Deploying and testing firmware/software on Zynq MPSoC with GitLab CI/CD pipelines

Vasileios Amoiridis
EP-CMD | CMS DAQ group

Acknowledgements: P. Žejdl & M. Dobson

Workflow for the Zynq MPSoC UltraScale+

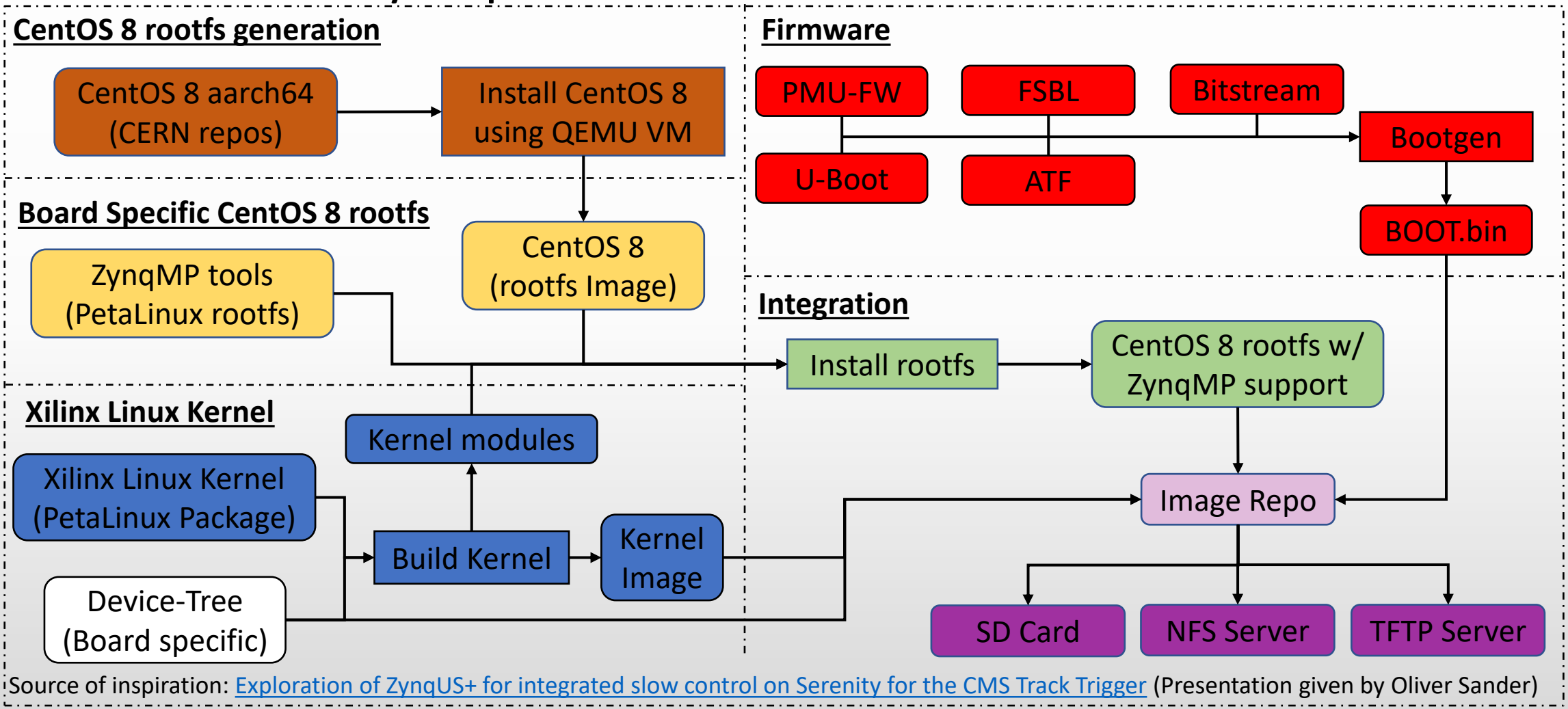
- Vivado creates the **Hardware Description File** which contains the configuration of the ARM processor and its peripherals. This file is used later by PetaLinux.
- PetaLinux creates the **Firmware Images** and the **PetaLinux rootfs**. The images are used to boot the Zynq and the PetaLinux rootfs is used to create the Board Specific ROOTFS.

