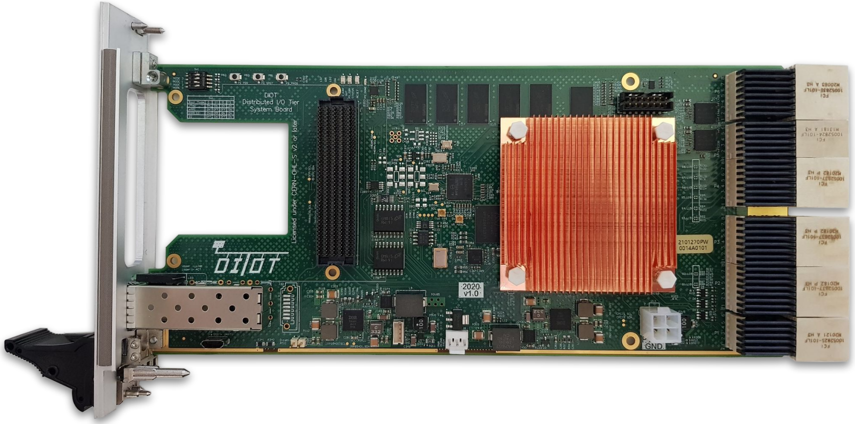
Distributed I/O Tier System Board Architecture



Presenters: Vasileios Amoiridis, Bernard Guncic
Acknowledgements: Greg Daniluk, Juan David Gonzalez Cobas, Alen Arias Vazquez

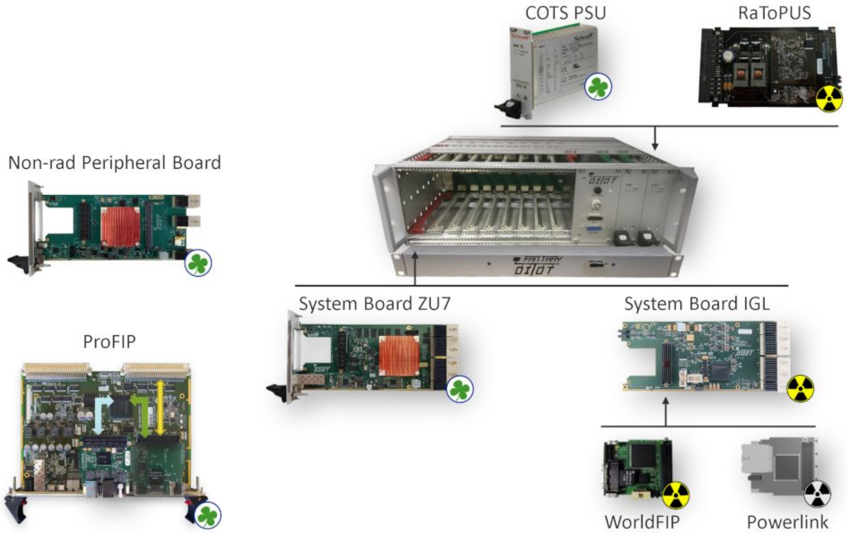
Presentation Overview

- DI/OT System General Overview
- Software
- Gateware

DI/OT Architecture

- What is the Distributed I/O Tier?
  - Essentially, it is electronics that:
    - Are close to particle accelerator
    - Non Rad-tol or Rad-tol
    - Communicate with master over fieldbus or Ethernet

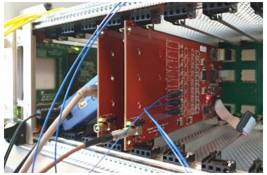
- Non-rad Variant
  - System board: ZYNQ UltraScale+ MPSoC
  - PSU: COTS
  - DI/OT management software and gateware
  - FMC Carrier: Kintex Ultrascale (optional)



More info on the Rad-tol variant:
https://indico.cern.ch/event/1207406/contributions/5078442/

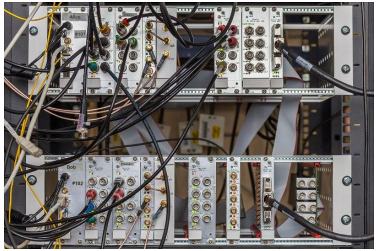
DI/OT Applications (non-rad)

BE-GM / BE-CEM



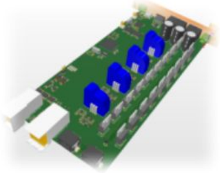
Frequency Scanning Interferometry (FSI)

Warsaw Univ. of Technology



Quantum Computing Control

BE-CEM

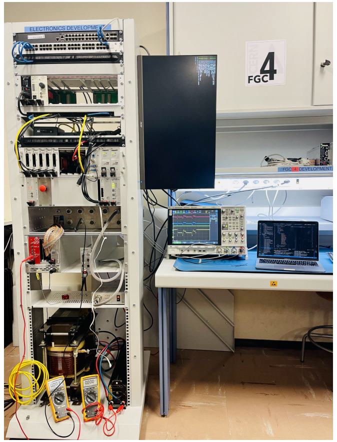


Temperature readout and motor driving for SAMbuCa

SY-ABT

Fast Interlock Detection System

SY-EPC

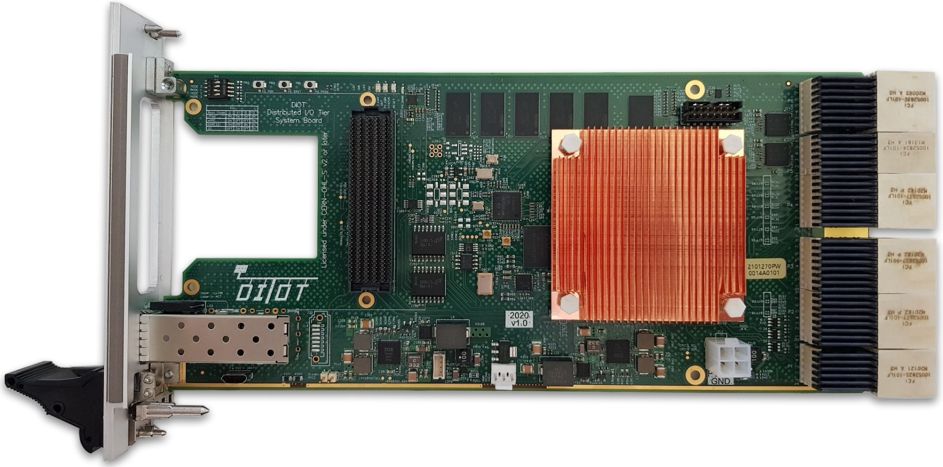


FGC4 Platform

Around 500 DI/OT Systems will be deployed during LS3.

DI/OT System Board

- ZYNQ UltraScale+ MPSoC
- Custom PCB (No SoM)
- 8GB of RAM
- FMC carrier slot
- RJ45 connector for Network
- Multiple Sensors for monitoring
- FEC-OS Kernel & Distribution
- Software Reference Design
- Gateware Reference Design

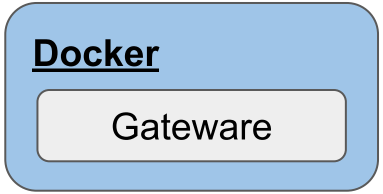


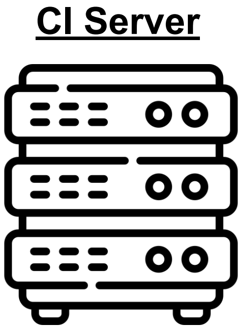
***Disclaimer**: The aforementioned specs are from the ongoing design for v3 of the System Board. More info: https://ohwr.org/project/diot-sb-zu/wikis/home*

Boot Image Creation

- GitLab Continuous Integration is used
  - From HDL to BOOT.BIN
- Docker Container with the necessary tools
  - Picked up by CI jobs
- Yocto or PetaLinux are **not** used
  - Tcl scripts and Xilinx repositories are used instead
  - Faster, easier, highest control over generated images

Build Infrastructure



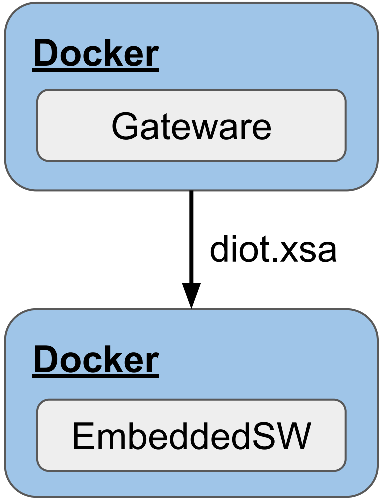


Gateware is generated through `tcl` scripts that recreate and build the Vivado project.

Build Infrastructure

CI Server

FSBL, PMUFW, DT are generated through `tcl` scripts and XSCT (Xilinx Software Command-line Tool).



