FPGA Manager

State of the Union

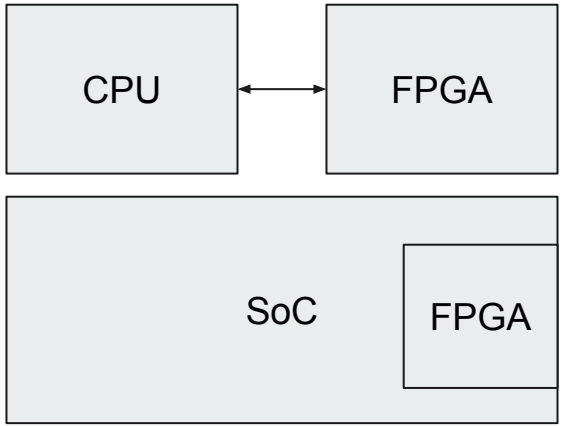
Moritz Fischer, National Instruments

$whoami

- Embedded Software Engineer at National Instruments
- Other stuff I do:
  - U-Boot, OE, Linux Kernel…
  - Co-Maintainer of FPGA Manager Framework
  - Random drive-by contributions to other projects

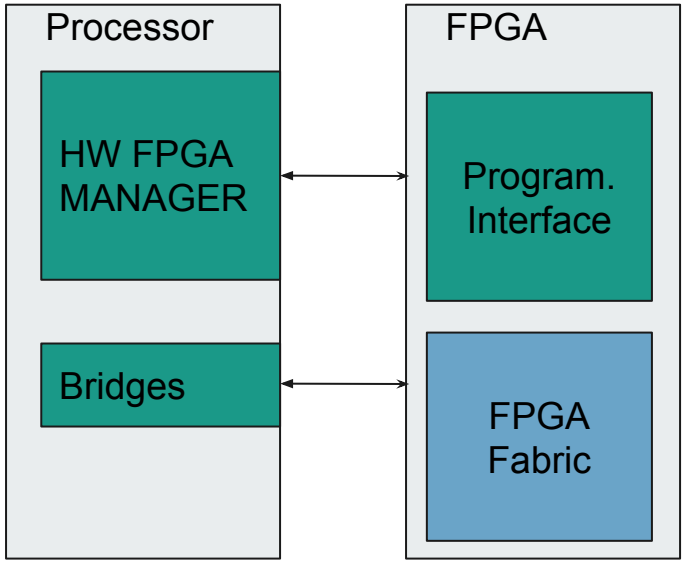
FPGA-Manager - The Problem

- Modern embedded systems often come with FPGAs of some form
- Accelerator or reconfigurable hardware
- Peripherals that need kernel drivers might be implemented in FPGA
- Drivers generally don't like if hardware goes away without telling them
- The sequence of things going away and coming back might be tricky



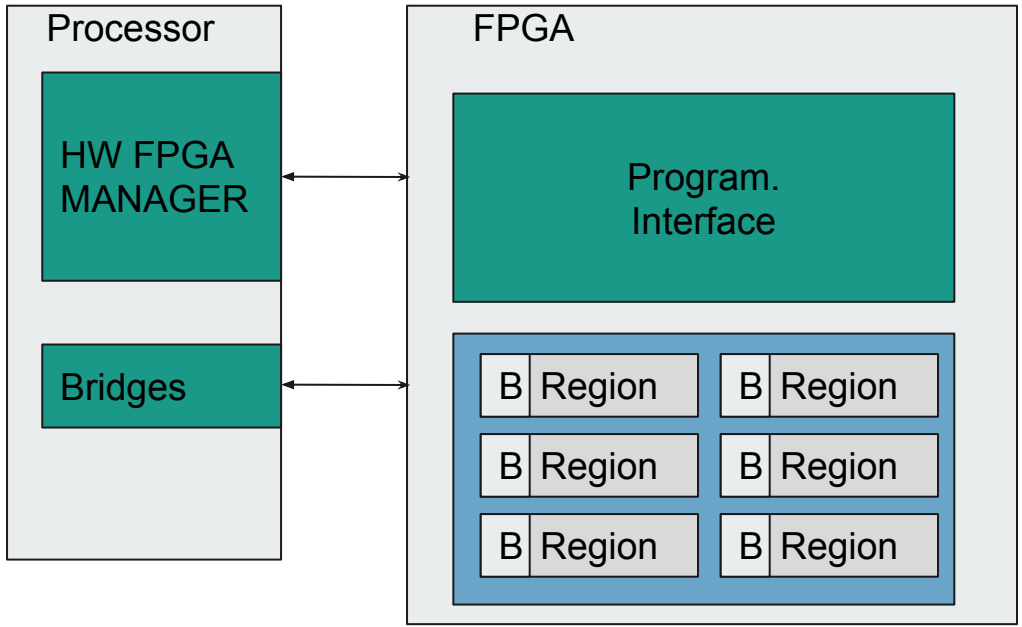
Overview - Full Reconfiguration

- System consists of HW FPGA manager that does the actual reconfiguration
- Optional bridges that isolate system during reconfiguration
- FPGA fabric that is being reconfigured
  - Discoverable
  - Non-discoverable



