Why Split Boot?

Zynq MPSoC

Processing System

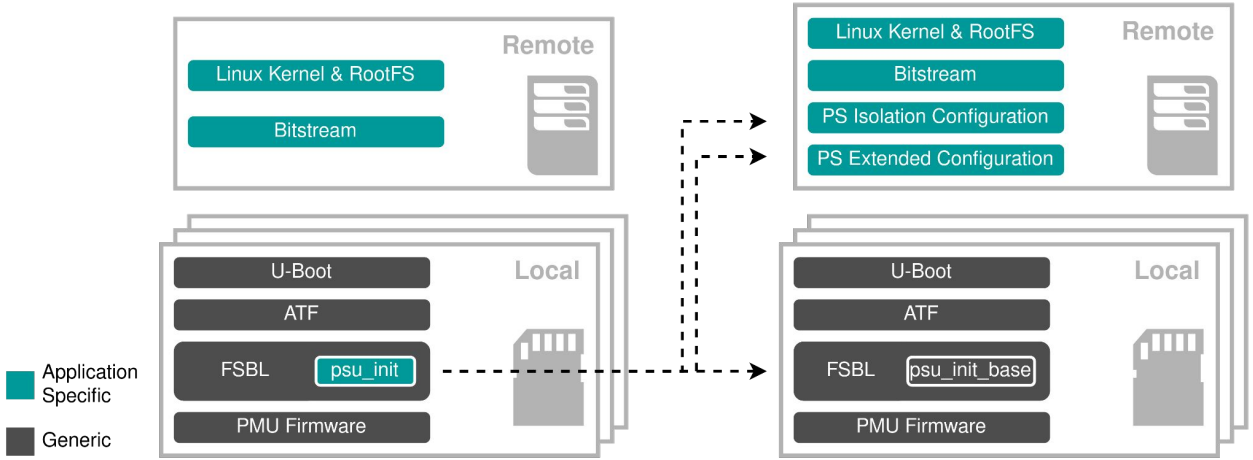
- 4x ARM Cortex-A53
- 2x ARM Cortex-R5

Programmable Logic

- FPGA

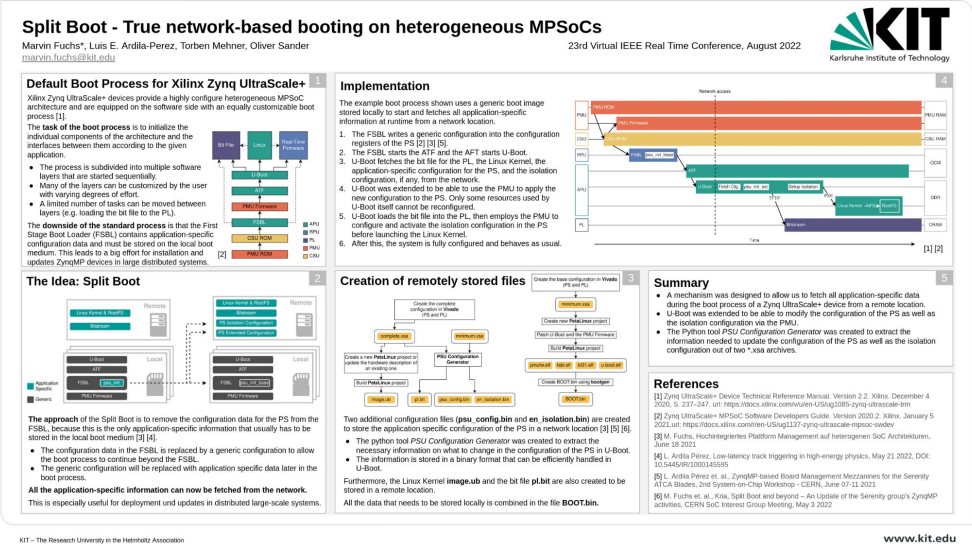
Challenge:

- Big effort to conventionally **deploy** and **update** a large, distributed system with many Xilinx ZynqMP devices



What is on our Poster

- Summary of the default boot process
- The idea behind Split Boot
- Implementation
- Integration in the development workflow



Split Boot - True network-based booting on heterogeneous MPSoCs

Marvin Fuchs*, Luis E. Ardila-Perez, Torben Mehner, Oliver Sander
marvin.fuchs@kit.edu

23rd Virtual IEEE Real Time Conference, August 2022

Karlsruhe Institute of Technology

Default Boot Process for Xilinx Zynq UltraScale+

Xilinx Zynq UltraScale+ devices provide a highly configure heterogeneous MPSoC architecture and are equipped on the software side with an equally customizable boot process [1].