We use Network Boot:

- The images need to be deployed to the **SD Card** and the **TFTP server** that are going to be used for booting. The ROOTFS needs to be deployed on the **NFS server**.
- When the board is booting its booting procedure is being monitored.
- After the board is booted, it is ready for automated testing!

In order to implement all the above steps, GitLab's CI/CD Pipeline framework is used!

Linux on Zynq MPSoC UltraScale+

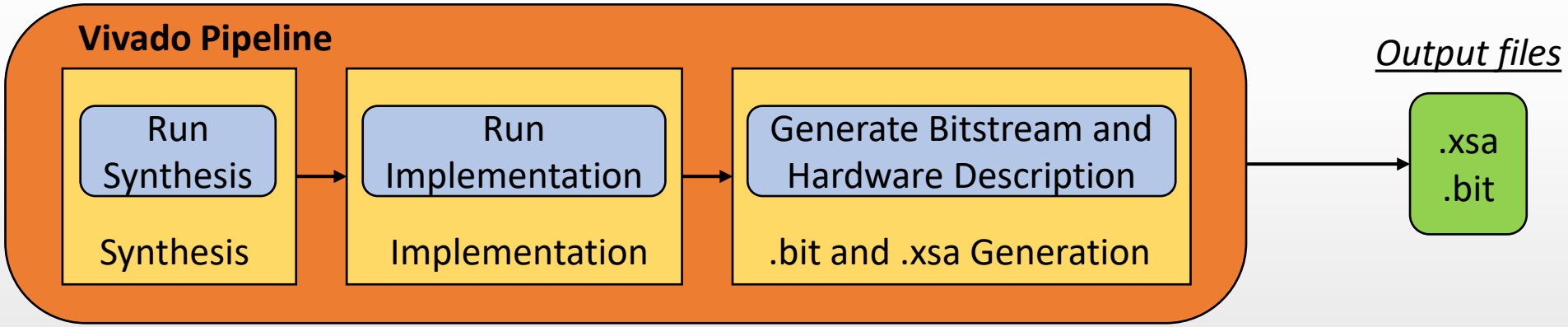


Source of inspiration: [Exploration of ZynqUS+ for integrated slow control on Serenity for the CMS Track Trigger] (Presentation given by Oliver Sander)

Vivado Pipeline

What is the role of the Vivado Pipeline?

- It is responsible to produce the Hardware Description File (**.xsa**) that is used by the PetaLinux Pipeline
- The **.xsa** file is the *Hardware Description* file. It has the configuration of the ARM processor and its peripherals
- The **.bit** file is the *Bitstream* file. It configures the Programmable Logic – FPGA. (Optional, i.e additional AXI bus)



- Every Vivado project is going under **Synthesis** and **Implementation** in a clean environment
- Runs inside a Docker Container with Vivado installed