Docker

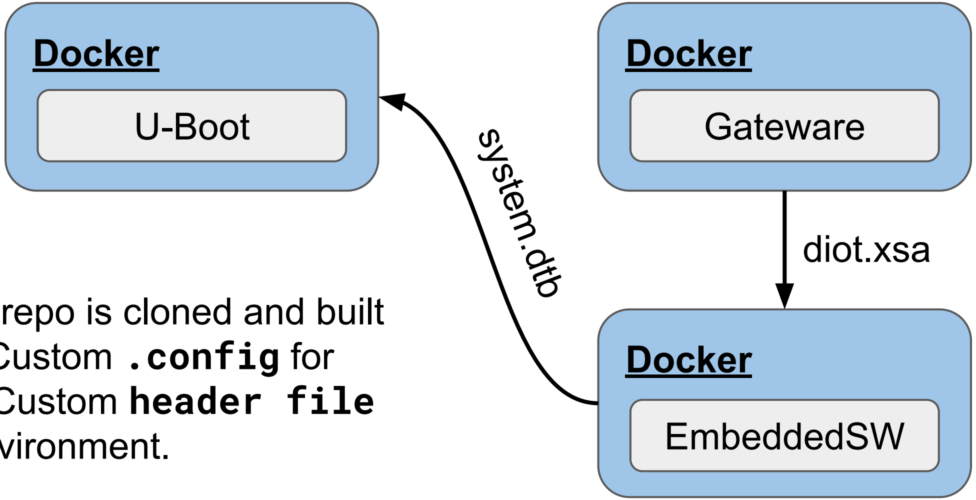
Gateware

diot.xsa

Docker

EmbeddedSW

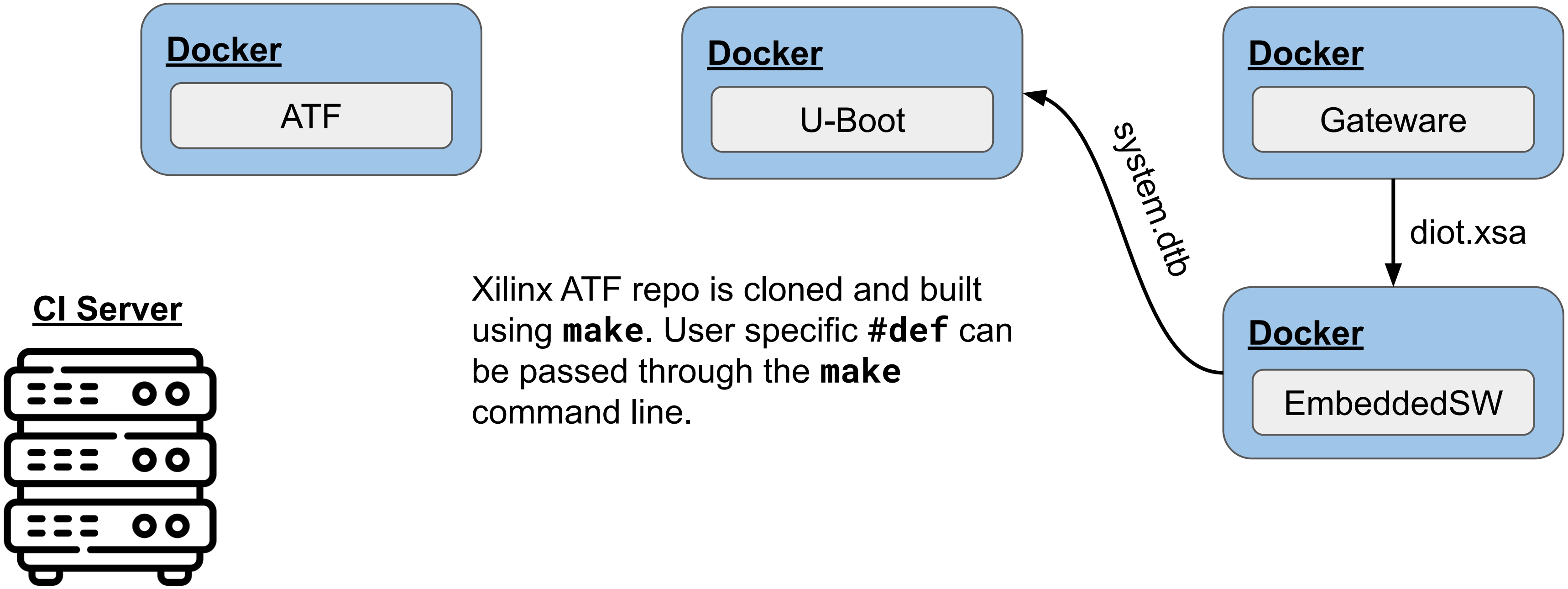
Build Infrastructure



Xilinx U-Boot repo is cloned and built using **`make`**. Custom **`.config`** for DIOT board. Custom **`header file`** for U-Boot environment.

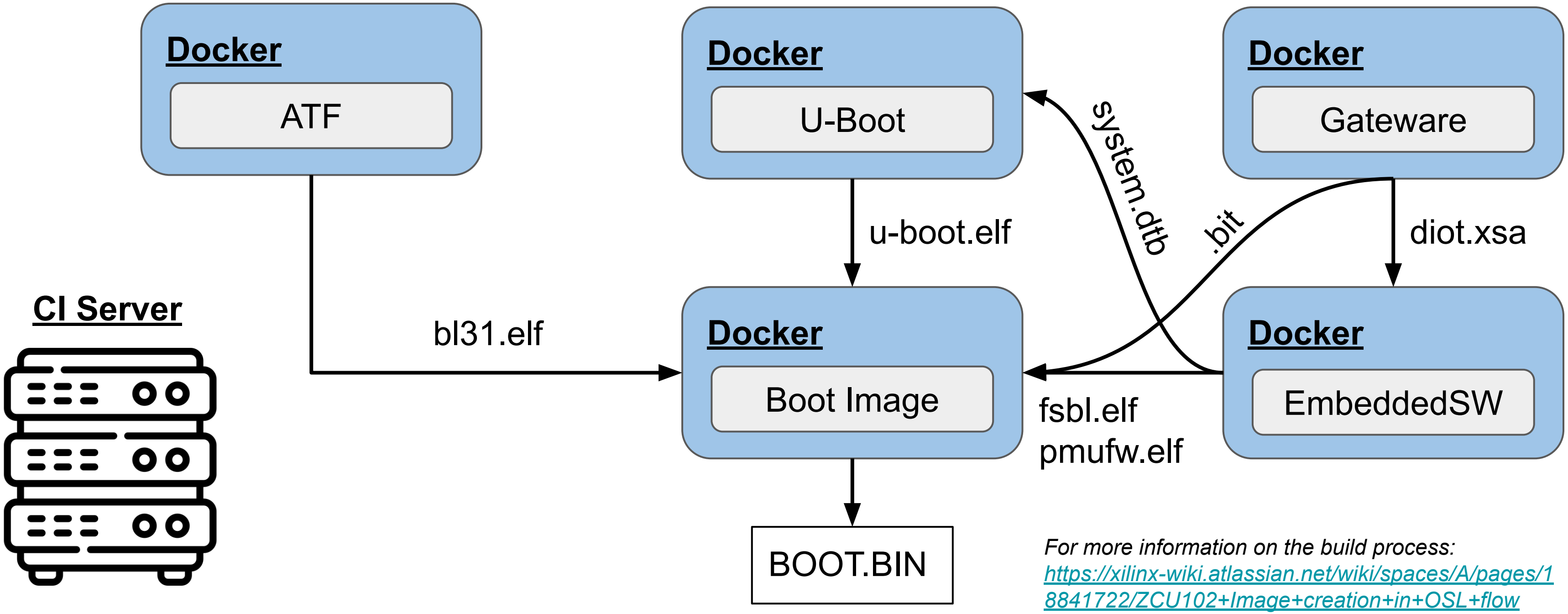
CI Server

Build Infrastructure



Xilinx ATF repo is cloned and built using **`make`**. User specific **`#def`** can be passed through the **`make`** command line.

Build Infrastructure



For more information on the build process:
https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841722/ZCU102+Image+creation+in+OSL+flow

FEC-OS: Kernel and Distribution

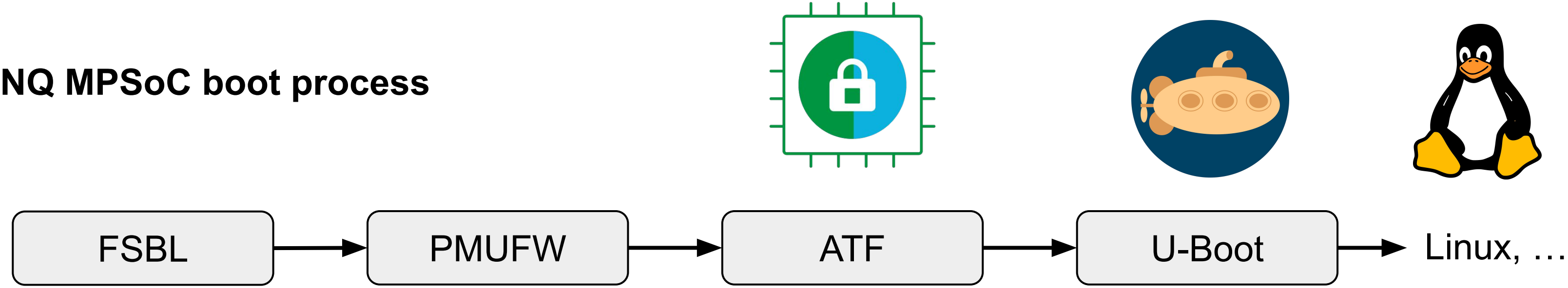
- Front-End Computer - Operating System
  - Designed and maintained by BE-CEM and BE-CSS
- FEC-OS-kernel for DIOT
  - Xilinx Linux Kernel built by the FEC-OS Infrastructure
  - aarch64 architecture
  - Custom `.config` for DIOT
  - Custom Kernel Drivers for fan tray and PSU
  - Built-in drivers, no kernel modules
- FEC-OS-Distribution
  - Live Image
  - Based on Debian 12

