Introduction

The FPGA subsystem supports reprogramming FPGAs dynamically under Linux. Some of the core intentions of the FPGA subsystems are:

- The FPGA subsystem is vendor agnostic.
- The FPGA subsystem separates upper layers (userspace interfaces and enumeration) from lower layers that know how to program a specific FPGA.
- Code should not be shared between upper and lower layers. This should go without saying. If that seems necessary, there's probably framework functionality that can be added that will benefit other users. Write the linux-fpga mailing list and maintainers and seek out a solution that expands the framework for broad reuse.
- Generally, when adding code, think of the future. Plan for reuse.

The framework in the kernel is divided into:

FPGA Manager

If you are adding a new FPGA or a new method of programming an FPGA, this is the subsystem for you. Low level FPGA manager drivers contain the knowledge of how to program a specific device. This subsystem includes the framework in fpga-mgr.c and the low level drivers that are registered with it.

FPGA Bridge

FPGA Bridges prevent spurious signals from going out of an FPGA or a region of an FPGA during programming. They are disabled before programming begins and re-enabled afterwards. An FPGA bridge may be actual hard hardware that gates a bus to a CPU or a soft ("freeze") bridge in FPGA fabric that surrounds a partial reconfiguration region of an FPGA. This subsystem includes fpga-bridge.c and the low level drivers that are registered with it.

FPGA Region

If you are adding a new interface to the FPGA framework, add it on top of an FPGA region.

The FPGA Region framework (fpga-region.c) associates managers and bridges as reconfigurable regions. A region may refer to the whole FPGA in full reconfiguration or to a partial reconfiguration region.

The Device Tree FPGA Region support (of-fpga-region.c) handles reprogramming FPGAs when device tree overlays are applied.