SERENITY

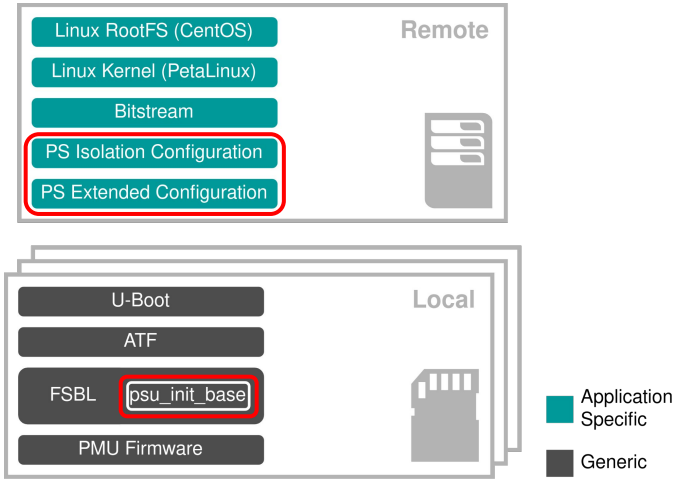
Kria, Split Boot and beyond – An update of the Serenity group ZynqMP activities

Marvin Fuchs (KIT)
on behalf of the Serenity Collaboration

Why Split Boot?

Challenge: Big effort to conventionally **deploy** and **update** a large, distributed system with many ZynqMP because of application specific boot code (psu_init in FSBL)

Idea: Move project specific parts (e.g. PS/PL interface) to remote, Base configuration comparable to BIOS / UEFI on desktop PC (fundamental boot capabilities to enable network boot)

