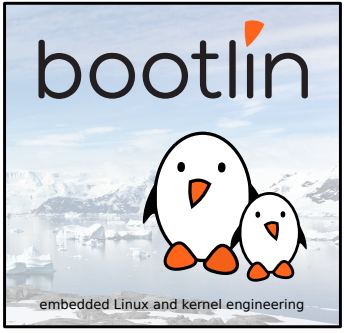
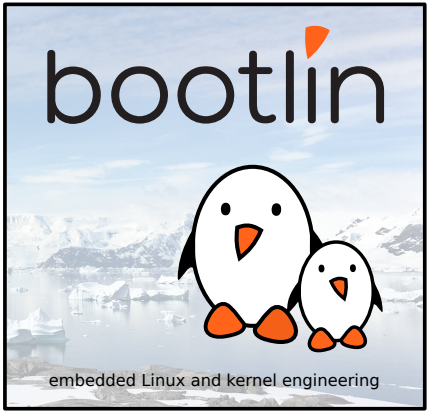
AMP on Cortex A9 with Linux and OpenAMP

Gregory CLEMENT
Bootlin *gregory.clement@bootlin.com*



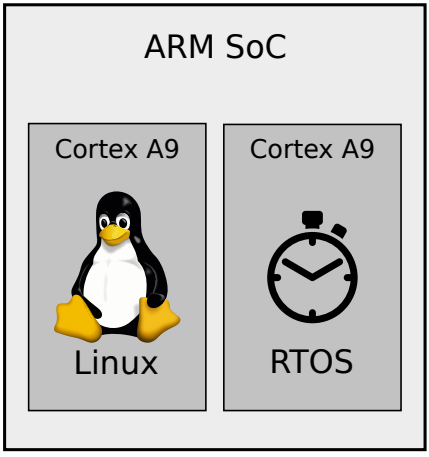
Gregory CLEMENT

- Embedded Linux engineer and trainer at Bootlin
  - Embedded Linux **expertise**
  - **Development**, consulting and training
  - Strong open-source focus
- Open-source contributor
  - Contributing to **kernel support** for the Armada 370, 375, 38x, 39x and Armada XP ARM SoCs and Armada 3700, 7K/8K ARM64 SoCs from Marvell.
  - Co-maintainer of mvebu sub-architecture (SoCs from Marvell Engineering Business Unit)
  - Living near **Lyon**, France



AMP on **Cortex A9**, why ?

- Usually, with Linux, multiple CPU cores of the same kind are used by the same OS in **SMP**
- However, one may want to use one of the cores separately, the system becomes **asymmetric**
- The usual reason for it is running a **Real Time OS** with dedicated resources on one of the cores
- This allows reducing the disturbance on the second core to the minimum and in the same time benefiting of a full feature OS: Linux



Images from Wikipedia, credit:
- Tux: lewing@isc.tamu.edu Larry Ewing - CC0
- Chronometer: http://www.simpleicon.com/ SimpleIcon - CC BY 3.0

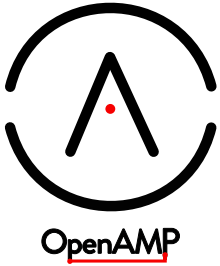
Overview of this talk

- Presentation of **OpenAMP**
- How Linux kernel communicates with external OS
- Adapting Linux kernel and OpenAMP for Cortex A9
- Pro, Cons and alternatives

Presentation of **OpenAMP**

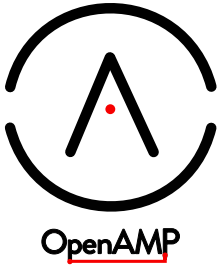
OpenAMP purpose

- **A**symmetric **M**ulti-**P**rocessing
  - In **SMP**: CPU cores treated equally, none reserved for special purposes
  - In **AMP**: cores treated in different way, either by choice or because cores are different
- Modern SoCs comes with variety of core, main cores runs Linux in SMP, other smaller cores run RTOS, bare metal: need to make them communicate
- **OpenAMP** provides a way to manage interactions between operating environments
- The project seeks to standardize them through open source solutions
- **OpenAMP** licence is BSD 3-clause.



OpenAMP code

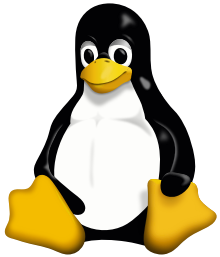
- Modern build automation software: **CMake**
- Split in two parts:
  - **libmetal**: this library is the abstraction layer that is specific to the system where OpenAMP will run
  - **libopen_amp**: this one provides the API allowing to communicate between the different systems
- Can be run as bare metal or as a library used by an RTOS or even Linux



How Linux kernel communicates with external OS

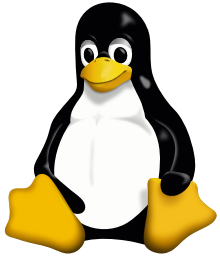
Linux kernel communication with other systems: **remoteproc**