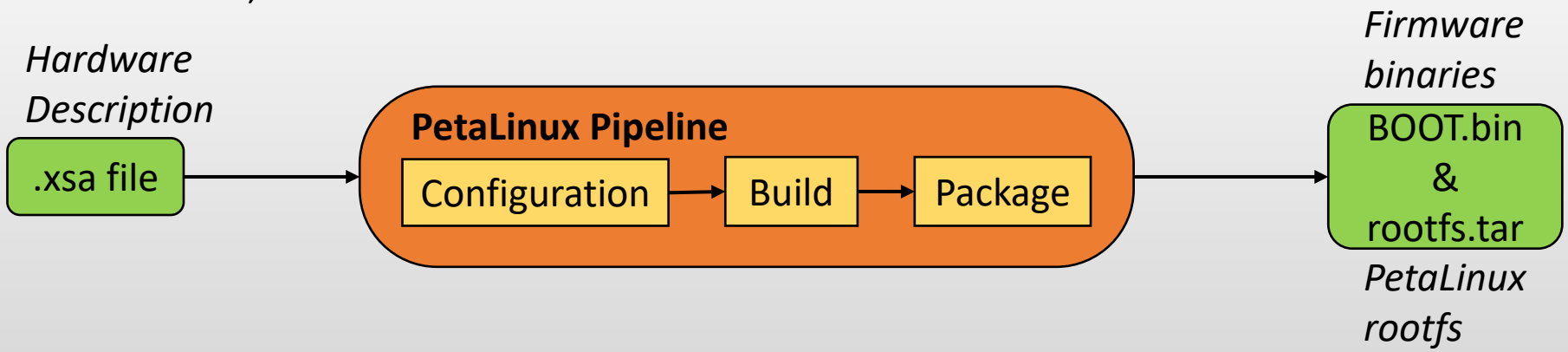
PetaLinux Pipeline

What is the role of the PetaLinux Pipeline?

- It uses the Hardware Description File produced by the Vivado Pipeline
- It creates the binaries that are going to be downloaded in the Zynq
- It creates the packages and the kernel modules that are needed for the root filesystem

What are the different stages in the PetaLinux Pipeline?

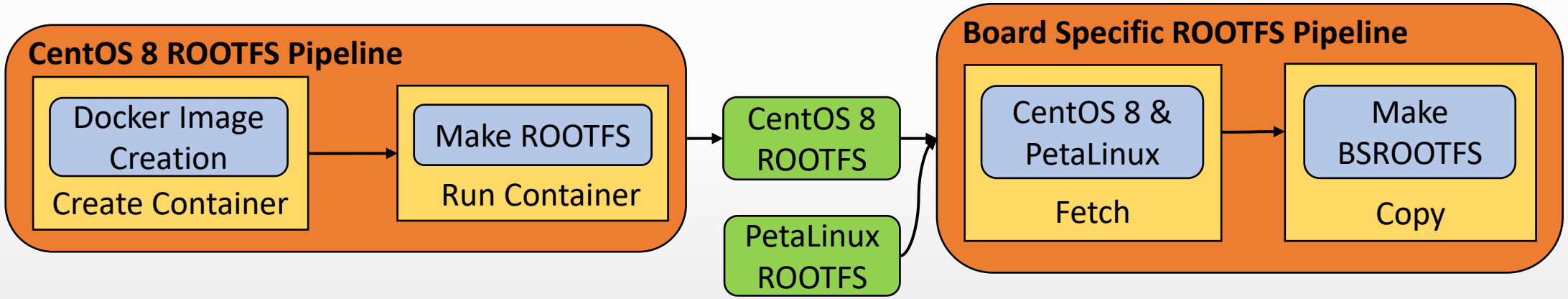
- There are 3 different stages: *Configuration*, *Build*, *Package*
- The configuration stage is responsible to generate configuration files for the build process of the firmware binaries, the bootloaders and the kernel