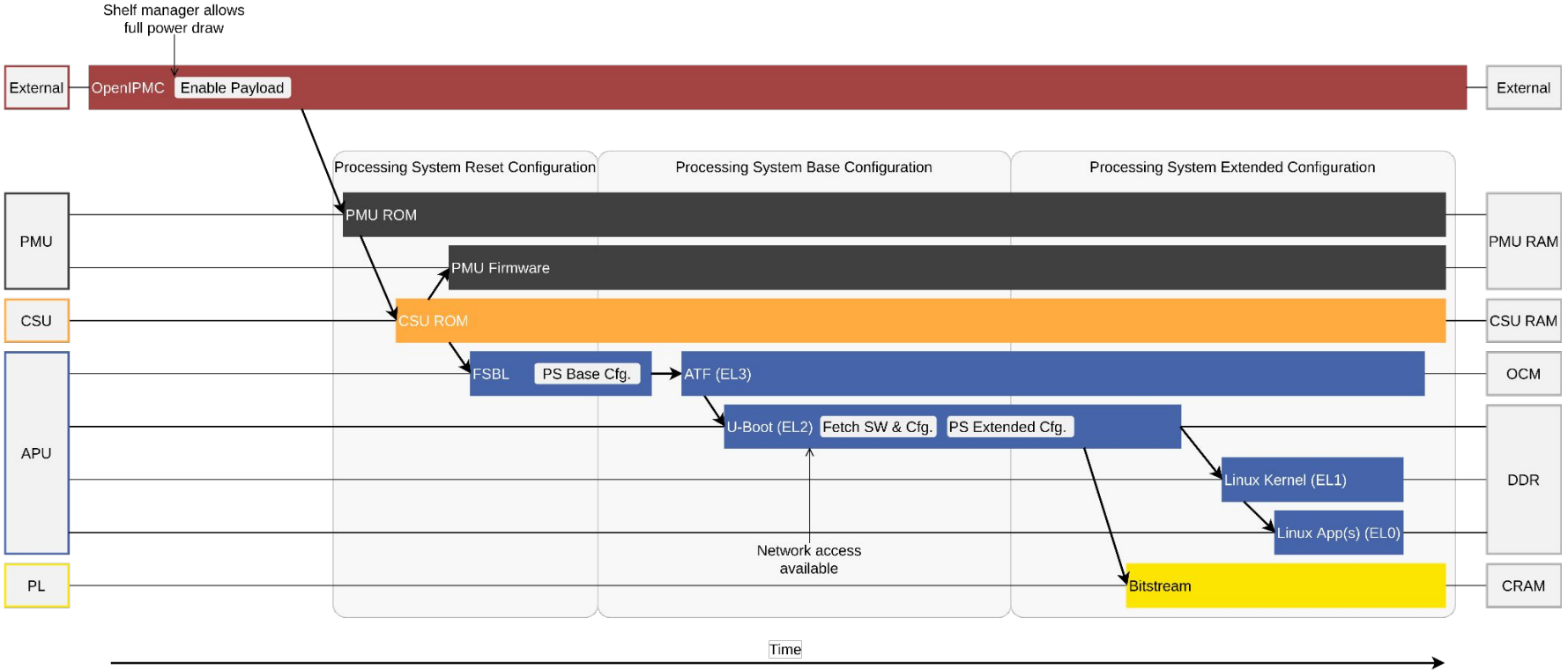


Advantages:

- Static base configuration on the boards stored locally
- Easy and quick to update application specific files and configurations, potentially for many boards, in a single location
- Easy and quick to replace broken boards

https://indico.cern.ch/event/996093/contributions/4376631/attachments/2261102/3837884/2021-06-09_SoC_Workshop.pdf

ZynqMP Split Boot on ATCA Blades



Initialization Files psu_init.c and psu_init.h

Encapsulate the configuration of the Processing System (PS) in the FSBL

psu_init.h:

- Independent of the configuration in the PS Configuration Wizard (PCW) in Vivado

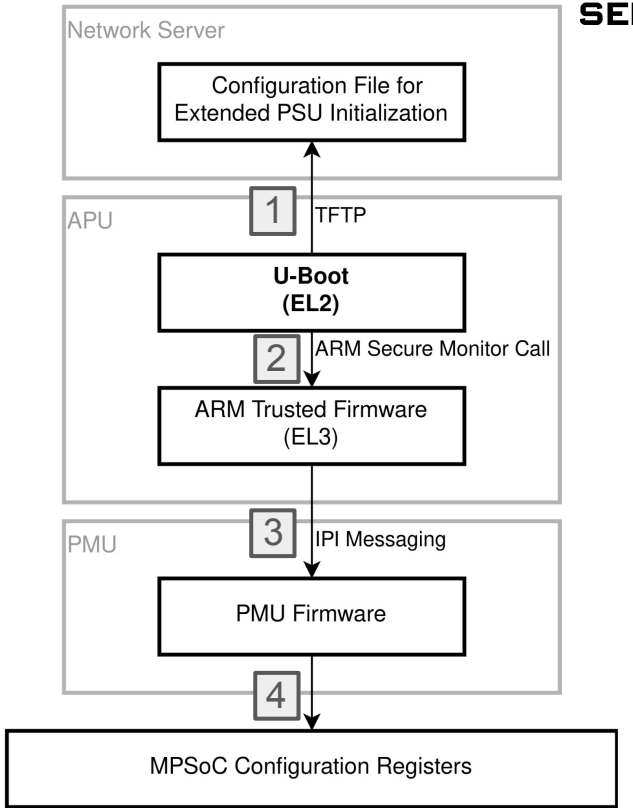
psu_init.c:

- The function **psu_init** contains the configuration of the PS
- The function **psu_protection** enables the isolation
- Most functions are just a long list of read-modify-writes, polls and some delays
  - There content depends on the configuration in the PCW in Vivado
- Some functions are more complex
  - Most of these functions are related to SerDes
  - The content does not depend on the configuration in the PCW in Vivado
  - Are adapted to the configuration via transfer parameters

Integration into U-Boot

1. U-Boot fetches configuration data from the network
2. U-Boot initiates a register access by a request to the ARM Trusted Firmware (ATF)
3. The ATF forwards the request to the PMU Firmware
4. The PMU Firmware accesses the register as requested
5. Continue with 2. for the next register access

