Reprogrammable Hardware under Linux

Alan Tull
Altera Corp Embedded Linux Group
ELC Dublin 2015
Introductions

- Background in driver development for ARM and x86
- Linux driver developer 15 years
- Altera Corporation
- Embedded Linux Group in Austin, TX
- Driver support for SoCFPGA = FPGA on a SoC
FPGA Basics

- FPGA = Field Programmable Gate Array
- Designed to be configured after manufacturing
- Array of programmable logic blocks ("Fabric")
  - Also I/O, DSPs, and other specialized blocks
- Design in some Hardware Design Language (HDL) compiled into a bitstream
- **Bitstream** is used to program the FPGA
- Fully or partially reconfigured
FPGAs in a system (hardware)

- Server with FPGA’s on PCIe card
- Embedded CPU including the FPGA
Altera SoC FPGA

- **ARM Cortex-A9**
- **FPGA**

### Hard Processor System (HP5)

<table>
<thead>
<tr>
<th>Component</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM Cortex-A9</td>
<td>NEON™/FPU L1 Cache</td>
</tr>
<tr>
<td>USB OTG (x2)</td>
<td></td>
</tr>
<tr>
<td>Ethernet (x2)</td>
<td></td>
</tr>
<tr>
<td>L2 Cache</td>
<td></td>
</tr>
<tr>
<td>GPIO</td>
<td></td>
</tr>
<tr>
<td>PC (x2)</td>
<td></td>
</tr>
<tr>
<td>Quad 5PI Flash Control</td>
<td>64-KB RAM</td>
</tr>
<tr>
<td>SPI (x2)</td>
<td>SPI Debug/Trace</td>
</tr>
<tr>
<td>CAN (x2)</td>
<td></td>
</tr>
<tr>
<td>NAND Flash</td>
<td>SD/SDIO/MM C</td>
</tr>
<tr>
<td>SPI (x2)</td>
<td>DMA (x8)</td>
</tr>
<tr>
<td>UART (x2)</td>
<td></td>
</tr>
<tr>
<td>Shared Multiport DDR SDRAM</td>
<td>HP5 tp FPGA</td>
</tr>
<tr>
<td>Controller</td>
<td>FPGA to HPS</td>
</tr>
<tr>
<td>FPGA Config</td>
<td></td>
</tr>
</tbody>
</table>

### FPGA

- 28LP Process
- 8-input ALMs
- Variable-precision DSP
- M10K memory and 640 bit MLABs
- Fractional PLLs

### Other Components

- Hard Multiple DDR SDRAM Controller
- Hard PCI Express® (PCIe)
- 3.4, 5, 6, 8, and 10 Gbps Transceivers

© 2015 Altera Corporation — Public
FPGA under an OS – A Few Examples

- FPGAs can be used as accelerator or as reconfigurable hardware
- Page processing in printers
  - Altera CycloneV SoC FPGAs
  - Pipelining processing the pages
  - Reconfiguring FPGA to switch out processes in smaller FPGA
- Server acceleration
  - ½ width Open Compute servers, each with one 2 Xeons + 1 StratixV
- FPGA Based Hardware Soft IP
  - such as uarts, gpio, mailbox, triple speed ethernet, etc
Problem Statement:
- No standard way of configuring FPGAs in Linux kernel
- Each FPGA driver has custom interface

Proposed FPGA Manager framework
- Common configuration interface
- Different FPGAs supported
- Bitstreams are FPGA device specific, but interface is shared
- Separate interfaces suited for use models
FPGA Manager Framework History

Both biggest FPGA manufacturers (Altera, Xilinx) involved

My first version (in Altera GIT) ~ April 2013
- Low Level Ops
- FPGA specific low level drivers register ops with the framework
- Userspace driven interface
  - `cat image.rbf > /dev/fpga0`

Xilinx
- v1: `cat image.rbf > /sys/class/fpga/fpga0/fpga`
- v2: `echo image.rbf > /sys/class/fpga/fpga0/firmware`

My next version ~ Aug 2014
- Core Framework with no userspace interface
- Device Tree Overlays support
- Lots of mailing list feedback
- 11 versions so far since then with other interfaces (sysfs, configfs).
- kept coming back to DT overlays
FPGA Manager Framework History - Interfaces

- Interfaces driven by userspace
  - cat’ing the image file to the driver
    - Either writing to the devnode or to a sysfs file
  - Writing the name of the image file to a sysfs file
    - firmware loads it the file, gets loaded to FPGA

- Workable, sort of, but not pretty

- Giving userspace control of a low level function
  - Stability (easily crash)
  - Security

- Userspace still had to modprobe the drivers
  - Drivers had to be modules

- Bridges also controlled from userspace?
FPGA Manager Framework – Current Proposal