ROOTFS Generation

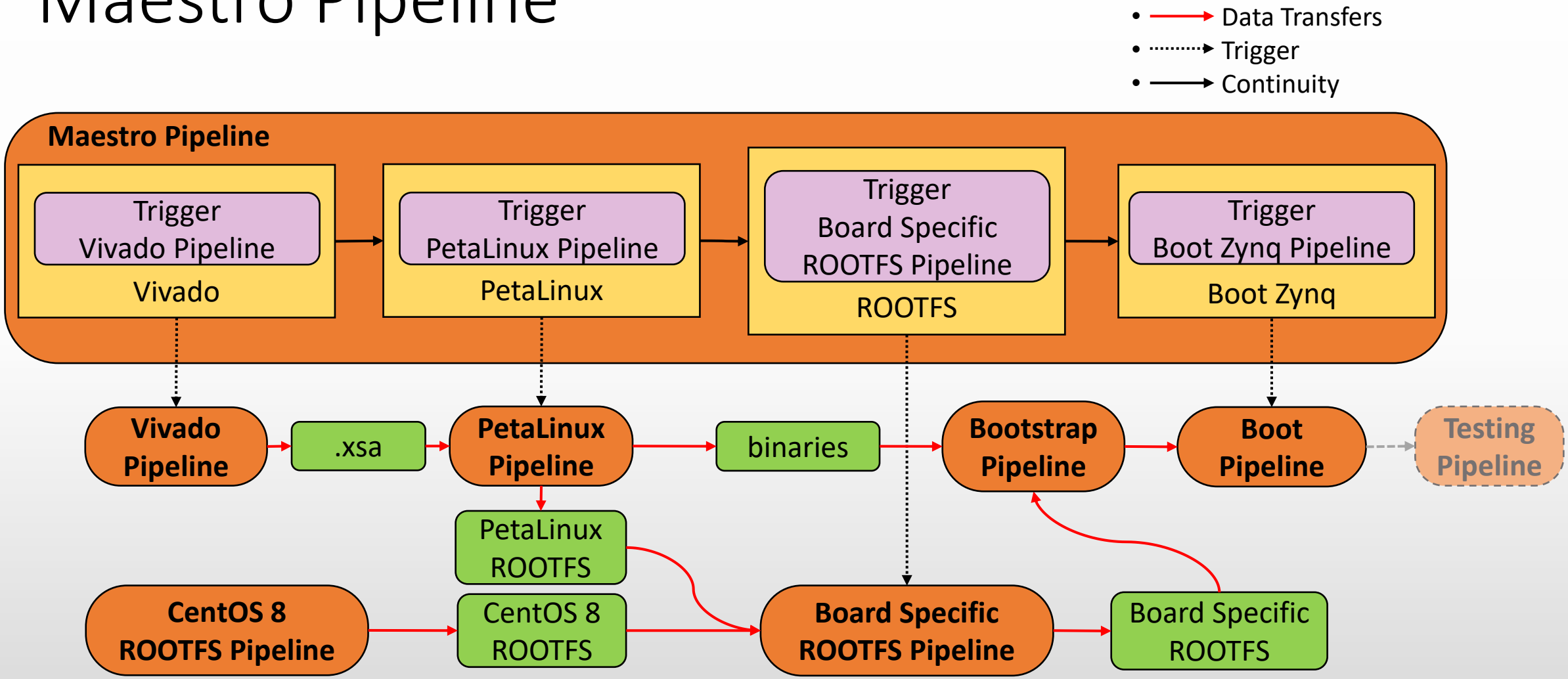
What is the role of the ROOTFS Pipeline?

- The ROOTFS Pipeline is responsible for creating the ROOTFS for the Zynq
- The Board Specific ROOTFS, needs the CentOS 8 ROOTFS and some extra PetaLinux tools and kernel modules