The **task of the boot process** is to initialize the individual components of the architecture and the interfaces between them according to the given application.

- The process is subdivided into multiple software layers that are started sequentially.
- Many of the layers can be customized by the user with varying degrees of effort.
- A limited number of tasks can be moved between layers (e.g. loading the bit file to the PL).

The **downside of the standard process** is that the First Stage Boot Loader (FSBL) contains application-specific configuration data and must be stored on the local boot medium. This leads to a big effort for installation and updates ZynqMP devices in large distributed systems. [2]

The Idea: Split Boot

The approach of the Split Boot is to remove the configuration data for the PS from the FSBL, because this is the only application-specific information that usually has to be stored in the local boot medium [3] [4].

- The configuration data in the FSBL is replaced by a generic configuration to allow the boot process to continue beyond the FSBL.
- The generic configuration will be replaced with application specific data later in the boot process.

All the application-specific information can now be fetched from the network.

This is especially useful for deployment und updates in distributed large-scale systems.

Implementation

The example boot process shown uses a generic boot image stored locally to start and fetches all application-specific information at runtime from a network location.

1. The FSBL writes a generic configuration into the configuration registers of the PS [2] [3] [5].
2. The FSBL starts the ATF and the AFT starts U-Boot.
3. U-Boot fetches the bit file for the PL, the Linux Kernel, the application-specific configuration for the PS, and the isolation configuration, if any, from the network.
4. U-Boot was extended to be able to use the PMU to apply the new configuration to the PS. Only some resources used by U-Boot itself cannot be reconfigured.
5. U-Boot loads the bit file into the PL, then employs the PMU to configure and activate the isolation configuration in the PS before launching the Linux Kernel.
6. After this, the system is fully configured and behaves as usual.

[1] [2]

Creation of remotely stored files

Two additional configuration files (**psu_config.bin** and **en_isolation.bin**) are created to store the application specific configuration of the PS in a network location [3] [5] [6].

- The python tool *PSU Configuration Generator* was created to extract the necessary information on what to change in the configuration of the PS in U-Boot.
- The information is stored in a binary format that can be efficiently handled in U-Boot.

Furthermore, the Linux Kernel **image.ub** and the bit file **pl.bit** are also created to be stored in a remote location.

All the data that needs to be stored locally is combined in the file **BOOT.bin.**

Summary

- A mechanism was designed to allow us to fetch all application-specific data during the boot process of a Zynq UltraScale+ device from a remote location.
- U-Boot was extended to be able to modify the configuration of the PS as well as the isolation configuration via the PMU.
- The Python tool *PSU Configuration Generator* was created to extract the information needed to update the configuration of the PS as well as the isolation configuration out of two *.xsa archives.

References

[1] Zynq UltraScale+ Device Technical Reference Manual. Version 2.2. Xilinx. December 4 2020. S. 237–247. url: https://docs.xilinx.com/v/u/en-US/ug1085-zynq-ultrascale-trm

[2] Zynq UltraScale+ MPSoC Software Developers Guide. Version 2020.2. Xilinx. January 5 2021.url: https://docs.xilinx.com/r/en-US/ug1137-zynq-ultrascale-mpsoc-swdev

[3] M. Fuchs, Hochintegriertes Plattform Management auf heterogenen SoC Architekturen, June 18 2021

[4] L. Ardila Pérez, Low-latency track triggering in high-energy physics, May 21 2022, DOI: 10.5445/IR/1000145595

[5] L. Ardila Pérez et. al., ZynqMP-based Board Management Mezzanines for the Serenity ATCA Blades, 2nd System-on-Chip Workshop - CERN, June 07-11 2021

[6] M. Fuchs et. al., Kria, Split Boot and beyond – An Update of the Serenity group's ZynqMP activities, CERN SoC Interest Group Meeting, May 3 2022

KIT – The Research University in the Helmholtz Association

www.kit.edu

I am looking forward to answer your Questions in the poster session!