Access through PMU necessary to circumvent restrictions in the permissions of U-Boot (ARM Exception Level)



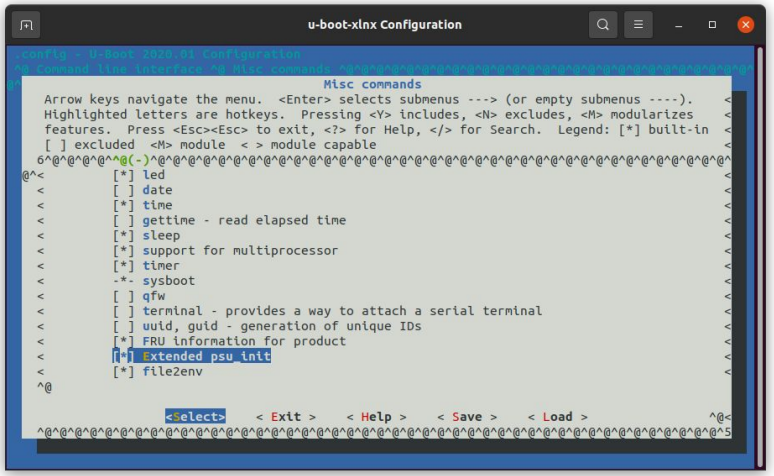
Split Boot Extension for U-Boot

Extension encapsulated in U-Boot command **psuinite**

- Integrated in the U-Boot Kconfig system
- Directly build into U-Boot

Implementation as U-Boot standalone application also tested

- Standalone applications can be fetched from the network
- For security reasons the **go** command in Xilinx U-Boot switches to exception level EL1 and rejects parameters



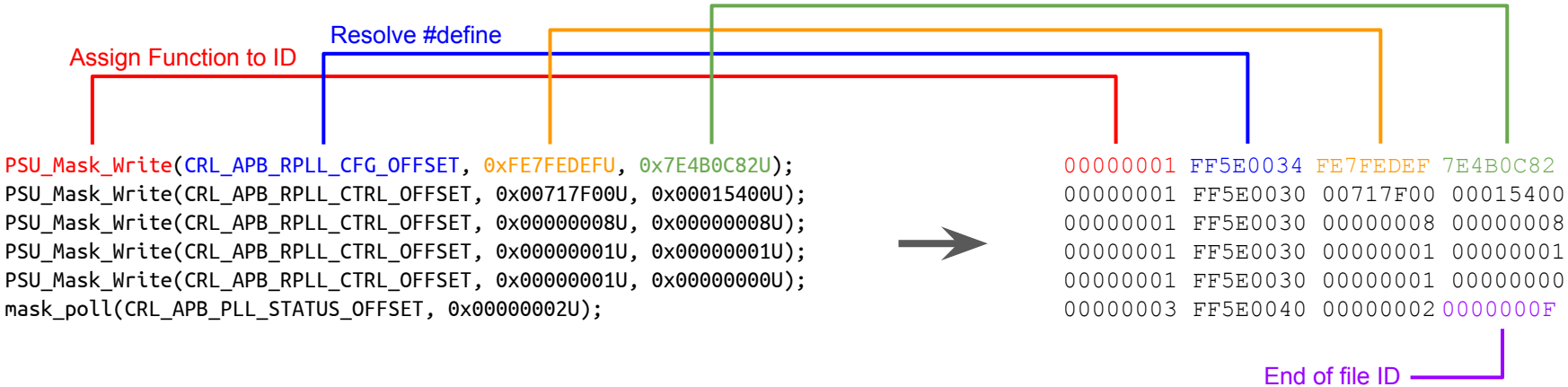

```
ZynqMP> help psuinite
psuinite - Execute the extended psu_init

Usage:
psuinite [-phv] -a <addr>

ZynqMP>
```