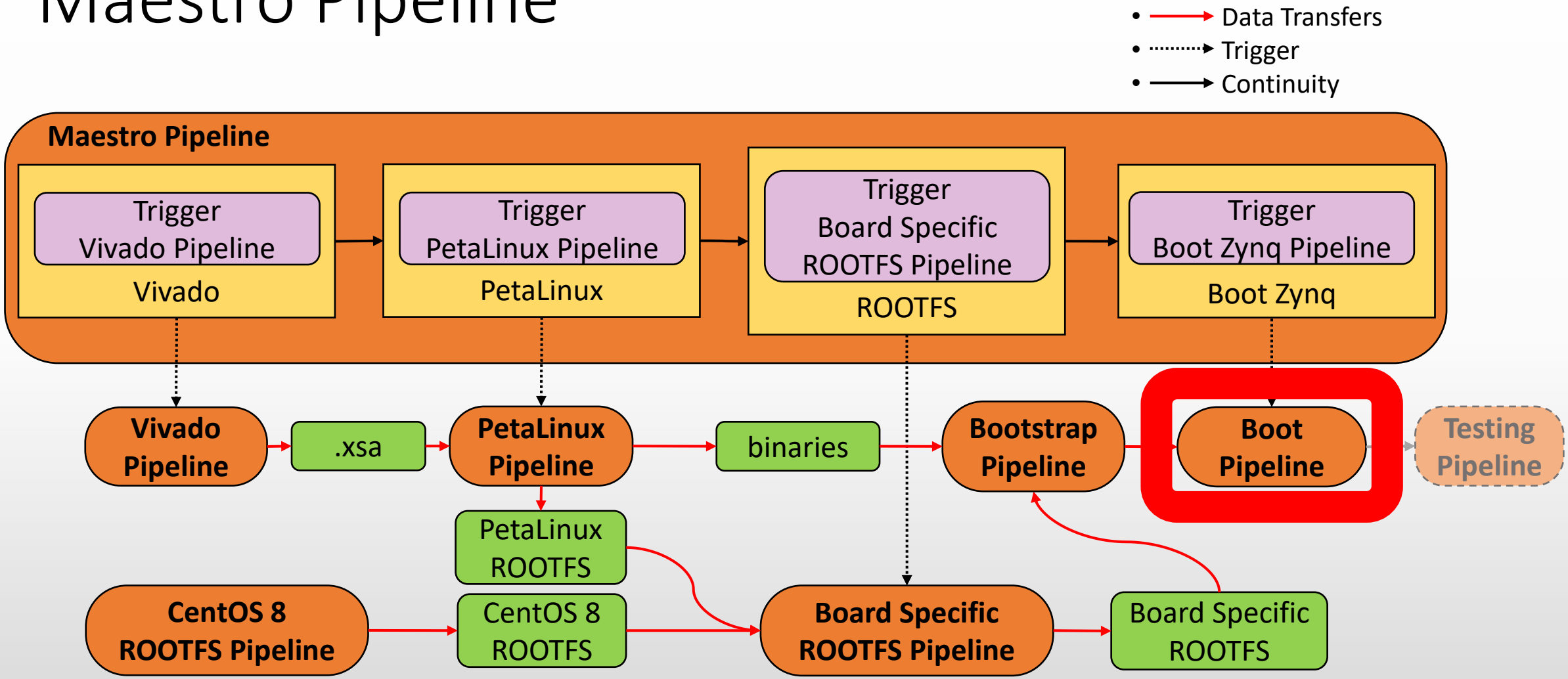


- QEMU and Dependencies are installed
- Docker Image is saved on *gitlab-registry.cern.ch*
- It is independent of the ROOTFS
- It is reused every time the ROOTFS is being built
- A python script runs **dnf** to install the ROOTFS

Maestro Pipeline



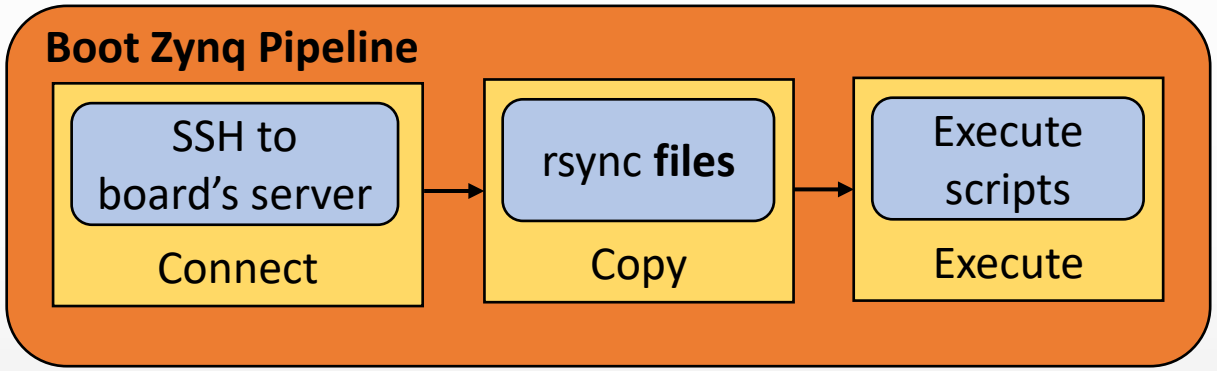
Maestro Pipeline



Boot Zynq Pipeline

What is the role of the Boot Zynq Pipeline?

