Upstreaming ARM64 SoC's easier than before

Manivannan Sadhasivam
Kernel Engineer, Linaro-96Boards

Linaro connect
San Diego 2019

Contents

- Overview
- Who am I?
- Conventional question - Why upstreaming?
- ARM vs ARM64 in Linux kernel (Simplistic)
- ARM64 SoC Pull Request Tips
- ARM64 SoC Upstreaming Checklist
  - Basic SoC and board support
  - Add core SoC infrastructure
  - Bring up Non Volatile memory
  - Bring up Networking
  - Bring up Display and Audio
  - Add Nice-to-have features

Overview

- What are we going to discuss?
  - How should upstream ARM64 SoC's in Linux kernel
- What is not applicable?
  - This talk is not about how to do generic kernel upstreaming
  - For this, please refer: **Upstreaming 101** track
- Target audience
  - SoC/Board vendors
  - Developers familiar with kernel upstreaming process
- Outcome
  - Know how to do ARM64 SoC upstreaming seamlessly!

Who am I?

- Manivannan Sadhasivam - manivannan.sadhasivam@linaro.org
  - Kernel Engineer at **Linaro-96Boards**
    - Taking care of the SW ecosystem of **96Boards**
    - Frequently answer the question, **"What is 96Boards?"**
    - Encouraging vendors to participate in **Open Source** activities
    - Helping the **community**!
  - Open Source contributor - OpenHub
    - Contributing to Linux Kernel
      - Maintaining **Bitmain**, **RDA Micro** SoCs
      - Co-Maintaining **Actions Semi** SoCs
      - Random contributions all over the kernel
    - Contributing to U-Boot
      - Maintaining **Actions Semi** SoCs
      - Co-Maintaining **HiSilicon** SoCs
      - Maintaining few Dev boards (Mostly 96Boards)
    - Contributions to Zephyr
      - Maintaining few SoCs, boards, drivers
      - Maintaining **LED** subsystem
      - Proposed **LoRa** support (under review)

Conventional question - Why SoC upstreaming?

- Why a vendor should upstream?
  - Maintaining downstream SoC port is **hard**
  - Show up your SoC in official kernel released by **Linus Torvalds**
  - Spare developers worldwide to work on your SoC
  - Solve the problem with **community**
  - Allow customers to work on **cutting edge** features of kernel
  - Make use of **LTS** kernel
  - Build up **Open Source** reputation

- Why a random developer should upstream?
  - Gain knowledge
  - Get yourself listed in **MAINTAINERS** file
  - Send pull requests to **arm-soc** maintainers
  - Gain reputation and become **famous**!

ARM vs ARM64 in Linux Kernel (Simplistic)

- **This is not a performance comparison**
- ARM
  - No vendor based DTS hierarchy (for now)
    - All SoC DTS are listed under: `arch/arm/boot/dts/`
  - Need SoC specific code to start secondary processors
    - Don't use "**pen_release**" stuff
  - Missing standardization
    - It comes from the architecture
  - Patches need to be prefixed with `ARM:`
  - Mailing list for Pull Req and Patches to arm-soc maintainers: soc@kernel.org
    - **Not to be used for other purposes**
    - **For general discussions/patches, use:** linux-arm-kernel@lists.infradead.org

ARM vs ARM64 in Linux Kernel Contd...

- ARM64
  - Vendor based DTS hierarchy exists
    - For instance, `arch/arm64/boot/dts/actions/`
  - No need of SoC specific code to start Secondary processors
    - Generic code exists which make use of PSCI
  - Much more standardized
    - It comes from the architecture
  - Patches need to be prefixed with `arm64:`
  - Mailing list for Pull Req and Patches to arm-soc maintainers: soc@kernel.org
    - **Not to be used for other purposes**
    - **For general discussions/patches, use:** linux-arm-kernel@lists.infradead.org