For more information on the FEC-OS you can attend Federico Vaga’s presentation on Friday: [ATS - Linux for FECs and SoCs]

FEC & SoC Boot Process

FEC & SoC Boot Process

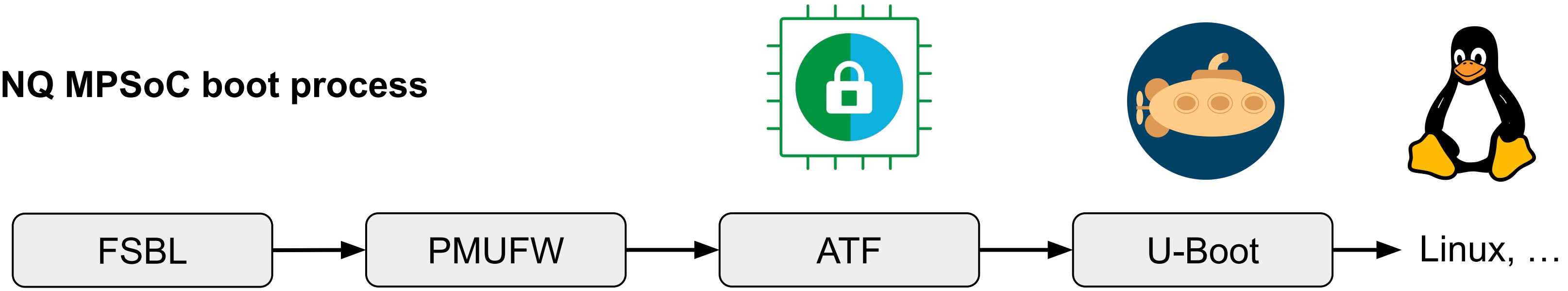
ZYNQ MPSoC boot process



FSBL → PMUFW → ATF → U-Boot → Linux, …

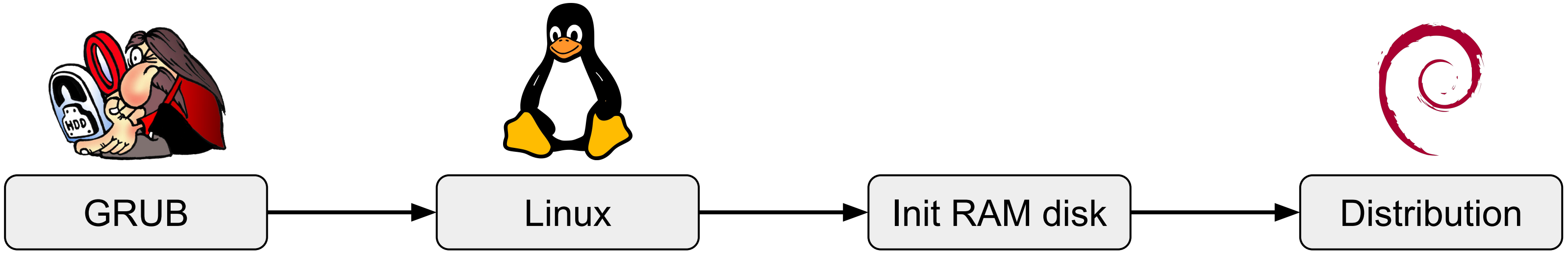
FEC & SoC Boot Process

ZYNQ MPSoC boot process



FSBL → PMUFW → ATF → U-Boot → Linux, …

FEC-OS boot process

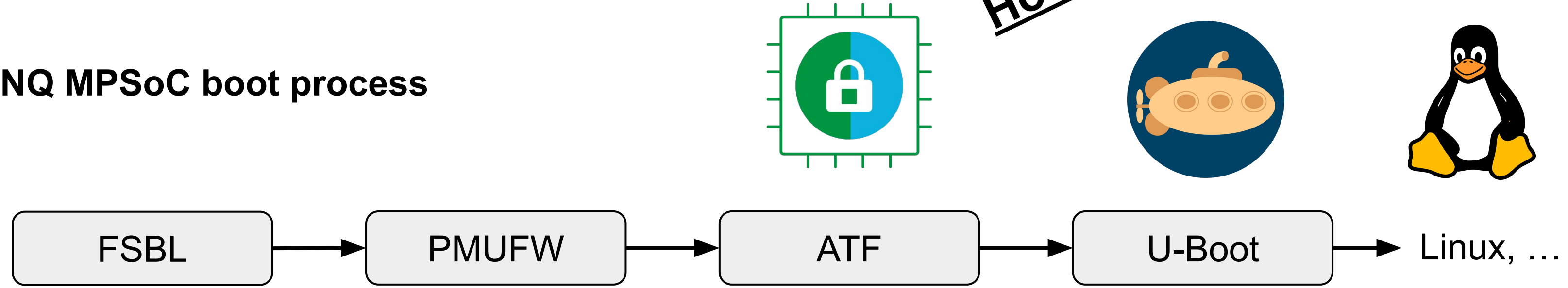


GRUB → Linux → Init RAM disk → Distribution

FEC & SoC Boot Process

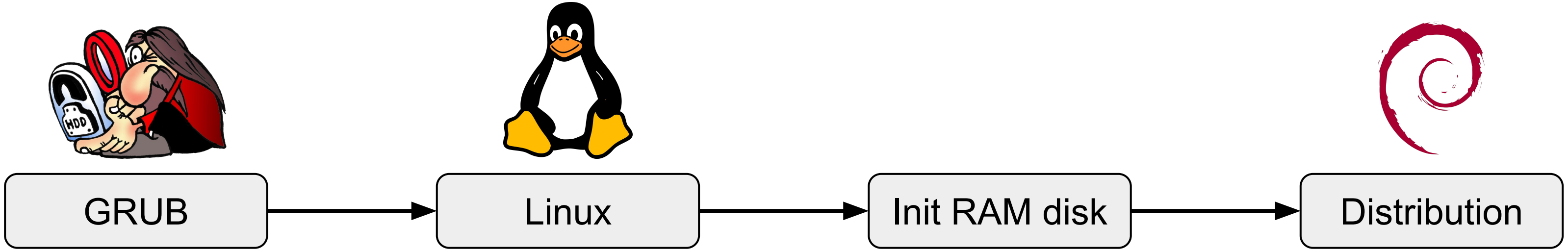
How to merge them???