FPGA Manager Framework

Shared Interfaces
Not specific to any FPGA

Higher Level Interface

FPGA Manager Framework

Higher Level Interface

Low Level FPGA Driver

Low Level FPGA Driver

FPGA Device Specific Code
(configuring)
Proposed High Level Interface

- Simple FPGA Bus
- Uses Device Tree Overlays
  - adding/removing to the live tree
- Overlay could drive:
  - FPGA getting programmed with the right image
  - Bridges being enabled/disabled
  - Drivers getting probed
- This is normal kernel stuff, we get most of this for free
FPGA on a SoC (simplified)

- CPU programs FPGA
  - FPGA Manager

- Bridges allow memory mapped access between FPGA and host processor
  - Logic in FPGA can have registers
  - DMA
Reconfiguration

- **Bitstream** compiled from hardware design
- CPU uses **FPGA Manager** to write the FPGA
- **Bridges** allow memory mapped access FPGA ↔ CPU
  - Must be disabled during programming
- Linux drivers for hardware on FPGA
  - Register access is through bridges
  - DMA access through bridges
  - Stop access during driver remove
  - Remove drivers before disabling bridges
FPGA configuration sequence

1. Disable Bridges
2. CPU writes bitstream to FPGA
3. Enable Bridges
4. Load Drivers
Full Reconfiguration

Control hardware bridges

Diagram:
- Processor
  - FPGA Manager HW
  - Bridges
- FPGA
  - Programming Interface
  - FPGA Fabric
Needs bridges within the fabric for each region
FPGA Manager Framework – API and ops

API functions to talk to interface

Higher Level Interface

FPGA Manager Framework

Low Level FPGA Driver

Ops to talk to low level driver

Higher Level Interface

Low Level FPGA Driver
FPGA Manager Framework

- Exposes methods for reconfiguring FPGA
  - Manufacturer agnostic API functions
- Low level drivers register with framework
  - ops for FPGA specific stuff
- No user space interface (other than status in sysfs)
Framework – 6 API functions

- Register/Unregister a low level driver:
  - `fpga_mgr_register`
  - `fpga_mgr_unregister`

- Get/Put a reference to a particular FPGA Manager:
  - `of_fpga_mgr_get`
  - `fpga_mgr_put`

- Write a bitstream to a FPGA from a buffer
  - `fpga_mgr_buf_load`

- Write a bitstream to a FPGA using firmware class
  - `fpga_mgr_firmware_load`
Using FPGA Manager Framework API to configure a FPGA

Get a reference to a specific FPGA manager:
- `struct fpga_manager *mgr = of_fpga_mgr_get(dn);`

Load the FPGA from a buffer in RAM or from firmware.
- `fpga_mgr_buf_load(mgr, flags, buf, count);`
- `fpga_mgr_firmware_load(mgr, flags, "image.rbf");`

Put the reference
- `fpga_mgr_put(mgr);`
FPGA Manager Framework ops

Ops for the write cycle (in call order):

1. write_init
   - Do FPGA specific steps to prepare device to receive bitstream

2. write
   - Send a bitstream buffer to FPGA

3. write_complete
   - Do FPGA specific steps after configuration

Two other ops:

- state
  - Return FPGA state from low level driver

- fpga_remove
  - Called if the fpga manager driver is removed
Simple FPGA Bus

- Simple FPGA Bus
- Other Higher Level Interface
- FPGA Manager Framework
- Low Level FPGA Driver
- Low Level FPGA Driver
Simple FPGA Bus

- Built on top of the FPGA Manager Framework
- Uses Device Tree Overlays
- Handles:
  - Bridges
  - FPGA configuration
  - Drivers
- `configfs` interface:
  - `mkdir /config/device-tree/overlays/1`
  - `echo "overlay.dtbo" > /config/device-tree/overlays/1/path`
Simple FPGA Bus (2)

An overlay will have this information:
- Which FPGA
- Which image file
- Which bridges to enable and disable
- Child nodes for devices that are about to get loaded

Load order – when you load an overlay, this happens:
1. Disable bridges
2. Load FPGA
3. Enable bridges
4. Probe drivers (call of_platform_populate)

Unload order is in reverse order

Currently on the mailing list, may need some consideration about how to represent bridges
More Considerations – Firmware

‒ The FPGA Manager uses the firmware layer to load the whole image into RAM
  ‒ Then the FPGA Manager Framework can load to the FPGA.
  ‒ Then release the firmware and get the RAM back.

‒ On an embedded platform, RAM can be very small while the FPGA image can be large. Some users may run up against this.

‒ A kernel method to stream firmware files without loading the whole file would be great.
On the mailing list

- FPGA Manager (soon v12)
- simple-fpga-bus
Acknowledgements

- Pantelis Antoniou - for his work on Device Tree Overlays
- Thanks for all the feedback on the mailing list!
Exciting Free Stuff – Win a SoCFPGA eval board

- Drop of your business card at the Altera booth #33 for a chance to win an Atlas SoC evaluation kit
- Meet Altera Linux people
- Check out Altera’s technology showcase at booth #33
Thank You