FPGA and Devicetree

Linux Plumbers' Conference 2018

Alan Tull (Intel) / Moritz Fischer (National Instruments)

Problem Statement that FPGA Manager addresses

- FPGAs are hardware that can be reconfigured at runtime
- There are **little to no restrictions** on what can be implemented bus-wise in an FPGA
- Users might want to reprogram either the full FPGA (**full reconfiguration)** or parts (**partial reconfiguration**) of the FPGA **at any point** during runtime
- When an FPGA is reprogrammed, a single FPGA image adds many interconnected devices to a bus
- FPGA Manager framework presents APIs on several levels to deal with the sequencing and dependencies for programming FPGAs under the control of the Linux Kernel

FPGA Manager components

- A **FPGA manager** deals at the lowest level with how to program an FPGA with a new piece of firmware (bitstream)
- A **FPGA region** represents part of (or an entire) FPGA that can be reprogrammed
- FPGA regions sometimes need to be isolated from surrounding logic while being reprogrammed. We model this using **FPGA bridges**

More info:
https://elinux.org/images/5/5b/FPGAs-under-Linux-Alan-Tull-v1.00.pdf

https://www.kernel.org/doc/html/v4.19/driver-api/fpga/index.html

How does DT fit in there?

- Most of FPGA designs are fundamentally not discoverable (SPI, I2C, MMIO …)
- DT is made to describe non-discoverable hardware
- DT code largely assumes static devicetree
- DT overlays allow to work with runtime changes in the devicetree
- When FPGA manger was being developed (v4.4) DT overlays looked like a perfect fit for DT based FPGA systems

- **Problem**: Most (DT) code predates DT overlays
- **Result**: As of 4.19 FPGA Manager does not have a workable userspace interface for DT based systems

```
[...]
mgr0: fpga-manager{
        compatible = “foo-mgr”;
        [...]
};

fpga_bridge0: fpga-bridge {
        compatible = “foo-bridge”;
};

fpga_region0: fpga-region {
        compatible = “fpga-region”;
        bridges = <&fpga_bridge0>;
        fpga-mgr = <&mgr0>;
        [...]
};

[...]
```

+Overlay

```
[...]
mgr0: fpga-manager{
        compatible = “foo-mgr”;
        [...]
};

fpga_bridge0: fpga-bridge {
        compatible = “foo-bridge”;
};

fpga_region0: fpga-region {
        compatible = “fpga-region”;
        bridges = <&fpga_bridge0>;
        fpga-mgr = <&mgr0>;

        firmware-name = “mybitstream.bin”;

        gpio9: gpio-controller {
                compatible = “bar,gpio”;
        };

        uart7: uart {
                compatible = “foo,uart”;
                [...]
        };
};

[...]
```

- The overlay targets the **FPGA Region** to be programmed
- Applying DT overlay will:
  - Specify an FPGA image for programming [1]
  - Specify information about the image type such as full vs partial, encrypted, compressed, ...
  - Describe the HW added in the FPGA image

[1] More info on bindings: Documentation/devicetree/bindings/fpga/fpga-region.txt

DT Overlays for FPGA - The Original Plan$^{\text{TM}}$

DT Overlays for FPGA - The Original Plan^{TM}

```
[...]
mgr0: fpga-manager{
        compatible = “foo-manager”;
        [...]
};

fpga_bridge0: fpga-bridge {
        compatible = “foo-bridge”;
};

fpga_region0: fpga-region {
        compatible = “fpga-region”;
        bridges = <&fpga_bridge0>;
        fpga-mgr = <&mgr0>;
        [...]
};

[...]
```

+Overlay

```
[...]
mgr0: fpga-manager{
        compatible = “foo-manager”;
        [...]
};

fpga_bridge0: fpga-bridge {
        compatible = “foo-bridge”;
};

fpga_region0: fpga-region {
        compatible = “fpga-region”;
        bridges = <&fpga_bridge0>;
        fpga-mgr = <&mgr0>;

        firmware-name = “mybitstream.bin”;

        gpio9: gpio-controller {
                compatible = “bar,gpio”;
        };

        uart7: uart {
                compatible = “foo,uart”;
                [...]
        };
};

[...]
```

1. **of_overlay_apply()** calls **of_overlay_notify(OF_OVERLAY_PRE_APPLY)**
2. **of_fpga_region_notify_pre_apply()** looks at overlay, parses **firmware-name** property and other properties that affect FPGA programming [1].
3. **fpga_region_program_fpga()**
4. If previous step succeeds, notifier returns success and overlay changeset gets applied to live-tree. Otherwise notifier returns error and overlay is rejected.

[1] More info on bindings: Documentation/devicetree/bindings/fpga/fpga-region.txt

How does the DT overlay get into the kernel in the first place?

- Configfs interface proposed by Pantelis Antoniou **"OF: DT-Overlay configfs interface (v7)"**
  - Generic Interface that allows application of DT overlays to any node from userland
  - Geert Uytterhoeven somewhat unofficially maintains that in his tree [1]
  - Widely used, e.g. upstream Yocto kernel ships it by default
  - Discussion around why this is not a good idea [2], to summarize: A lot of things break if you apply to random nodes, we need a mechanism to lock down where we apply the overlays
- Bake it into your FPGA image at known location (i.e. **make your FPGA design discoverable**)
  - Block RAM in FPGA is expensive for common dtbo sizes
  - **Doesn't work for all FPGA designs**, especially existing ones

References:

[1] https://git.kernel.org/pub/scm/linux/kernel/git/geert/renesas-drivers.git/log/?h=topic/overlays

[2] https://lkml.org/lkml/2017/10/18/609

Ideas on how to lock down where we apply overlays

- Alan submitted a RFC
  [RFC 1/2] of: overlay: add whitelist [1]
  - Driver centric, i.e. driver declares it’s ok with overlays
  - Feedback mostly around implementation
    - Rob: Function naming
    - Rob/Frank: Implementation: Use flag vs actual list
    - Frank: Use DT connectors

- DT connectors RFC by Pantelis Antoniou
  [RFC 0/3] Portable Device Tree Connector
  - Presentation on that at [2]
  - Problem statement by Frank [3] & [4]
  - At this point more or less conceptual?
  - Most discussions around actual implementation of connectors (tooling, dtc, …)

References:

[1] https://lkml.org/lkml/2017/12/7/1462
[2] https://elinux.org/images/d/d0/Panto.pdf
[3] https://lkml.org/lkml/2016/7/4/472
[4] https://lkml.org/lkml/2016/6/30/734

Discussion: Whitelisting for DT overlays

- Is this something we generally wanna look at?
- Can we salvage Alan’s RFC and make this work?
- Should the drivers declare themselves able to deal with overlays?

Discussion: Connectors for FPGA

- Is this something we generally wanna look at?
- Recent discussion at Linaro Connect [1] suggests GPIO has at least nexus part figured out?
- Offline discussion between Alan & me seemed like the concept proposed for connectors could work somewhat
- **Caveat:** FPGAs mostly care about the MMIO / arbitrary hardware case which seems to benefit the least from connectorized approach

[1] https://connect.linaro.org/resources/yvr18/yvr18-404

Let’s keep the discussion going

- Offline after this talk / hallway
- linux-fpga (linux-fpga@vger.kernel.org) and devicetree (devicetree@vger.kernel.org) mailing lists
- Alan Tull (atull@kernel.org)
- Moritz Fischer (mdf@kernel.org)