ZYNQ MPSoC boot process



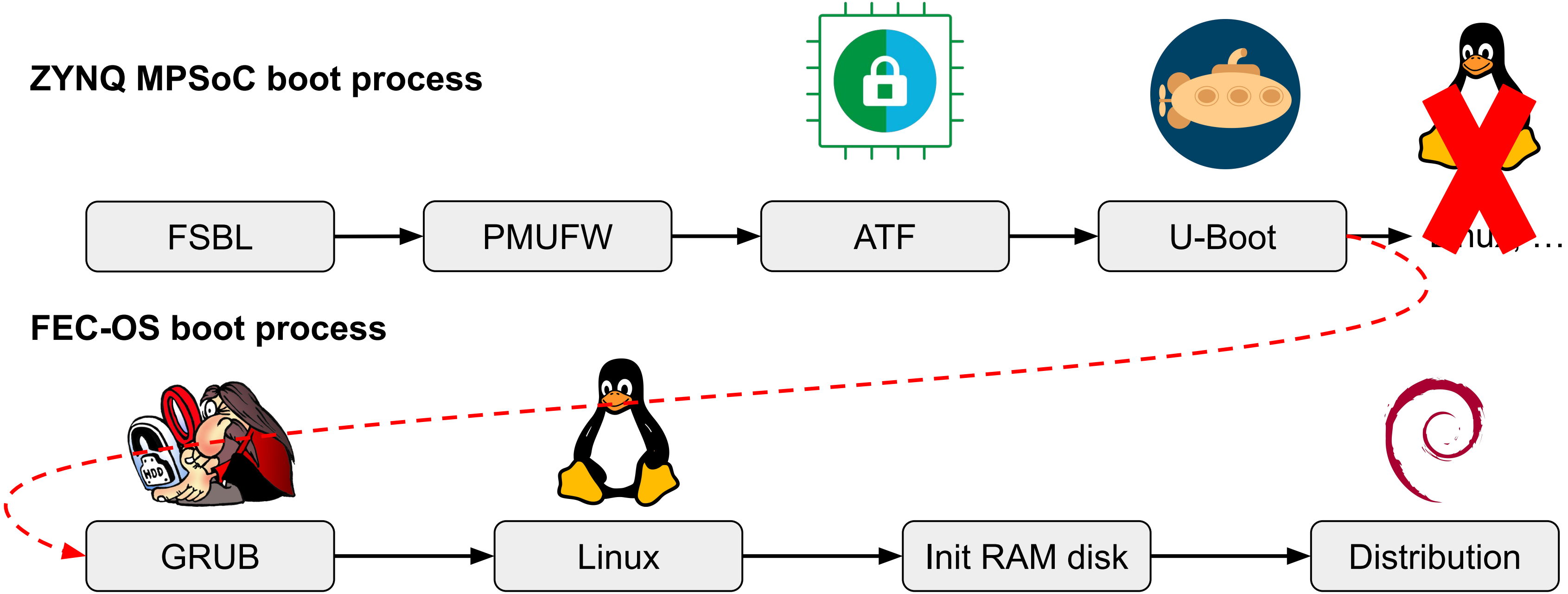
FSBL → PMUFW → ATF → U-Boot → Linux, …

FEC-OS boot process



GRUB → Linux → Init RAM disk → Distribution

FEC & SoC Boot Process



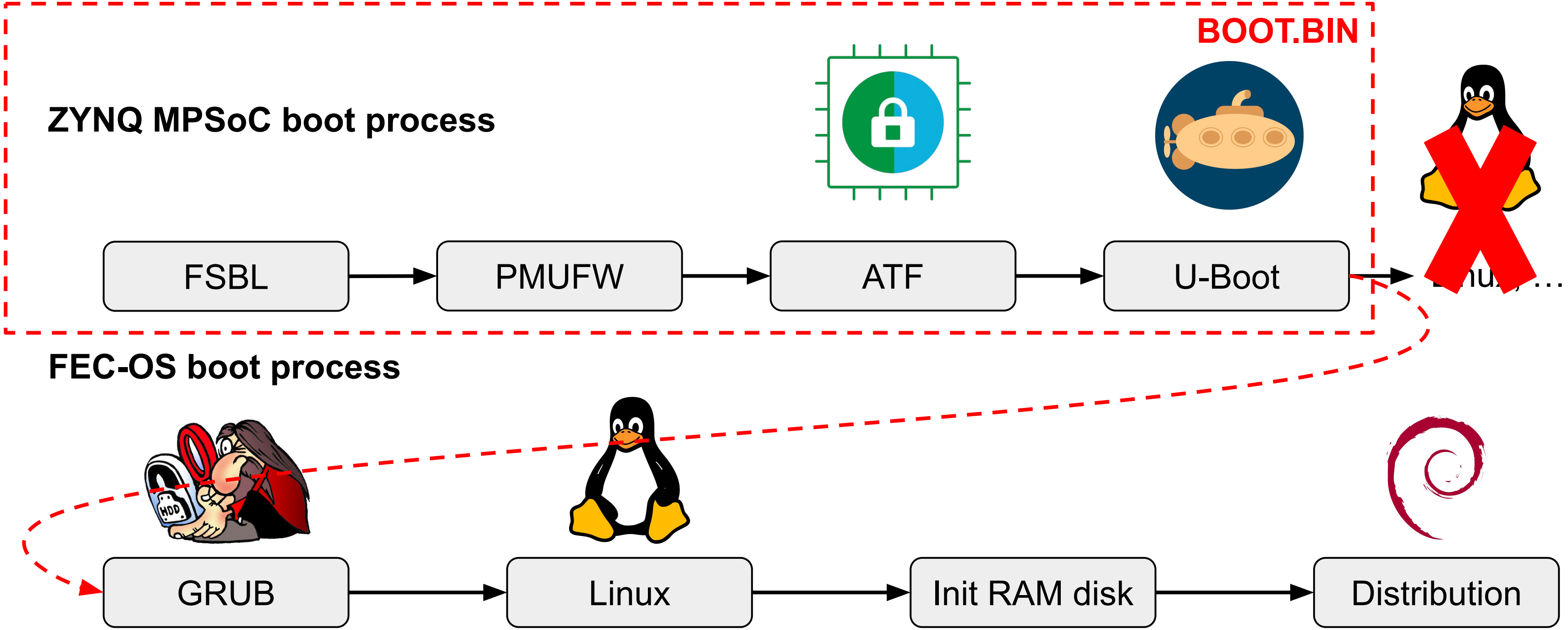
ZYNQ MPSoC boot process

FSBL → PMUFW → ATF → U-Boot → Linux, …

FEC-OS boot process

GRUB → Linux → Init RAM disk → Distribution

FEC & SoC Boot Process



BOOT.BIN

ZYNQ MPSoC boot process

FSBL → PMUFW → ATF → U-Boot → Linux, …

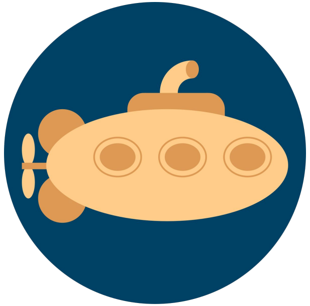
FEC-OS boot process

GRUB → Linux → Init RAM disk → Distribution

U-Boot, GRUB and SMBIOS

- How do we boot GRUB?
  - Build it for EFI and aarch64 platform
  - Fetch it through TFTP
  - Boot it using the `bootefi` U-Boot command.

- How does GRUB identifies the platform?
  - Through System Management BIOS information
  - Passed to U-Boot through the Device-Tree
  - U-Boot stores them in specific structures in memory
  - GRUB is able to read them
  - Accessible from linux-userspace with `dmidecode` command



SMBIOS to U-Boot and Userspace

SMBIOS to U-Boot and Userspace

- Device-Tree

```
smbios {
        compatible = "u-boot,sysinfo-smbios";

        smbios {
                system {
                        manufacturer = "CERN";
                        product = "DIOT";
                        version = "2";
                        serial = "1234567890";
                        family = "FSI";
                };

                baseboard {
                        manufacturer = "CERN";
                        product = "DIOT";
                        asset-tag = "TEST";
                };

                chassis {
                        manufacturer = "CERN";
                };
        };
};
```

SMBIOS to U-Boot and Userspace

- Device-Tree

```
smbios {
        compatible = "u-boot,sysinfo-smbios";

        smbios {
                system {
                        manufacturer = "CERN";
                        product = "DIOT";
                        version = "2";
                        serial = "1234567890";
                        family = "FSI";
                };

                baseboard {
                        manufacturer = "CERN";
                        product = "DIOT";
                        asset-tag = "TEST";
                };

                chassis {
                        manufacturer = "CERN";
                };
        };
};
```

- U-Boot

```
DIOT> tftpboot $efi_addr_r grub.arm64-efi.img
DIOT> tftpboot $fdt_addr_r system.dtb
DIOT> bootefi $efi_addr_r $fdt_addr_r
```

SMBIOS to U-Boot and Userspace

- Device-Tree

```
smbios {
        compatible = "u-boot,sysinfo-smbios";

        smbios {
                system {
                        manufacturer = "CERN";
                        product = "DIOT";
                        version = "2";
                        serial = "1234567890";
                        family = "FSI";
                };

                baseboard {
                        manufacturer = "CERN";
                        product = "DIOT";
                        asset-tag = "TEST";
                };

                chassis {
                        manufacturer = "CERN";
                };
        };
};
```

- U-Boot

```
DIOT> tftpboot $efi_addr_r grub.arm64-efi.img
DIOT> tftpboot $fdt_addr_r system.dtb
DIOT> bootefi $efi_addr_r $fdt_addr_r
```

- Grub + Linux

```
…
  Booting `CERN - DIOT - ZynqMP'

GNU GRUB 2
EFI stub: Booting Linux Kernel...
EFI stub: Using DTB from configuration table
EFI stub: Exiting boot services…
…
[0.000000] Booting Linux on physical CPU
0x0000000000 [0x410fd034]
…
[0.000000] efi: EFI v2.90 by Das U-Boot
[0.000000] efi: RTPROP=0x77c1a040
SMBIOS=0x77c16000 MEMRESERVE=0x57202040
```

SMBIOS to U-Boot and Userspace

- Device-Tree

```
smbios {
        compatible = "u-boot,sysinfo-smbios";

        smbios {
                system {
                        manufacturer = "CERN";
                        product = "DIOT";
                        version = "2";
                        serial = "1234567890";
                        family = "FSI";
                };

                baseboard {
                        manufacturer = "CERN";
                        product = "DIOT";
                        asset-tag = "TEST";
                };

                chassis {
                        manufacturer = "CERN";
                };
        };
};
```

- Userspace

```
root@diot-vamoirid:~>dmidecode
# dmidecode 3.3
# SMBIOS entry point at 0x77c16000
Found SMBIOS entry point in EFI, reading table from
/dev/mem.
SMBIOS 3.0 present.
7 structures occupying 250 bytes.
Table at 0x77C16020.

Handle 0x0000, DMI type 0, 24 bytes
BIOS Information
        Vendor: U-Boot
        Version: 2022.01-dirty
        Release Date: 01/01/2022
        ROM Size: 64 kB
        Characteristics:
                PCI is supported
                BIOS is upgradeable
                Selectable boot is supported
                Targeted content distribution is supported
                UEFI is supported
        BIOS Revision: 22.1
…
```

SMBIOS to U-Boot and Userspace

- Device-Tree

```
smbios {
        compatible = "u-boot,sysinfo-smbios";

        smbios {
                system {
                        manufacturer = "CERN";
                        product = "DIOT";
                        version = "2";
                        serial = "1234567890";
                        family = "FSI";
                };

                baseboard {
                        manufacturer = "CERN";
                        product = "DIOT";
                        asset-tag = "TEST";
                };

                chassis {
                        manufacturer = "CERN";
                };
        };
};
```

- Userspace

```
root@diot-vamoirid:~>dmidecode
# dmi
# SMB
Found                                         ble from
/dev/
SMBIO
7 str
Table

Handl
BIOS

        ROM Size: 64 kB
        Characteristics:
                PCI is supported
                BIOS is upgradeable
                Selectable boot is supported
                Targeted content distribution is supported
                UEFI is supported
        BIOS Revision: 22.1
…
```