Overview - Partial Reconfiguration

- System consists of HW FPGA manager that does the actual reconfiguration
- Optional bridges that isolate system during reconfiguration
- FPGA fabric that is being reconfigured, subpartitioned into reconfigurable regions
- Usually:
  - Base bitstream
  - Regions on top



FPGA-Manager - History

- Vendor solutions with character device, cat
  - Userland control
  - Modules need to be loaded / unloaded manually
  - Potentially dangerous, not pretty, but work
- Basic support for Altera SoCFPGA and Xilinx Zynq merged in **Linux 4.4**
- Altera Arria 10 and Altera freeze bridge support since **Linux 4.10**
- TS-7300, Xilinx SPI, Xilinx Logicore PR decoupler, Lattice ice40, Altera PR IP and encrypted bitstream for Zynq support since **Linux 4.12**
- Altera passive serial SPI since **Linux 4.13**
- **Mailing list at linux-fpga@vger.kernel.org**

Fpga-Manager - Managers

- In charge of one or more regions
- Implements the *how* to program an FPGA
- Operations
  - write_init(manager, image_info)
  - write(manager, buf, size)
  - write_sg(manager, scatter_table)
  - write_complete(manager)

Fpga-Manager - Bridges

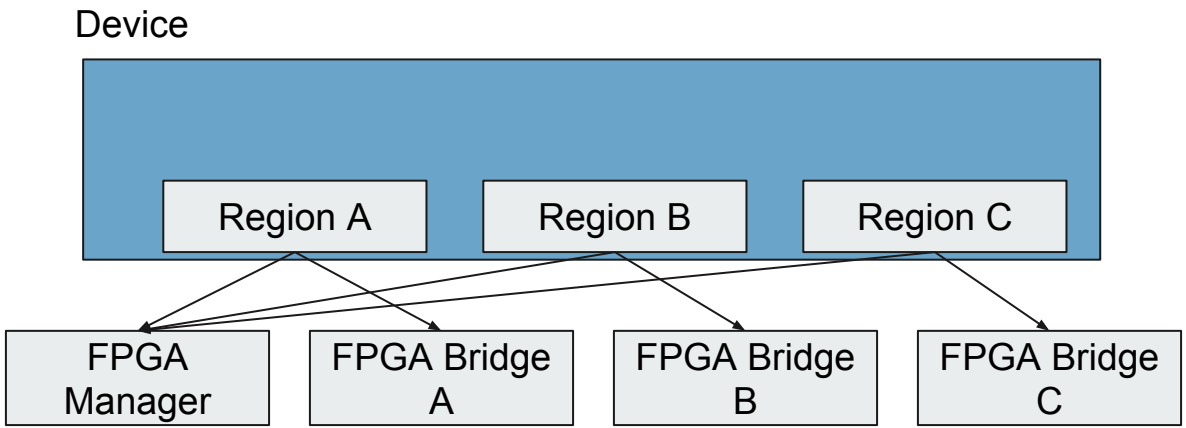
- In charge of isolating regions during reprogramming
- Implements the *how* to isolate a region
- Operations
  - enable_show(bridge)
  - enable_set(bridge, on)

FPGA-Manager - Regions

- Models a part of an FPGA that is reprogrammable
- Has a reference to a FPGA manager
- Has a list of bridges
- Currently only modifiable/usable via overlays, being refactored to include non-dt use case

FPGA-Manager - How does it fit together

- Region holds references to bridges and a manager
- Higher level code targets reprogramming a Region
- Think in terms of Regions instead of Managers and Bridges
- What vs. How



FPGA Manager - DT based regions

- Define regions in device tree
- Region got reference to manager
- Use overlay to modify properties
  - firmware-name
  - partial-fpga-config
  - encrypted-fpga-config
- **Caveat**: No userland interface for overlays in mainline