ARM64 SoC Pull Request Tips

- **Only applicable to new SoC families**
- Rule of thumb: **Start small and build it big**
- Push the code to open Git environment
    - Github repository is fine
- Base your changes on earlier RCs
    - Preferably `-rc2`
- Make sure they build and collect all Acks/Reviews
- Devicetree binding patches should come first
- Create and push the signed tag to Git tree
- Briefly explain the effects of the Pull Request in signed tag
    - Do not copy paste the commit's subject
    - Tag description should justify why your code needs to be pulled in, high level overview of what is going on with the SoCs, what is still missing etc...
- Submit the Pull Requests soon after merge window
    - Make sure the drivers (if any) are picked up by subsystem maintainers

ARM64 SoC Upstreaming Checklist

- Below is the checklist based on my experience (**order matters**)
  - Basic SoC and board port
  - Add core SoC infrastructure
  - Bring up Non Volatile memory
  - Bring up Networking
  - Bring up Display and Audio
  - Add Nice-to-have features

Basic SoC and board support

- SoC should boot into **initramfs** with all CPUs
- Most of the time (depending on the SoC design), a single DTS can do the job
- Reuse the existing drivers present in mainline
- Following drivers are needed:
    - Serial
        - Preferably with **earlycon** to ease debugging
    - IRQ
        - Check if the SoC has GIC as the first level* interrupt controller routing interrupt to the SoC
    - Timer
        - Check if the SoC has per core architectured timer
    - PSCI
        - EL3 firmware should support PSCI interface
- Add a development board based on the SoC and enable exposed serial ports
- Add the device tree binding for the SoC
    - Preferably in JSON schema
- https://lkml.org/lkml/2019/1/25/909

Add Core SoC Infrastructure

- SoC should boot into **initramfs** with **clk** and **gpio/pinctrl** support
- This is the critical and often tough task
- Following drivers are needed:
    - Common clk driver - **drivers/clk**
        - Try to use clk_hw* APIs instead of the clk_* APIs
        - Don't use strings for the parent clocks, use clk_hw instead
        - https://lkml.org/lkml/2019/2/26/811
    - Reset driver
        - Check if `reset-simple` driver can be used
        - If the reset functionality is provided by clock IP, then integrate it with common clk driver
    - Gpio driver - **drivers/gpio**
        - If a single IP exposes both Pinctrl and GPIO, use a single Pinctrl driver
        - Use hierarchical IRQ implementation if applicable
        - Include `linux/gpio/driver.h` instead of `linux/gpio.h`
        - Use libgpiod library to test - **Do not use SYSFS**
    - Pinctrl driver - **drivers/pinctrl**
        - Use `pinctrl-single` driver if applicable

Bring up Non-Volatile Memory

- SoC should boot a **distro** from any non-volatile memory attached to the SoC
  - Someone may use NFS in this stage but I don't prefer!
- Enable the SoC architecture in ARM64 defconfig if applicable
  - The SoC family can only be enabled in defconfig if there are enough drivers for it
  - But it is good to enable once it can boot a distro
- Following drivers are needed:
  - DMA Engine driver
    - Start with memcpy and add slave support
  - I2C/SPI/SPMI driver
    - This depends on the PMIC used
  - Regulators/PMIC
    - Most of the regulators offer multi functionality, so use MFD glue
  - Non Volatile controller driver
    - This can be MMC/UFS/NAND/NOR/USB/PCI-E

Bring up Networking

- Establish basic network access to the board
- User should be able to update the distro
- One of the following drivers are needed:
    - Ethernet
        - MAC/MDIO driver
        - PCI-E driver
            - If external Ethernet controller is used
    - WiFi
        - MMC driver
            - If on-board WiFi is used
        - USB
        - PCI-E

Bring up Display and Audio

- SoC should drive any on-board display interface
- SoC should output audio through any interface
- Following drivers are needed:
    - Display
        - DSI driver
        - I2C/SPI driver
    - Audio
        - I2S driver
        - I2C/SPI driver
        - ALSA machine driver if required

Add Nice-to-have features

- Add the rest of the key features of the SoC
- Following functionalities could be added:
  - GPU support
    - Preferably OpenGPU :-)
  - Video Encoder/Decoder support
  - Camera support
    - MIPI CSI2/Parallel
  - Anything left in previous steps!

Questions?

Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org

Linaro connect
San Diego 2019