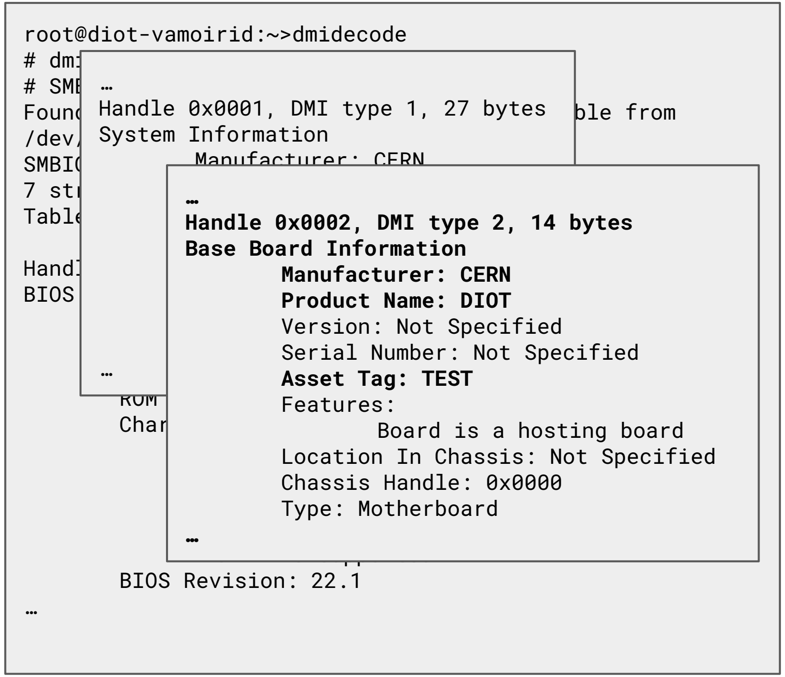
```
…
Handle 0x0001, DMI type 1, 27 bytes
System Information
        Manufacturer: CERN
        Product Name: DIOT
        Version: 2
        Serial Number: 1234567890
        UUID: Not Settable
        Wake-up Type: Reserved
        SKU Number: TEST
        Family: FSI
…
```

SMBIOS to U-Boot and Userspace

- Device-Tree

```
smbios {
	compatible = "u-boot,sysinfo-smbios";

	smbios {
		system {
			manufacturer = "CERN";
			product = "DIOT";
			version = "2";
			serial = "1234567890";
			family = "FSI";
		};

		baseboard {
			manufacturer = "CERN";
			product = "DIOT";
			asset-tag = "TEST";
		};

		chassis {
			manufacturer = "CERN";
		};
	};
};
```

- Userspace



```
root@diot-vamoirid:~>dmidecode
# dmi
# SMB
Found                                              ble from
/dev/
SMBIO
7 str
Table

Handl
BIOS
   ROM
   Char
   BIOS Revision: 22.1
…
```

```
…
Handle 0x0001, DMI type 1, 27 bytes
System Information
	Manufacturer: CERN
…
```

```
…
Handle 0x0002, DMI type 2, 14 bytes
Base Board Information
	Manufacturer: CERN
	Product Name: DIOT
	Version: Not Specified
	Serial Number: Not Specified
	Asset Tag: TEST
	Features:
		Board is a hosting board
	Location In Chassis: Not Specified
	Chassis Handle: 0x0000
	Type: Motherboard
…
```

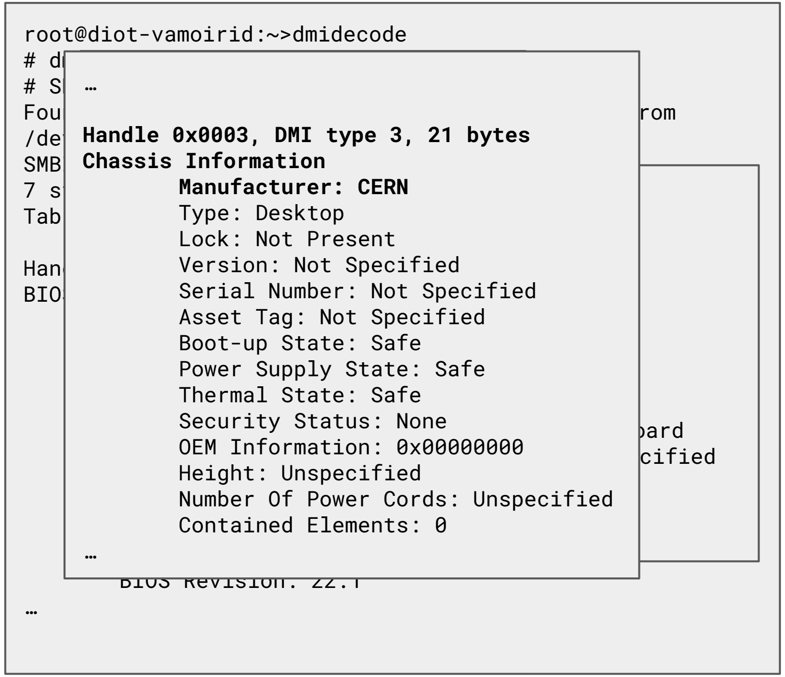
SMBIOS to U-Boot and Userspace

More on SMBIOS:
https://www.dmtf.org/standards/smbios

- Device-Tree

```
smbios {
        compatible = "u-boot,sysinfo-smbios";

        smbios {
                system {
                        manufacturer = "CERN";
                        product = "DIOT";
                        version = "2";
                        serial = "1234567890";
                        family = "FSI";
                };

                baseboard {
                        manufacturer = "CERN";
                        product = "DIOT";
                        asset-tag = "TEST";
                };

                chassis {
                        manufacturer = "CERN";
                };
        };
};
```

- Userspace



```
root@diot-vamoirid:~>dmidecode
# d
# S
Fou                                                      rom
/de
SMB
7 s
Tab

Han
BIO
                                                         ard
                                                         cified

         BIOS Revision: 22.1

…
```

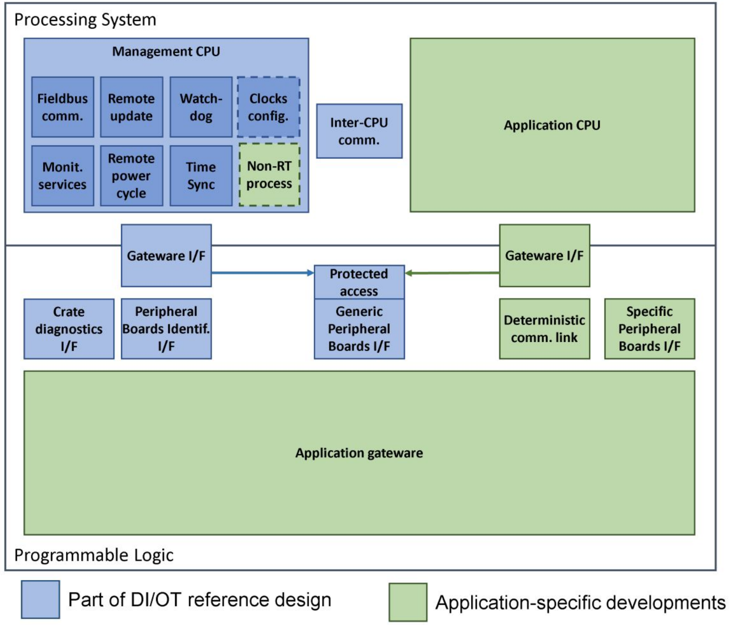
```
…

Handle 0x0003, DMI type 3, 21 bytes
Chassis Information
        Manufacturer: CERN
        Type: Desktop
        Lock: Not Present
        Version: Not Specified
        Serial Number: Not Specified
        Asset Tag: Not Specified
        Boot-up State: Safe
        Power Supply State: Safe
        Thermal State: Safe
        Security Status: None
        OEM Information: 0x00000000
        Height: Unspecified
        Number Of Power Cords: Unspecified
        Contained Elements: 0

…
```

DIOT Software Applications (Ongoing)

- Library that provides monitoring services
    - Temperatures
    - Voltages
    - Currents
    - Monitoring through the Network with Grafana
- PTP Synchronization
    - Clock synchronization with PTP Master (<100ns)
    - Clock Frequency Adjustment
- Communication with Real-Time CPU
    - Inter-Processor Interrupt, Shared memory, remoteproc, etc…
- Recovery Services - Remote Power Cycle
    - Kernel panic, halted system, failed boot, etc…

System Board Architecture

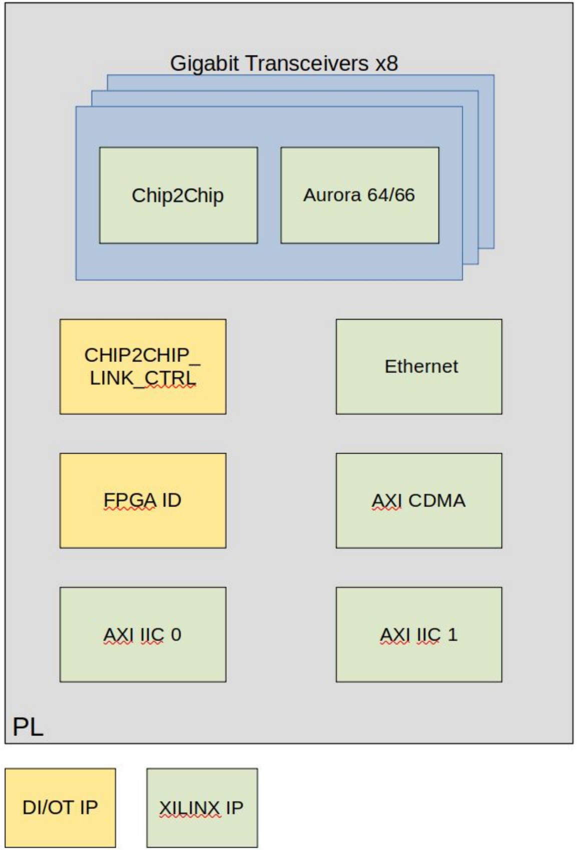
- Management CPU
  - Runs DI/OT default software services
- Application CPU
  - Runs user real-time tasks
- DI/OT crate specific
  - Peripheral Board (PB) detection
  - Service mode
  - I2C shared bus over backplane



