Customizable ARM Designs and Linux
Abstracting The Hardware At The Right Level

Dipl.-Math. techn. Detlev Zundel (dzu@denx.de)
DENX Software Engineering GmbH
Kirchenstr. 5
82194 Gröbenzell, Germany

May 15, 2014
1. **Hardware**
   - Meeting Today’s Trends
   - The Device

2. **Software**
   - Kernel
   - Device Drivers

3. **Give me something to play with**
Currently we see these technologies growing towards each other:

**General processors**
- Software programmable
- Great flexibility
- Poor power efficiency
- Few application-specific features

**Application-specific**
- Hard-wired, not programmable
- Poor flexibility
- Great power efficiency
- Many contain embedded processors
Where the trends meet

These trends meet in SoC FPGAs with Hard Processor Systems (HPS)

- Hardware programmable
- Great flexibility
- Good power efficiency
- Set of commonly required interfaces (Ethernet, \( I^2C \), SPI, CAN, ... )
ARM Cortex A9 + Altera FPGA

ARM Processor System
- Dual Core ARM Cortex-A9
- MPCore Processor
- Hard Memory Controller
- Peripherals

28-nm FPGA
- Cyclone V
- Arria V

DENX
Customizable ARM Designs and Linux
System Architecture

- Processor
  Dual-Core ARM Corex-A9 cpu
  4000 DMIPS (@800 MHz per core)
  NEON coprocessor with double-precision FPU
  32 KiB I- and 32 KiB D L1 $ shared 512 KiB L2 $

- Multiport SDRAM controller
  Up to 533 MHz DDR3
  Up to 400 MHz DDR2

- High-bandwidth on-chip interface
  125 Gbps HPS-to-FPGA
  125 Gbps FPGA-to-SDRAM
Software Support

Even though the device can run “bare metal” applications just fine, GNU/Linux with its already existing excellent ARM and SMP support will be the main focus of the device. Linux support was developed from the beginning in the regular community context and - with some help from DENX - fed into the upstream projects, i.e. U-Boot and the mainline Linux kernel.
Operating System

Tasks of an operating system:

- Manage hardware resources including the processors, memory, data storage and I/O devices
- Handling interrupts generated by the I/O devices
- Share I/O between many programs on the system
Linux kernel and device drivers

The Linux kernel needs drivers for the hardware present in the system:

- HPS subsystem blocks like Ethernet and CAN (fixed)
- Functional blocks in the FPGA part (variable)

The first group of drivers is already mostly in mainline, whereas the second group needs to be provided by you.
To access a peripheral, we have three approaches:

- Full Linux driver representing the device at its functional level - this is what we aim for.
- uio framework to export the memory mapped registers into userspace.
- Bypass the whole operating system by using \texttt{mmap} on \texttt{/dev/mem}. This is fraught with peril and can only be a quick hack in early project phases.
“Real” Driver

The actual peripheral is completely abstracted by the operating system by appropriate “functional level” APIs. For networking adapters for example, this is the well known “socket” interface.
Mapping The Device To Userspace

Although tempting, exposing the peripheral memory map bypasses the operating system completely and has important limitations:
General Approach For New Blocks

- Find Linux driver class that matches target hardware as close as possible. Here are some examples
  
  - GPIO  GPIO chip
  - Video In  Video for Linux 2 (V4L2)
  - Network  Netdevice
  - Analog Data  Industrial I/O (IIO)
  - Crypto  Crypto Alg
  
  ...

- Implement a driver for that class as an “OF device” that gets instantiated through the device tree.

- Profit
Isn’t There An Easier Way?

Shirking a real driver is a tempting but short-sighted approach:
- Implementation knowledge escapes into userspace
- Userspace becomes tied to the implementation and is thus non-portable
- Userspace cannot span multiple hardware platforms
- Hooking up other software pieces will need patches
- Patched packages become maintenance burden

All sums up to a significant “technical debt” that will only get more expensive the longer it is not fixed.
But Linux Drivers Are Non-Trivial?

Device drivers living up to the kernel standards are indeed non trivial. Remember that designing and supporting new hardware previously was the domain of chip manufacturers and even they have a hard time getting the Linux drivers right.

- Leverage the knowledge of the kernel community and develop the driver as a mainline driver as early as possible
- Find competent partners early on
Proof Of Concept: Integrating Display and Touch

This demo is using an FPGA IFI display controller and an LNT touch display attached to an FPGA I2C controller. All FPGA blocks have regular Linux drivers so the setup needs zero patches in tslib or the Qt framework.
EBV SoCrates board

- Altera Cyclone V SoC device:
  - 5CSEBA6U23C7N
  - 110 K LEs
  - 112 DSP Blocks
  - 5.1 Mbit RAM

- Interfaces:
  - 1GiB Ethernet
  - USB 2.0 OTG
  - CAN
  - SPI
  - I²C

- Memory:
  - 128MiBx32 DDR3 Memory
  - μSD Card Slot
  - 2x EPCQ256 Configuration Device
DENX MCV Module

- Different versions of Altera Cyclone V SoC device:
  - 25 KLE - 110 KLE
  - 36 - 112 DSP Blocks
  - 6x 2.5 GiB transceivers

- Interfaces:
  - 1 GiB Ethernet
  - USB 2.0 OTG, CAN, SPI, I²C

- Memory:
  - 1 GiB DDR3 Memory
  - 256 MiB non-volatile storage
  - EPCQ256 Configuration Device
Online Community

Welcome to RocketBoards.org

Accelerate development by exchanging ideas with the open source community. Discover the right resources for your embedded solutions. Use one of the community-oriented development boards for the SoC, targeted for people interested in exploring and prototyping their applications.

- Get started now

Getting started
- Altera SoC Development Board
- Arrow SoC64 Evaluation Board
- EBY SoC64 Evaluation Board
- Marlini Heis SoC Evaluation Kit

Projects
- Register a Project
- Project Hosting
- List All Projects
- Search Projects

Community
- Mailing Lists
- Git Web
- Get Involved
- Register on RocketBoards.org

What's new
Fixed USB peripheral driver
Posted 14 Jun 2013 - 18:20

Upgrade to kernel version 3.9

Expand Your Open Source Embedded Universe with RocketBoards.org
Posted 02 May 2013 - 21:56

Upgrade to kernel version 3.8
Further reading

- http://rocketboards.org Online Community
- http://rocketboards.org/gitweb Git Repositories (U-Boot, Linux, Yocto)
- http://www.denx.de/wiki/ELDK-5 The ELDK 5 toolchain and target distribution
- http://www.denx.de/wiki/U-Boot Das U-Boot