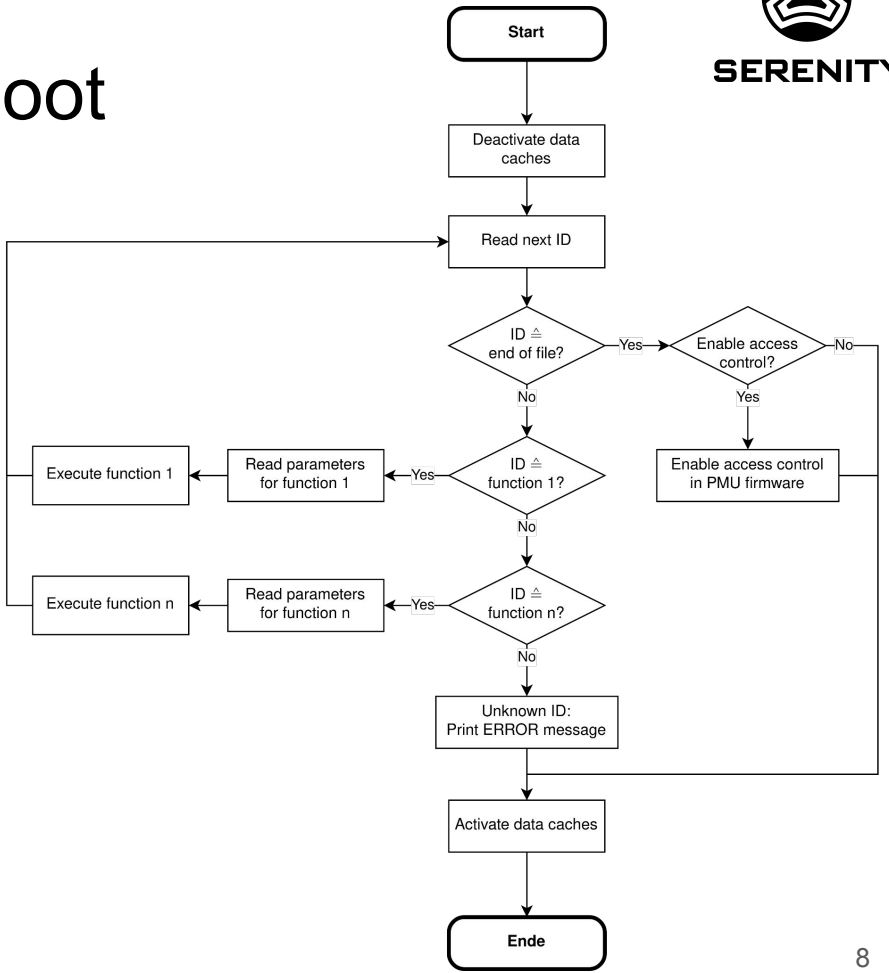
Split Boot Configuration Files

- Contain all the configuration-dependent information from psu_init.c/.h
- Represent a list of function calls with the corresponding parameters
- Binary format
  - Not human readable
  - Easy to create
  - Easy to read and use in U-Boot



Split Boot Extension for U-Boot

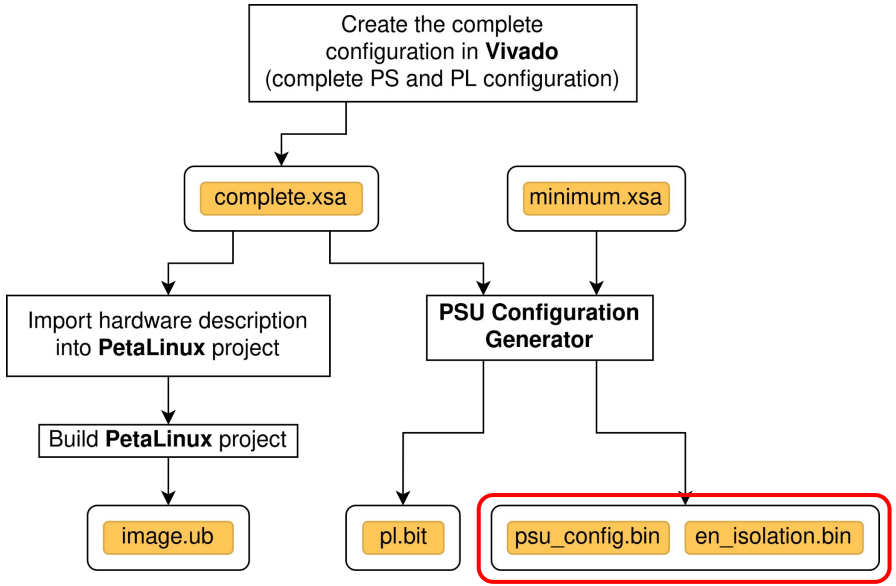
- Step through the configuration file and execute the function calls
- Deactivate data caches to enable the PMU to access the configuration registers
- Stop at the end-of-file ID
- After the configuration of the PS is completed, enable access control in the PMU firmware
    - Room for improvement / to improve security