- Boot the board and check that the board was booted correctly



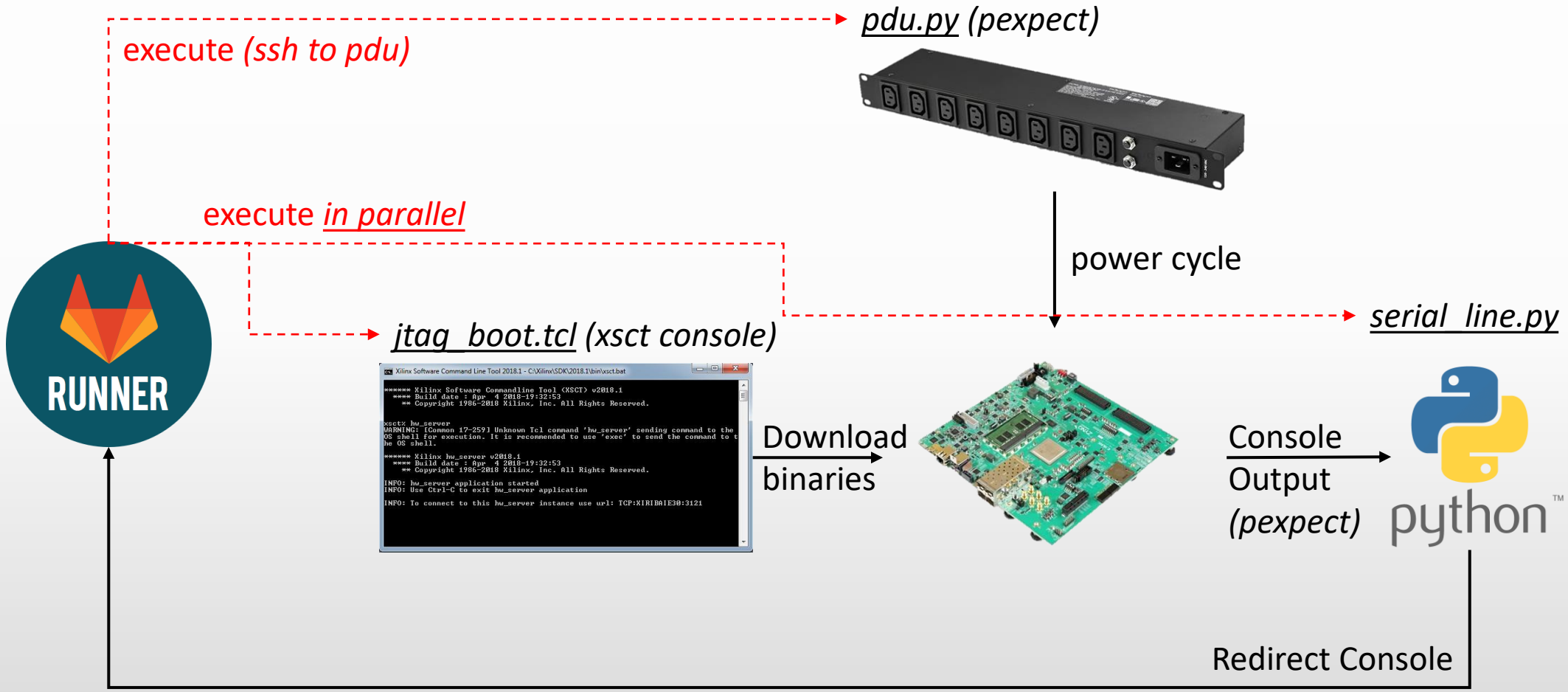
Files:

- **Dedicated images** (firmware, U-Boot) used for JTAG boot
- **Scripts** necessary to boot the board

How does the CI/CD interface with the board?