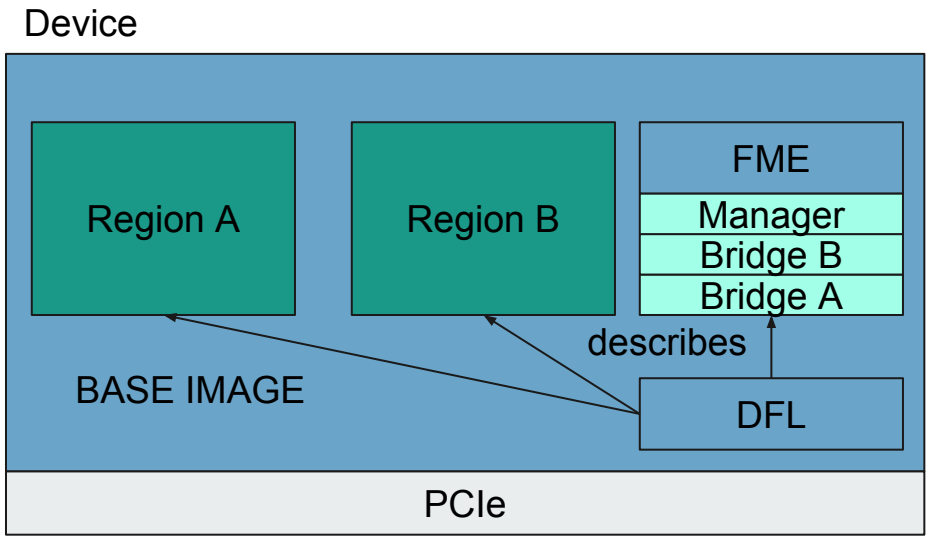
```
fpga_mgr: fpga-mgr@ff706000 {
        compatible = "altr,socfpga-fpga-mgr";
        interrupts = <0 175 4>;
};

fpga_region0: fpga-region0 {
        compatible = "fpga-region";
        fpga-mgr = <&fpga_mgr>;
        firmware-name = “foo.rbf”;

        gpio@10040 {
                compatible = "altr,pio-1.0";
                reg = <0x10040 0x20>;
                altr,gpio-bank-width = <4>;
        };
};
```

FPGA-Manager- Regions (revamped)

- Currently, separating out device-tree code into of_region
- New interface
  - fpga_region_register(struct device *dev, struct fpga_region *region)
  - fpga_region_unregister(struct fpga_region *region)
  - int fpga_region_program_fpga(struct fpga_region *region,
    struct fpga_image_info *image_info);
- Allows you to ‘bring your own region’ e.g. as part of a device, as part of your image info, ‘bring your own buffer’

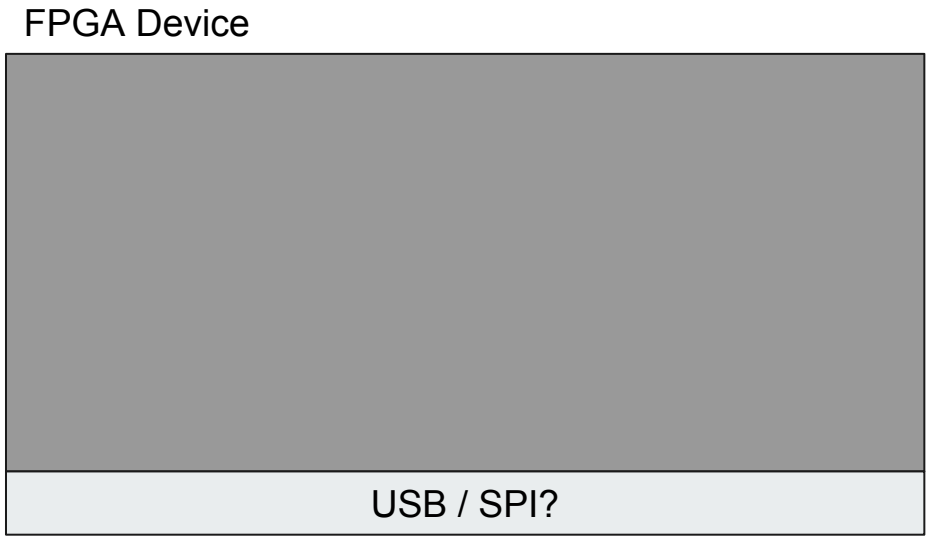
FPGA Manager - Intel DFL based PCIe (upcoming)

- Device Feature List
- PCIe base device binds driver
  - Creates bridges by parsing DFL
  - Create regions by parsing DFL
  - Create manager by parsing DFL
- ioctl() allows for partial reconfiguration of particular regions
- Working towards generalization



FPGA Manager - USB/SPI based?

- DT doesn't work for all cases
- Can't rely on device peripherals being self describing
- How to deal with non-discoverable device?
  - Binary header attached to bitstream?
  - DFL?
  - FDT based?



FPGA-Manager - the good

- Representing systems with device tree overlays and regions works pretty well
- Hardware support is growing

FPGA-Manager - the bad

- Currently doesn’t work well for non-dt platforms

FPGA-Manager - the ugly

- Making it work even on DT platforms relies on out of tree code
  - DT overlays
  - Sysfs / ioctl
- No userland API
- Some use cases flat out not supported at the moment
  - Discoverable bus in FPGA, e.g.

Not all is doom and gloom, here's my cat watching TV ...

Questions?

email:
moritz.fischer@ettus.com / mdf@kernel.org

gpg-fingerprint:
135A 2159 8651 9D68 DA5B C3F1 958A 4C04 7304 62CC

github:
mfischer