Creation of Remotely Stored Files

1. A conventional boot firmware consisting of the FSBL, the ATF, a patched U-Boot, and a patched PMU Firmware is created
2. The functional configuration for the PS and the PL are created, as well as the Linux OS

- In Vivado: **complete** configuration of PS and PL
- **New:** PSU Configuration Generator
  - Creates binary PS-Configuration files
  - Extracts bitstream of the complete design
- Output files stored on TFTP Server



There is Room for Improvement

SERENITY

- PMUFW still expects **list of active components** from FSBL

  -> To be added to Split Boot
- Find a way to **change the configuration** of the PS without synthesis in Vivado
- Create/Move to a **command line based build framework**

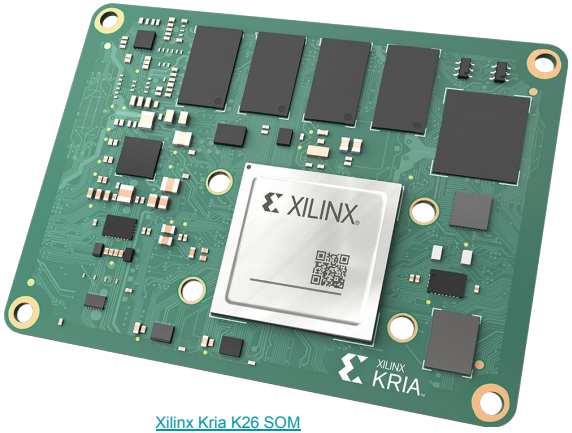
  to avoid the GUIs of Vivado and Vitis
- Closeness to **PC PXE boot**: test regular CentOS 8 installation with Split Boot on a ZynqMP

A new SoM: Xilinx Kria K26

Why Kria K26?

Multiple reasons:

- Availability
- Price
- Directly supported by Xilinx



Xilinx Kria K26 SOM

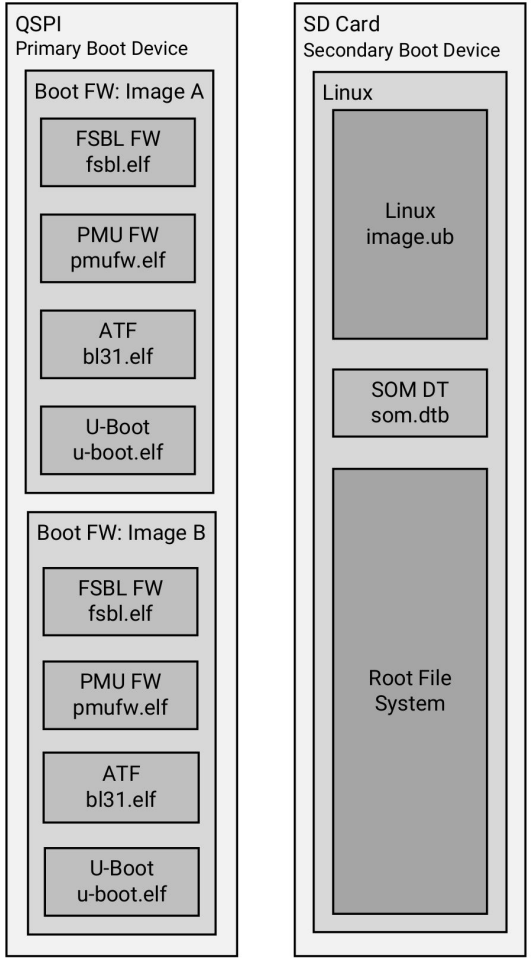
Downside:

- Schematics not publicly available

The decision was made to use the Kria on the **Serenity** boards. Serenity is an ATCA development platform, developed to support the CMS phase 2 upgrade

Boot Mechanism of the KV260

- Two firmware images in the QSPI memory
  - Update from PetaLinux via the tool xmutil
  - Update via a Web server running in the PMU Firmware (Ethernet Recovery Tool)
- Firmware images are booted alternatingly
  - Backup image in case a faulty Firmware image was flashed
- The Web server might be an interesting starting point for further exploration



Booting the KV260 directly from the SD card / JTAG

It is possible to overwrite the hardwired boot mode switches in software

- Apparently not from U-Boot, [the way it is possible on the ZCU102 board](), because the Kria gets stuck in reset
- It is possible to use the xsct tool for this, e.g. via a tcl script

- Maybe the PMU could be an onboard solution to switch the boot mode in software

```
################################################################################
# This script will overwrite the hardwired boot mode on the KV260 and boot the Kria   #
# from eighter JTAG or the SD card.                                            #
#                                                                              #
# Usage:                                                                       #
# $ source /opt/Xilinx/Vivado/2020.2/settings64.sh                             #
# $ xsct kria_bootmode.tcl sd                                                  #
################################################################################

# Check for arguments
if { $argc != 1 } {
    puts "The kria_bootmode.tcl script requires one argument."
    puts "$ xsct kria_bootmode.tcl sd"
    exit 2
}

# Assign list elements to variables
lassign $argv bootmode

connect
targets -set -filter {name =~ "PSU"}
stop

if {$bootmode == "jtag"} {
    puts "Switch to JTAG bootmode"
    targets -set -nocase -filter {name =~ "PSU"}
    stop
    # update multiboot to ZERO
    mwr 0xffca0010 0x0
    # change boot mode to JTAG
    mwr 0xff5e0200 0x0100
    # reset
    rst -system
} elseif {$bootmode == "sd"} {
    puts "Switch to SD1 bootmode"
    # update multiboot to ZERO
    mwr 0xffca0010 0x0
    # change boot mode to SD
    mwr 0xff5e0200 0x5100
    # reset
    rst -system

    #Sometimes A53 may be held in reset catch .. so start it with "con"
    after 2000
    con
} else {
    puts "Unsupported bootmode $bootmode"
}

puts "Done!!"
```

https://twiki.cern.ch/twiki/bin/view/SystemOnChip/KriaPetaLinux

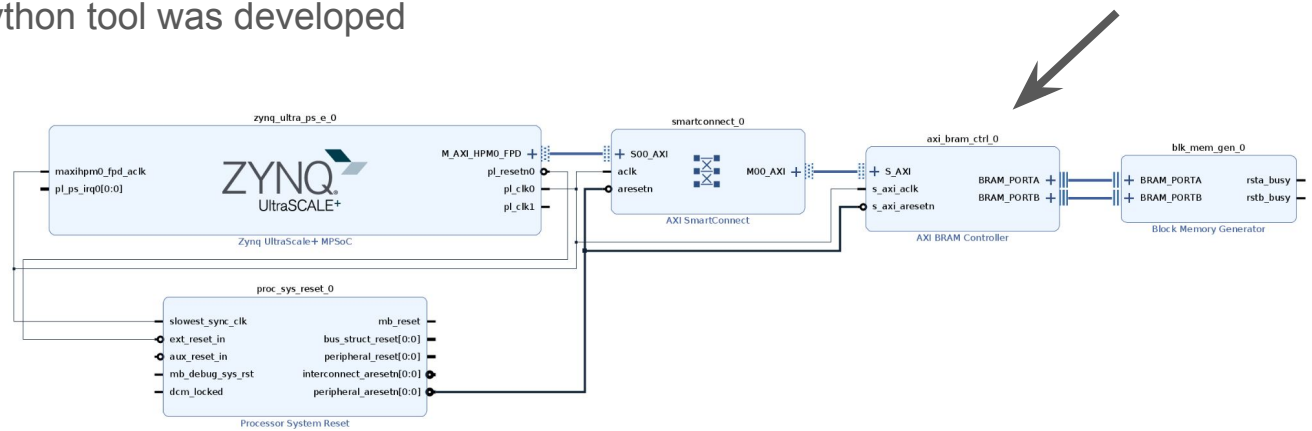
Enabling Ethernet in U-Boot on the KV260

