by the OS with the help of devicetree
  - Often the resource provided in devicetree (MEM) is not sufficient for connecting external GPUs

PCIe support on ARM

- But it’s not just BIOS/ACPI, it is also about PCIe controllers
  - In ARM SoCs, the chip vendors often integrate **3rd party** PCIe controller IPs (like Synopsys Designware) with their ARM CPU
  - So there are IP integration issues
- Since there is no BIOS/ACPI, ARM-based SoCs always require a dedicated device driver for their controller
  - If the vendor doesn’t upstream the controller driver, then it will be outdated and turns out to be buggy
  - People who upstream the driver support, often get no support from the vendors*

PCIe support on ARM

- What’s wrong with a dedicated driver for PCIe controllers in **upstream**?
  - Need to manually control all the resources (clocks, regulators, IRQ) for both the controller as well as the devices
    - Most of the drivers won’t control these resources during suspend, resulting in poor system power management
  - The driver needs to be updated regularly to support each SoC (if required) from the vendor
    - But most of the vendors don’t add support for all SoCs
  - The driver may not support all PCIe features like ASPM, Hotplug, SR-IOV etc..

PCIe support on ARM

- What’s wrong with a dedicated driver for PCIe controllers in **downstream**?

EVERYTHING

PCIe support on ARM

- Even if the driver support is upstreamed and maintained, there are issues with packaging
  - There might be no proper distro support (Ubuntu, Debian, Fedora, etc…)
  - The vendors often provide ancient Yocto release or community based distros such as Armbian, Raspbian, Linaro debian etc…

Key Takeaways

- The PCIe issue on ARM SoCs is **NOT** due to ARM
    - The issue is mostly due to the chip vendors and their product use case (Embedded)
- Standard* BIOS/ACPI is needed to get a seamless PCIe experience
- Even with devicetree, the PCIe controller integration should not be buggy
    - There should also be a good upstream support for the device driver of the PCIe controller
- Standard distro support is also nice to have

Key Takeaways

- For connecting GPU cards to ARM boards, a standard PCIe connector on the board (not just M.2) along with a decent PCIe MEM range (> 1GiB) in devicetree is necessary
- There are also ARM based PCs and Laptops, started to emerge in the market and they have decent PCIe support
  - Socionext Synquacer (ACPI)
  - Lenovo Thinkpad X13s (only M.2 and devicetree)

Thank you

Linaro