- The GitLab Runner initiates a **ssh** connection to the board's server that the board is connected to
- The appropriate files are copied from the repository to the board's server in order to interface with it

Booting Procedure - Overview



Booting Procedure - Zynq

What is the booting procedure and how do we implement the Network boot?

- The booting procedure is triggered from the Boot Zynq pipeline
- The repository of the Boot Zynq pipeline contains **JTAG images** to prepare the SD-Card

Stage 1: Prepare the SD-Card

- Boot through JTAG with the images and **dedicated U-Boot** for stage 1.
- During **dedicated U-Boot**:
  - Fetch the final images from the TFTP server that were built in the previous pipelines
  - Store these images in the SD-Card
  - Change boot-mode through the internal registers (JTAG mode -> SD-Card mode)
  - Reset the board (soft reset)

Stage 2: Default boot-mode

- The board starts booting from the SD-Card (firmware, U-Boot) and then continues with network boot:
  - Linux Kernel and Device-Tree are downloaded from the TFTP server
  - Mount root filesystem over NFS

DEMO – Scenario 1

Let’s create an error.

DEMO – Scenario 1 (Result)

Let’s create an error.

```
59  Exit from FSBL
60  [serial_line.py]: FSBL has exited successfully!
61  NOTICE:  ATF running on XCZU9EG/silicon v4/RTL5.1 at 0xfffea000
62  NOTICE:  BL31: Secure code at 0x60000000
63  NOTICE:  BL31: Non secure code at 0x8000000
64  NOTICE:  BL31: v2.0(release):xilinx-v2019.1-12-g713dace9
65  NOTICE:  BL31: Built : 08:29:34, Mar 18 2022
66  [serial_line.py]: ATF has exited successfully!
67  Traceback (most recent call last):
68    File "./scripts/serial_line_pexpect.py", line 142, in <module>
69      serialByte = serial.read_nonblocking(size=1,timeout=timeoutSerial)
70    File "/usr/local/lib/python3.6/site-packages/pexpect/fdpexpect.py", line 147, in read_nonblocking
71      raise TIMEOUT('Timeout exceeded.')
72  pexpect.exceptions.TIMEOUT: Timeout exceeded.
73  [runme.sh]: Exit code of serial_line_pexpect.py: 1
74  [runme.sh]: Exit code of jtag_boot.tcl: 0
75  [runme.sh]: An unexpected error occured in serial_line_pexpect.py. Exiting...
77  Cleaning up file based variables
79  ERROR: Job failed: exit status 1
```

DEMO – Scenario 2

Let’s fix the error.

DEMO – Scenario 2 (Result)

Let’s fix the error.

```
785  [  OK  ] Started Permit User Sessions.
786  [  OK  ] Started Login Service.
787  [  OK  ] Started Command Scheduler.
788  [  OK  ] Started Serial Getty on ttyPS0.
789  [  OK  ] Started Getty on tty1.
790  [  OK  ] Reached target Login Prompts.
791  [  OK  ] Reached target Multi-User System.
792  [  OK  ] Reached target Graphical Interface.
793           Starting Update UTMP about System Runlevel Changes...
794  [  OK  ] Started Update UTMP about System Runlevel Changes.
795  CentOS Linux 8
796  Kernel 4.19.0-xilinx-v2019.2 on an aarch64
797  [serial_line.py]: CentOS Linux 8 is ready!
798  [serial_line.py]: Script's life is coming to an end... Arrivederci!
```

Summary and Future Plans

Summarizing, the CI/CD pipelines:

- Compile the firmware and the software
- Build the root filesystem
- Deploy the images and the root filesystem
- Boot the board and monitor its booting process

The board is ready to be extensively tested! Peripherals and components on the PCB can be tested using this infrastructure in order to test their functionality.