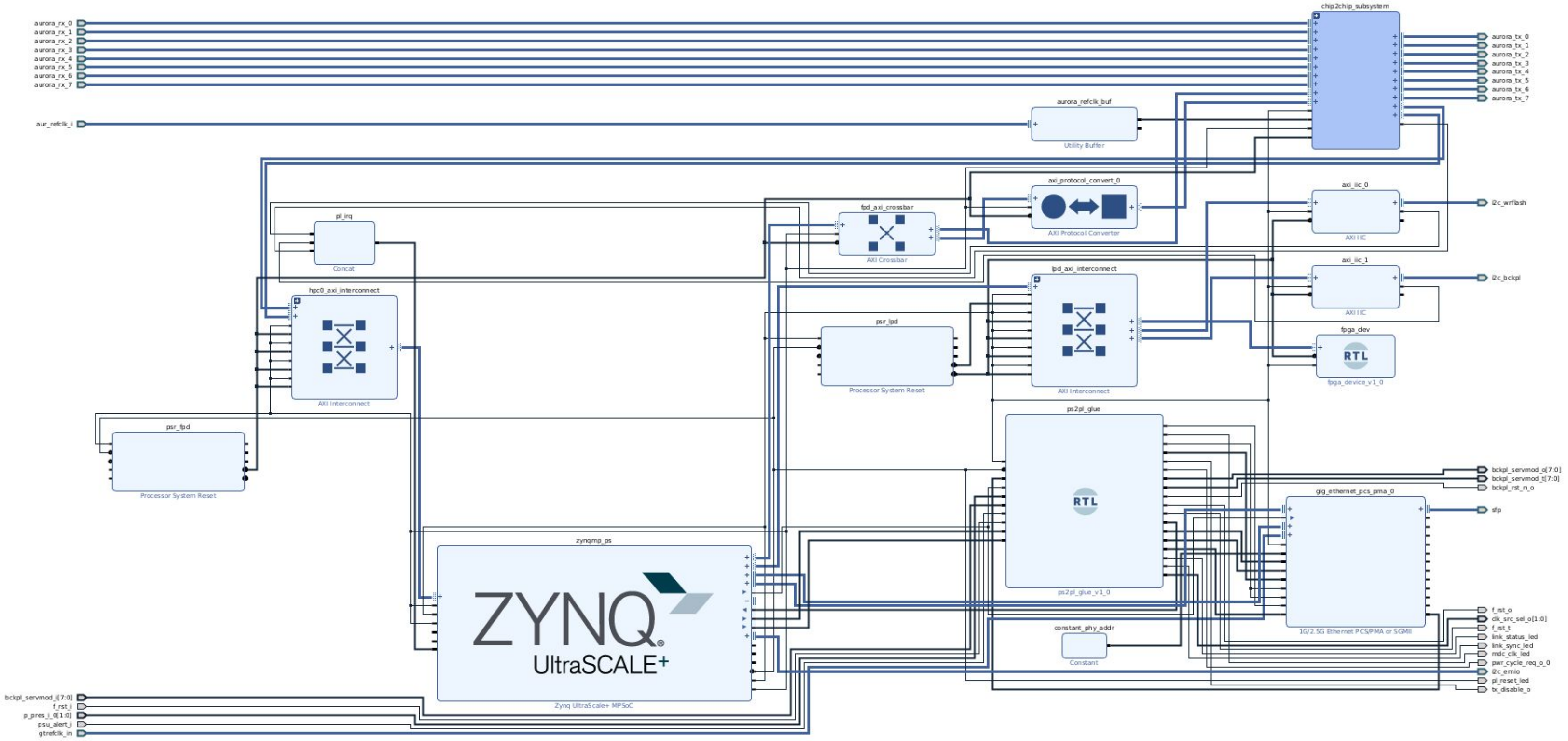
DIOT Architecture v0.1.pdf

Gateware

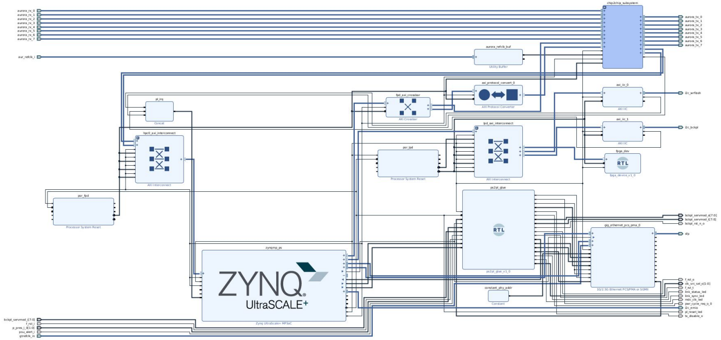
- Chip2Chip LINK CTRL
  - Status & Reset of each link
- FPGA ID
  - Vendor, Device, Name, Build Date, Hash (gitlab)...
- I^{2}C
  - EMIO (sensors, clock)
  - PL (Backplane, EEPROMs)
- AXI CDMA
- 8 x GT Links to PB
  - Chip2Chip + Aurora 64/66b up to 12.5Gbps
- Ethernet PCS/PMA