- For some reason the Ethernet PHY initially reads a wrong address from the resistor configuration on power up
- Thus, **U-Boot** needs to address the PHY via the MDIO bus at address **0x05**
- One of the Linux Kernel drivers triggers a hard reset of the PHY. After that the PHY reads the correct address from the resistor configuration
- Thus, **Linux** needs to address the PHY via the MDIO bus at address **0x01**

- Just triggering a hard reset in U-Boot does not result in the correct address being read. It was not further investigated what needs to be done to read the correct address in U-Boot

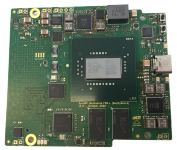
Porting Split Boot to the KV260

- Originally developed on a custom board using the **Trenz SoM** TE0803-04-4BE11-A and the Xilinx toolset in version **2020.2**
- Now successfully ported to the **Kria K26** with toolset version **2020.2.2**
  - The Split Boot was verified with a Block RAM in the PL and the I2C interfaces of the PS
  - Only limited possibilities to further verify Split Boot on the KV260 platform
- To automate the process of checking if the **Split-Boot-patches need to be updated**, a Python tool was developed



Conclusion


Trenz TE0803-04-4BE11


Serenity ZynqMP Mezzanines

Xilinx Kria K26 SOM

Split Boot

- Split Boot has now been tested on three different SoMs (Trenz, Serenity Zynq US+ Mezzanine, Kria K26)
- There are tools to support the integration in the development process
- There is a Python tool to check if the patches are compatible with a new toolset
- There are new things to be added (Provide more flexibility during development and in the second PS configuration stage, improved build system, …)

Kria K26 / KV260

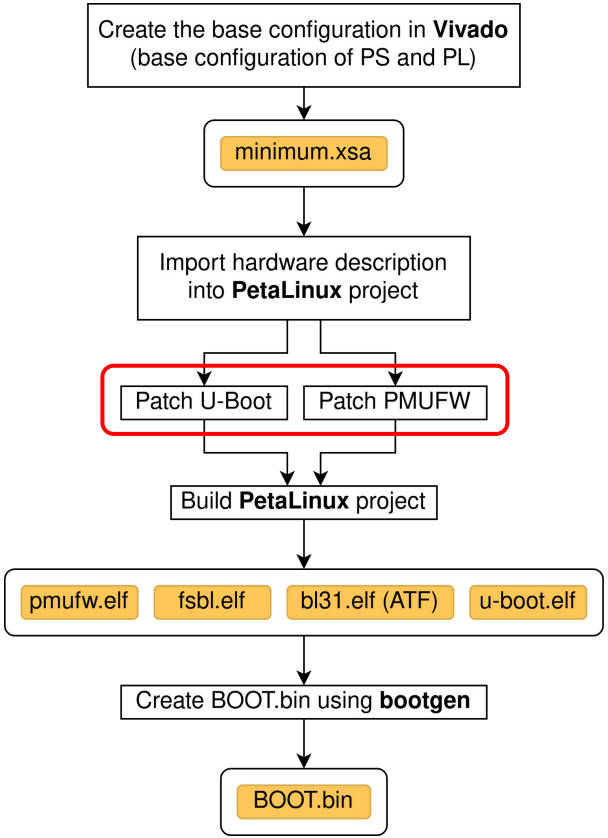
- We experienced multiple quirks when working with the KV260, but they are mostly related to the carrier card
- The Kria K26 can be used just like other ZynqMP SoMs
- There might be more interesting things to find in the software stack provided for the KV260

We are collecting all of our findings on the Kria SoM in the CERN TWiki
Feel free to take a look and share your findings there!