- Ethernet connectivity
- AXI Chip2Chip communication
- I2C, UART, SPI connectivity
- …

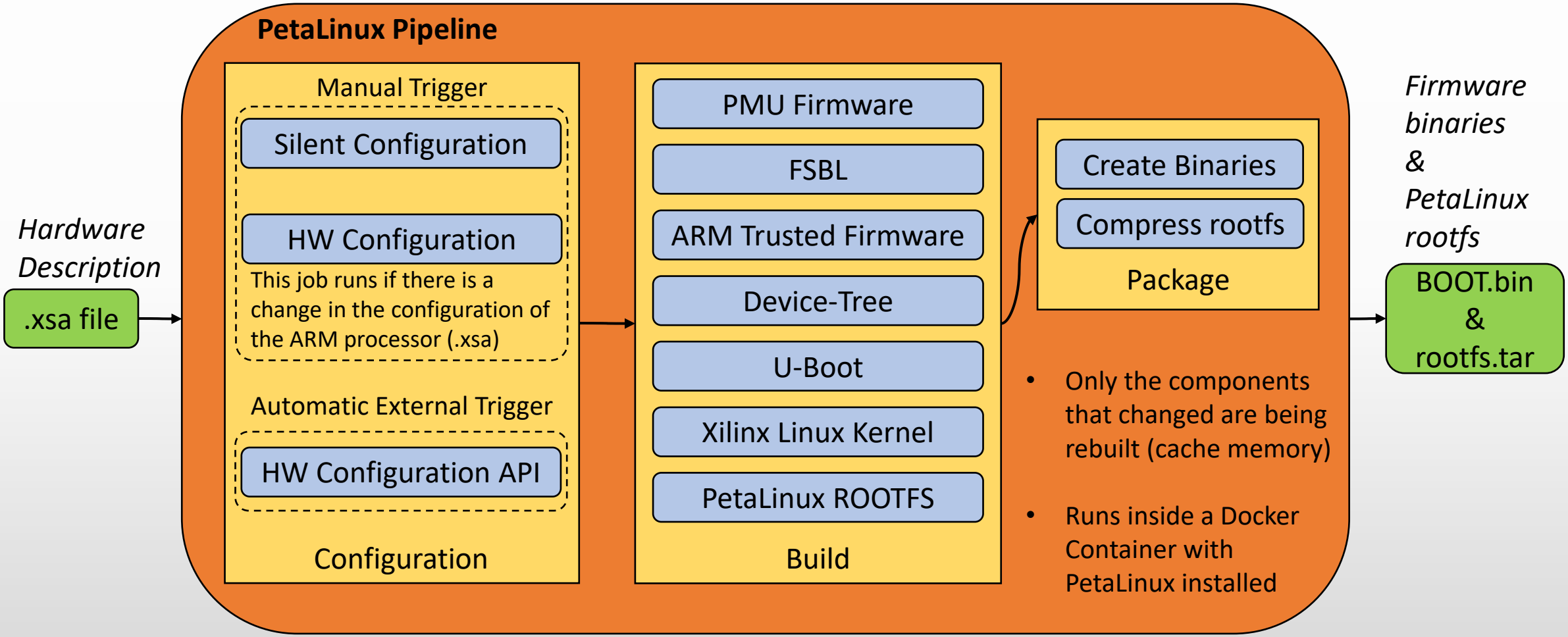
Merci beaucoup pour votre attention!

Any Questions?

Contacts:
petr.zejdl(at)cern.ch
marc.dobson(at)cern.ch
vasileios.amoiridis(at)cern.ch

BACKUP SLIDES

PetaLinux Pipeline



Extra Scripts

boot.tcl *(Using xsct console)*

- Connects to the board through JTAG
- Downloads the default binaries and boots the board

pdu.py *(using pexpect package)*

- Connects to the Power Distribution Unit (PDU) in the lab and power cycles the board

serial_line.py *(using pexpect package)*

- Reads the serial line output of the board while it is booting
- Prints out to the user the output of the serial line and checks the state of the booting process
- Generates an error if the board stuck during booting

runme.sh

- Runs the previous scripts in the correct order