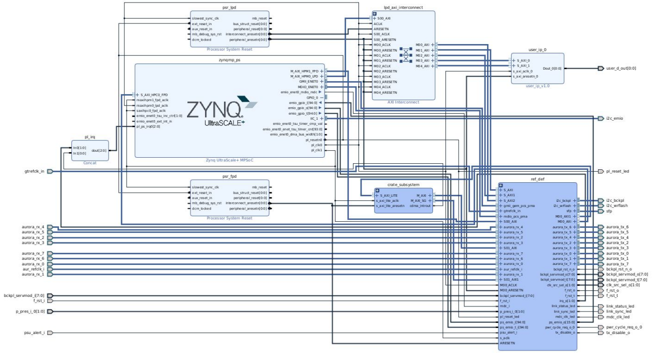
Reference Design



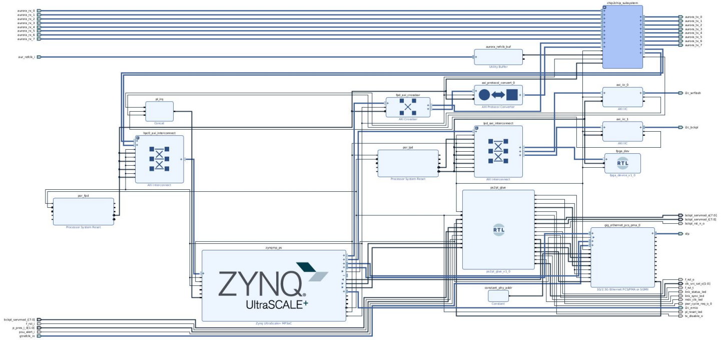
Reference Design



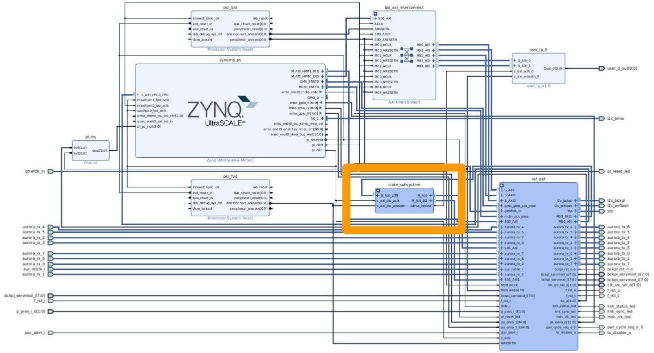
Hierarchies



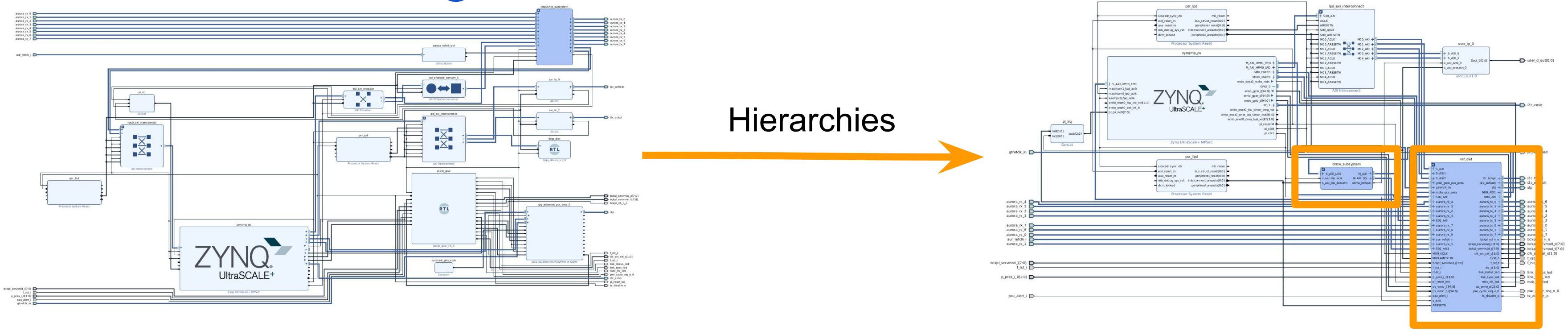
Reference Design



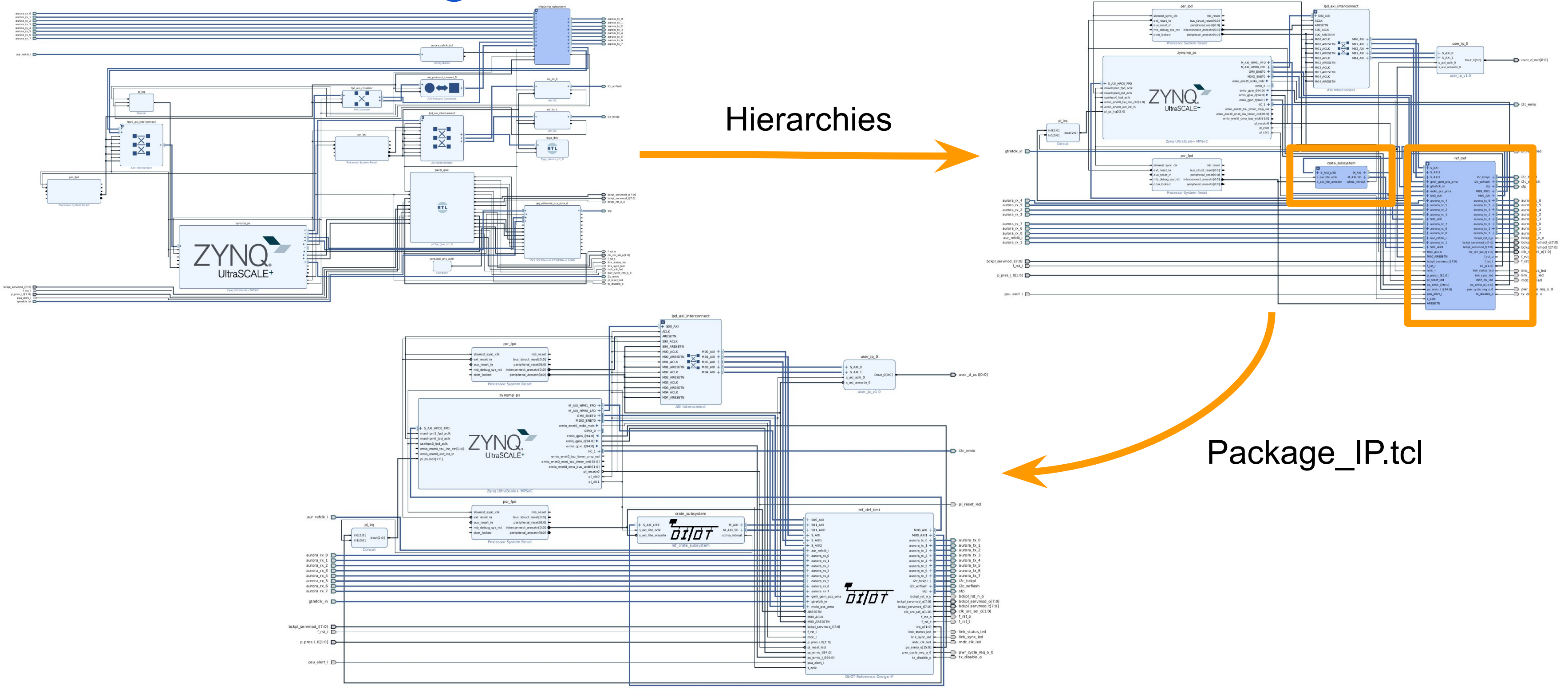
Hierarchies



Reference Design



Reference Design

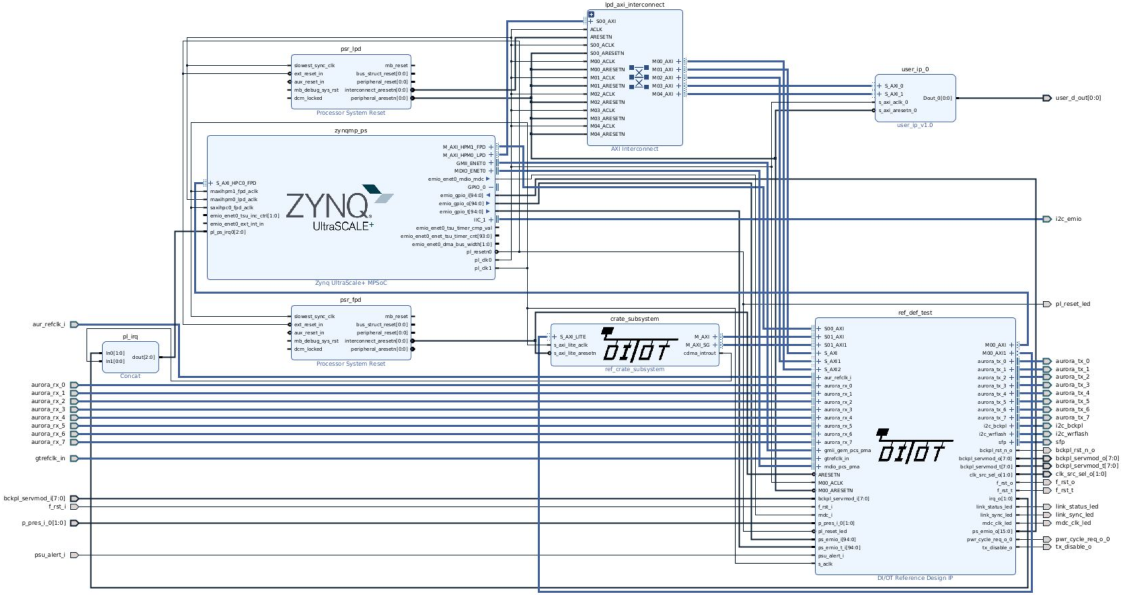


Hierarchies

Package_IP.tcl

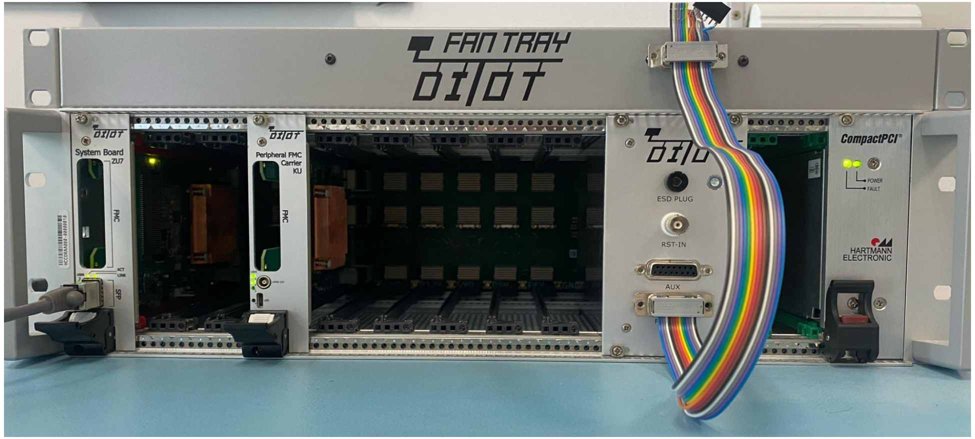
Reference Design with DI/OT IP Cores

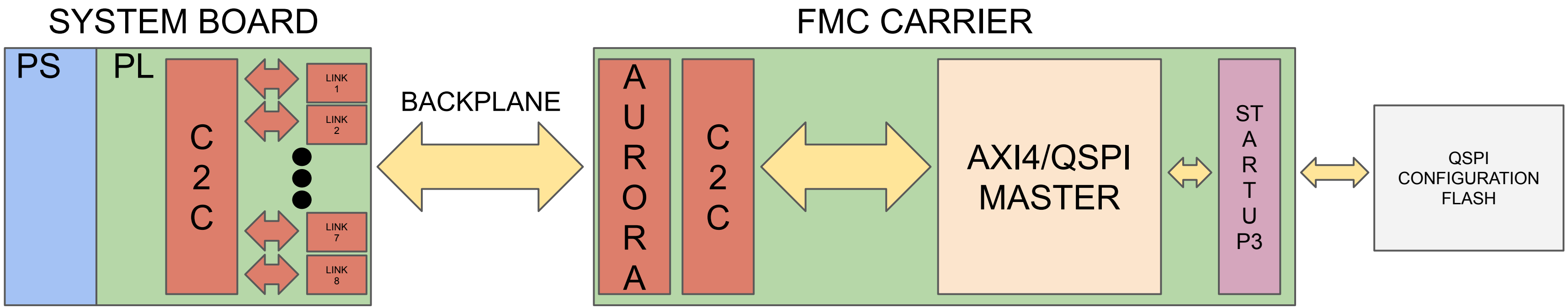
- User friendly design
- Users can still modify PS
- Easier to maintain
- Packaged IPs delivered with
  - User guide (internal block dia)
  - Version Log file
- Adding users logic
  - By using IP Catalog
  - By sourcing tcl script