Creation of Locally Stored Files

- The PS needs a (base) configuration to boot to a stage where network access is available

- Standard workflow with Xilinx tools
- In Vivado: **base** configuration of PS (and PL)
- **New:** Patches add required modifications



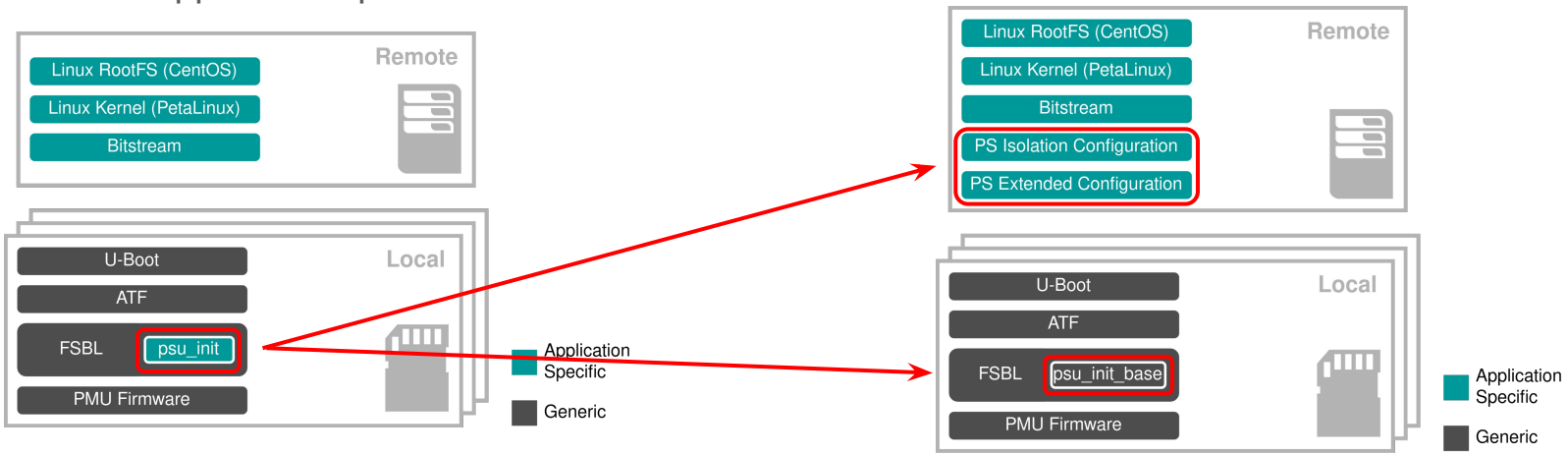
Recall: Why Split Boot?

Problem: Big effort to conventionally deploy and update a large, distributed system with many MPSoCs

- Requires flashing all the individual devices
- Mainly the configuration of the PS in the FSBL is application specific

Idea: Split the boot process in two parts

- Start the boot process on generic files, locally stored nearby the ZynqMP
- Fetch application specific Configuration and files from a network source



Recall: Why Split Boot?

SERENITY

Advantages:

- U-Boot offers PXE already -> Make the boot process as flexible as PXE on desktop PCs
- Base configuration comparable to BIOS / UEFI on desktop PCs

- Easy and quick to update application specific files and configurations, potentially for many boards, in a single location
- Ideally one single base configuration can be shared between different boards -> delivery of boards solely flashed with this base configuration
- During every boot process the boards fetch the latest data for their use case
- Eases tests during development

https://indico.cern.ch/event/996093/contributions/4376631/attachments/2261102/3837884/2021-06-09_SoC_Workshop.pdf

Thank you for your interest!