Non-discoverable devices in PCI devices

Rob Herring
Lizhi Hou

The Problem

Modern PCI devices expose a whole slew of hardware behind a single PCI "device". While the PCI device itself is discoverable, everything behind it is not. These devices aren't fixed in what downstream devices are exposed nor their configuration.

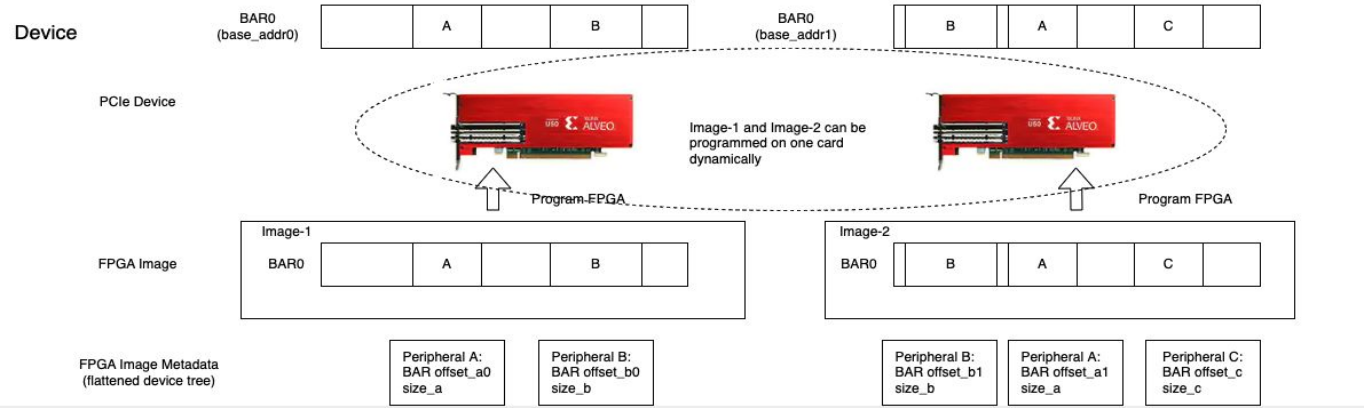
Want to reuse existing upstream drivers

Need a solution that works on DT and ACPI based systems

Use cases

- AMD Alveo FPGA PCIe cards
- Microchip LAN9662 Ethernet Controller SoC
- roadtest - platform devices behind virtio-pci device in UML
- CXL / MCTP over I2C on QEMU
- FTDI USB serial with I2C, SPI, GPIO

AMD Alveo Device Overview



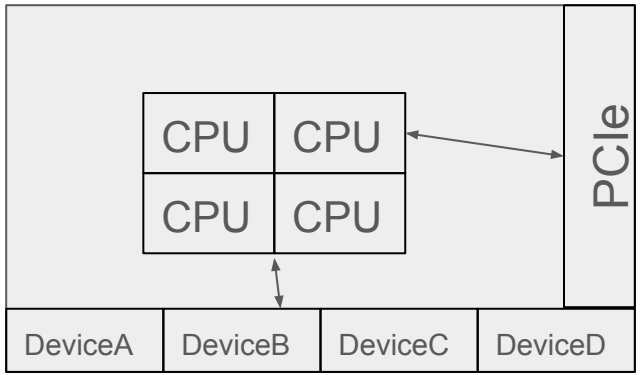
- Partial reconfigurable FPGA-based device
- Alveo device is PCI endpoint with multiple hardware peripherals exposed on its PCI BARs
- Each peripheral works independently and has its own driver in Linux kernel. For example
  - The AMD LogiCORE DMA (dma/xilinx/xdma)
  - The LogiCORE IP AXI UART(tty/serial/uartlite)

Use device tree for peripheral discovery

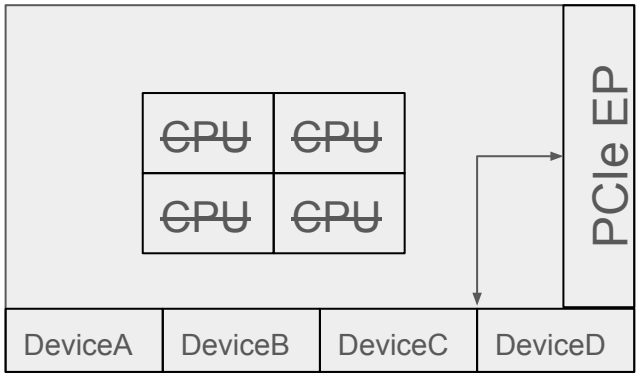
- The peripheral discovery needs to be data driven
  - Different FPGA images can be programmed on to the same Alveo device
  - FPGA images are created for different use cases or updated to fix hardware bugs
- Leverage device tree infrastructure for peripheral device discovery
  - FPGA Image bundles a flattened device tree to describe the peripherals
  - Assuming there is a device tree node for an Alveo PCI device, then the Alveo PCI driver overlays the FDT
- Benefits of using device tree
  - Device tree is widely used in Linux world to describe all kinds of hardware
  - Flattened device tree (FDT) is in compact format
  - Very rich FDT infrastructure in the Linux kernel
  - Align with FPGA framework to describe FPGA region

MicroChip LAN9662 SoC

Just your typical SoC. A bunch of on-chip devices and PCIe interface. All the non-discoverable parts described by Devicetree.



Same SoC, but cores are disabled and connected to host system as a PCIe Endpoint. Already have drivers for DeviceA, DeviceB, etc.



Solution

- A DT based host system is straight-forward
- Describing PCI devices in DT has been supported “forever”
- Below the PCI device are more non-discoverable devices (ISA?)
- Can be added dynamically with DT overlays
- 2 issues:
  - PCI devices are not populated in a DT by default. Location is not fixed.
    - Make the kernel generate DT nodes for PCI devices. Added in v6.6.
    - Similar to what OpenFirmware systems do providing discovered/configured PCI devices in the DT
    - Currently a kconfig option. Needs to be run-time.
  - Need to translate CPU->PCI->BAR->simple-bus
    - BAR# translation is similar to external chip select, but needs 3 address cells
    - PCI to downstream mapping likely PCI device specific

Solution cont.: What about ACPI systems?

- Why not use the same thing? No, really.
- What's needed:
  - A base DT to apply overlays to -> Already done when running DT unittest[1] and also needed in kunit[2]
  - PCI device DT nodes -> Already added for DT systems
  - PCI host bridge node(s) -> missing!
- Problems:
  - What happens when a device has/needs both a DT and ACPI node?
  - How to map interrupts, MSI, IOMMU, etc. from DT nodes to ACPI?
    - Keep generating DT resources until it works
    - Make the PCI device's driver figure it out
- What about ACPI overlays?
  - A debug feature
  - Doesn't solve re-using drivers

[1] https://lore.kernel.org/all/20230317053415.2254616-1-frowand.list@gmail.com/

[2] https://lore.kernel.org/all/20230327222159.3509818-1-sboyd@kernel.org/