- To communicate with the other core in an heterogeneous configuration, the Linux kernel provides a framework: **Remote Processor Framework** aka **remoteproc**
- This framework allows to control the remote processor: power on, load firmware, power off.
- It also allows to setup **virtio** devices with the remote processor



Linux kernel communication with other systems: **rpmsg**

- **rpmsg** is a **virtio**-based messaging bus that allowing communicating with remote processors
- **virtio** is a framework supporting virtualization and provides a transport layer based on a shared ring buffer: **vring**
- an **rpmsg device** is a *communication channel* with a remote processor: **channel**



Firmware management by **remoteproc**

- **remoteproc** being responsible of loading the firmware it expects a specific structure
- Currently ELF32 and ELF64
- **remoteproc** parses the firmware and loads the segments to memory according to the specified device address (might be a physical address)
- Beside the standard ELF segments **remoteproc** also manages a special section: the **resource table**
- This section holds a C structure which contains:
  - system resources required by the remote processor
  - resources exposed by the remote processor: trace buffers and supported virtio devices

Adapting Linux kernel and OpenAMP for Cortex A9

Design choice to support AMP on Cortex A9

- Shared memory
  - A part of the main memory (DDR) used as shared memory
  - No use of an SRAM: less dependent on a given SoC
- Signalization
  - Software interrupt used to signal the presence of a message
  - Use **SGI** (Software Generated Interrupt) provided by the **GIC** (the interrupt controller provided by ARM and used with Cortex A9)

Adding support to Open AMP: libraries

- For **lib_openamp**, nothing to do expect adding a platform cmake file to describe the toolchain option to use.
- For **libmetal**, a few more things, to add the support of the SoC (here i.MX6)
  - Platform cmake file for the toolchain as done for the openAMP library
  - Adding `sys.c` and `sys.h` files to mainly expose interrupt support

Using **OpenAMP** in RTOS

- For the RTOS project, **OpenAMP** is used to add the remote proc and rpmsg support:
  - Create the `imx6_a9_rproc.c` file defining the platform specific remoteproc implementation. It provides the `remoteproc_ops` that defines notification operation and remote processor management operations. Actually not specific to i.MX6 and only to Cortex A9
  - Create the `platform_info.[ch]` files responsible of initializing the libmetal, creating a remote proc instance and the associated rpmsg devices
  - Create the `rsc_table.[ch]` files that populate the resource table
  - Add parameter support when registering an interrupt: needed for libmetal to use a generic **ISR** (interrupt service Routine)

Adding support to Linux: remoteproc driver

- A **remoteproc** driver has to be added
- In probe function need to setup the GIC to use the **SGI** to receive the signal from the firmware that a specific virtqueue has pending messages available
- Also implement a `parse_fw` operation to prepare the shared memory
- Finally need to add `start` and `stop` operations
  - `start` moves the **Cortex A9** core outside of the SMP group to run the firmware (here **FreeRTOS**)
  - `stop` moves back the **Cortex A9** core to SMP from AMP

Adding support to Linux: SoC specific part

- While most of the support only rely on **Cortex A9**, some parts are SoC dependents
- `start` and `stop` operations uses generics functions to migrate core from or to the SMP group
- However the way to pass the entry point fo the firmware to the core is really SoC specific
- As well as the way to actually power up the core
- Those functions already exist but have to be made accessible outside of the `arch/arm/` directory

Pros, Cons and alternatives

Drawbacks of this solution

- Not a real **segregation** between Linux and the RTOS
- Nothing prevent the firmware to **access Linux memory** or to manage devices used by Linux kernel drivers
- The safety and security depend on the **design** of the solution
  - System Devices Trees might be used to describe each partition

Strength of this solution

- Based on established and open projects: Linux kernel and **OpenAMP**
- Few modifications to do on the mainline projects and most of them should be merged
- Could be easily extended to any SoC using Cortex A9
- **OpenAMP** allows to continue using an existing coding base in an RTOS and in the same time using a standard way to interface with Linux.

Alternative to **OpenAMP** to use a main core

- **Jailhouse** is a partitioning Hypervisor based on Linux
  - is able to run bare-metal applications or adapted RTOS besides Linux
  - manages to do a **real segregation** between Linux and the RTOS
  - relies on **virtualization extensions** that are not present on Cortex A9 but on more **recent ARMv7** (Cortex A7, Cortex A15, ...) and also on **ARMv8** (arm64)
- Full task isolation
  - Isolate a task from the rest of the OS: **won't be interrupted** at all by the kernel
  - With userspace driver using UIO it is also possible to access the hardware without any system call
  - However the solution is still **not merged**

Questions? Suggestions? Comments?

Gregory CLEMENT
gregory.clement@bootlin.com

http://bootlin.com/pub/conferences/2022/lee/clement-AMP-on-Cortex-A9