Remote programming

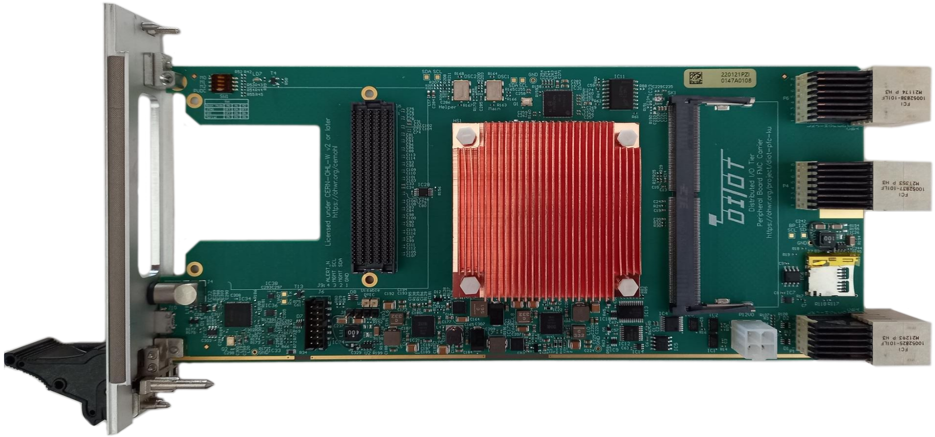
- Crate up to 8 peripheral boards
- Communication over C2C + Aurora 64/66b
- Speed up to 12.5 Gbps

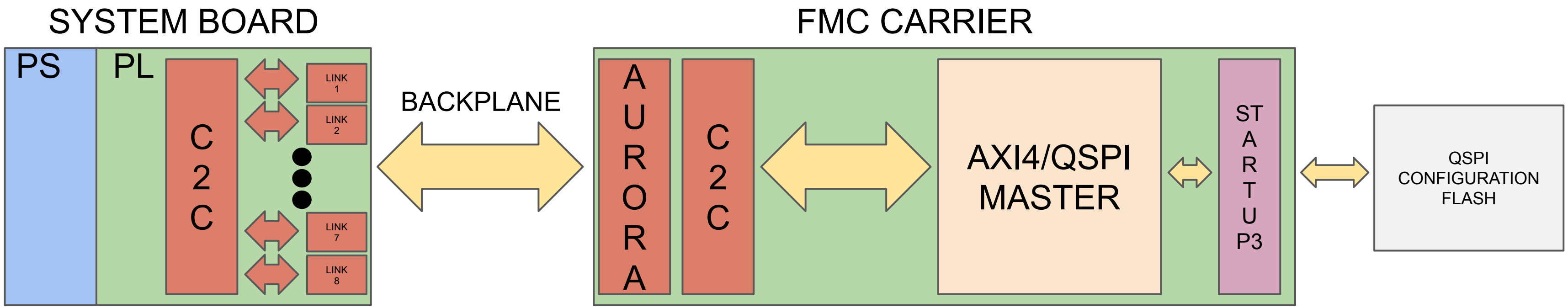




Remote programming

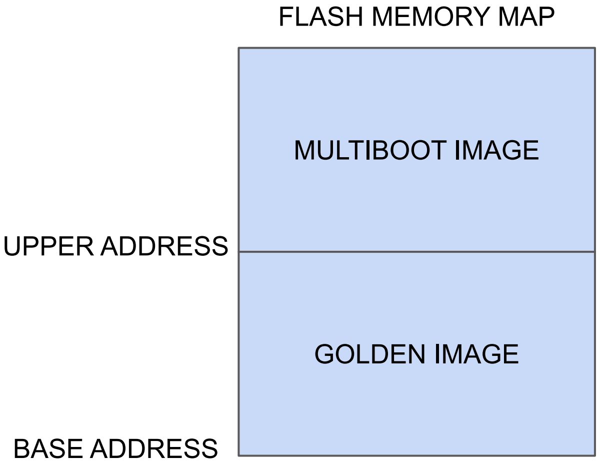
- Kintex Ultrascale (XCKU035)
- Custom PCB (No SoM)
- FMC carrier slot
- DDR4 SO-DIMM
- QSPI Flash

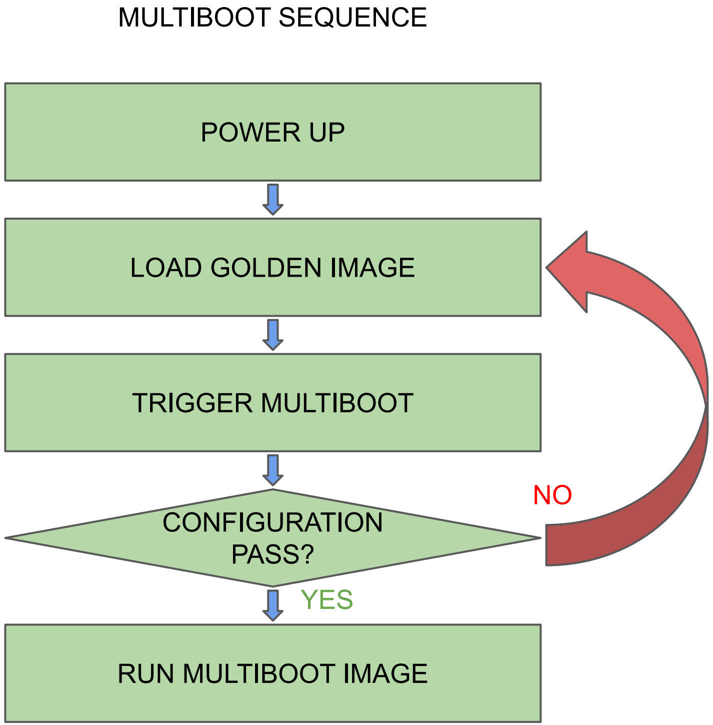




Multiboot on PB

- Multiple bitstream same flash
- Save mechanism for remote reprogramming





Summary

- SMBIOS Information to GRUB through U-BOOT
- FEC-OS Integration
- Standardized gateware design
- Clean split between platform IPs and users specific logic

More information can be found here:

- https://ohwr.org/project/diot-sb-zu/wikis/home
- https://gitlab.cern.ch/be-cem-edl/diot/zynqmp/boot-image

Backup slides

Boot Image Creation

- Gateware, FSBL, PMUFW and Device-Tree are generated through custom tcl scripts and, embeddedsw and device-tree repositories.
- ARM trusted firmware is generated from the atf-xlnx repository.
- U-Boot is generated from the u-boot-xlnx repository. Use of custom CONFIG and Environment.
- The *BOOT.BIN* is generated with the bootgen-xlnx tool.
- CI pipelines with Docker Containers are used in every step. **No PetaLinux used.**

- diot.xsa
- pl.bit
- fsbl.elf
- pmufw.elf
- system.dtb

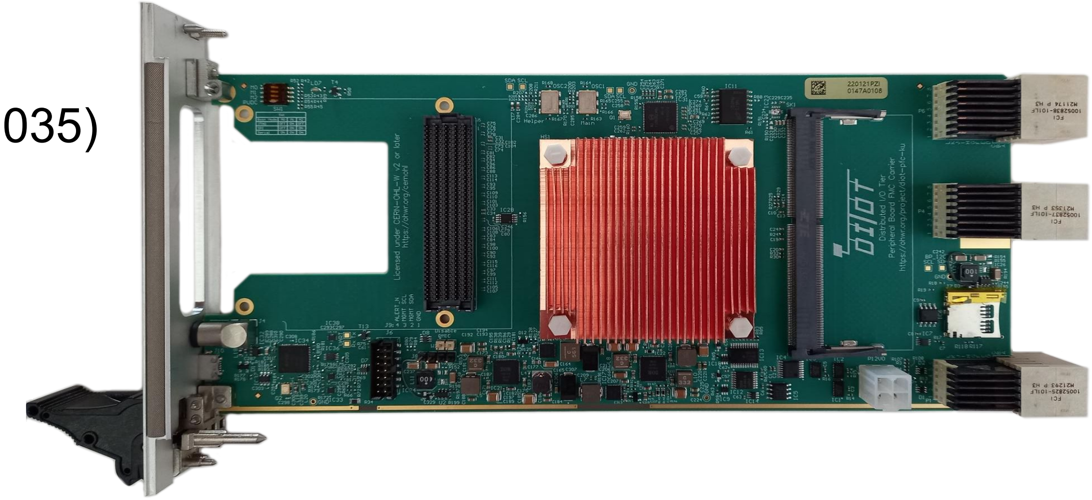
- bl31.elf

- u-boot.elf

- boot.bin

DIOT FMC Carrier

- Kintex Ultrascale (XCKU035)
- Custom PCB (No SoM)
- FMC carrier slot
- DDR4 SO-DIMM
- QSPI Flash
- Silab Clock Generator
- Multiple Sensors for monitoring
- Gateware Reference Design



***Disclaimer**: More info about features and gateware reference design*

- *https://ohwr.org/project/diot/wikis/diot-fmc-carrier*
- *https://gitlab.cern.ch/be-cem-edl/diot/fmc-carrier*