Yocto Embedded Linux for SoC based AMCs

Latest developments for demanding applications

Patrick Huesmann, Sven Stubbe, Carsten Duelsen
2022-12-08

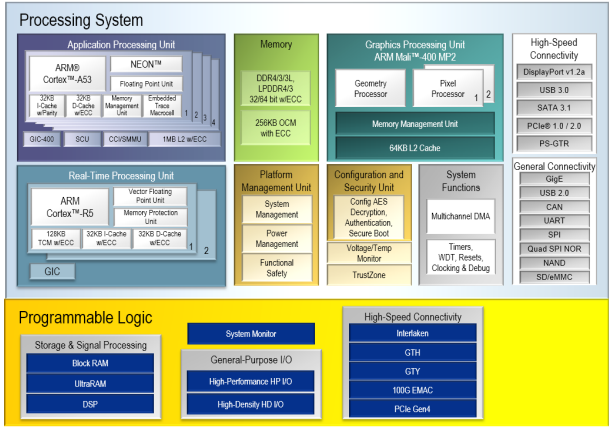
MicroTCA TECHNOLOGY LAB

DESY.

Contents

What is a SoC in our context?

- System on a chip: Integrating several components
- Processing System: One or more ARM processors with standard interfaces for peripherals
  - USB, SATA, Ethernet, DP
  - PCIe (host and device function)
  - CAN, UART, SPI, SD/eMMC
- Programmable Logic
  - Logic cells
  - Block RAM
  - GPIO
  - High-speed I/O

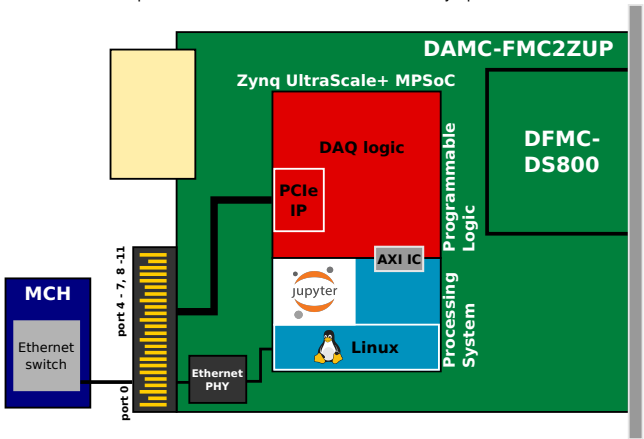


Xilinx Zynq UltraScale+ EG Block Diagram

Why Linux on AMC boards?

- Networking works out of the box
- Lots of standard software available
- Lots of device drivers available
- Standard protocols can be quickly implemented
- SSH for development/debugging
- Can easily reconfigure PL at runtime
- High-level languages (Python etc.) available

Example: DAMC-FMC2ZUP FMC+ Carrier with Zynq UltraScale+



from: "Developing for SoC-based AMCs", Jan Marjanovic, MTCA WS '21

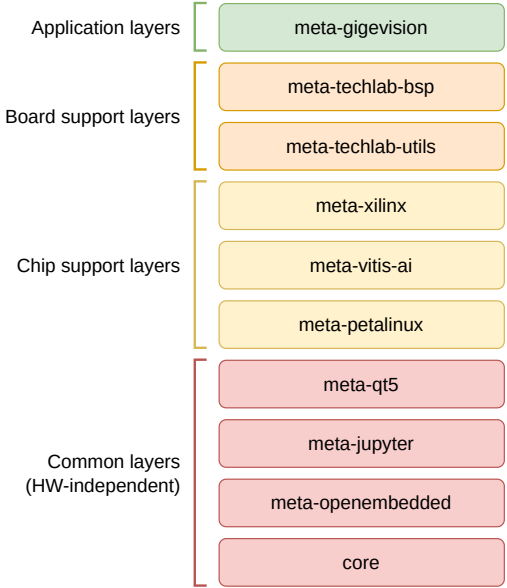
Yocto Linux basics

”It’s not an embedded Linux Distribution, it creates a custom one for you”
see also: `https://www.yoctoproject.org/`

- Provides different **layers** containing metadata (”recipes”) about software packages
- And a **tool** (`bitbake`) to compile an actual Linux system out of them
- The layers are usually open source repositories that are referenced in a **manifest** XML file which fully describes the codebase used in a Yocto system

Yocto Layer Structure

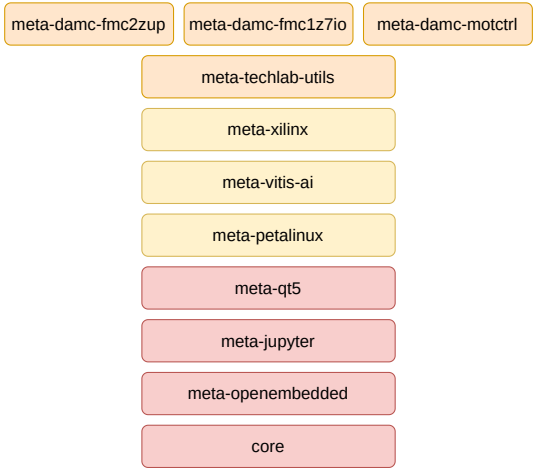
- Target Linux system is built from a stack of layers
- Layers are ordered by priority
  From ”universal” or ”generic” layers to more specialized ones
- Higher priority layers can override settings or configurations in lower layers
  Making possible to adapt ”generic” recipes to special hardware or applications
- The end result is an image for a storage medium that the target board can boot from



Restructuring of TechLab Yocto BSP

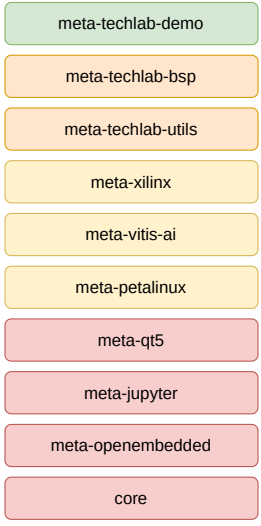
Legacy TechLab BSP Structure

- Separate BSP layer for each board ⟶
- Each BSP layer also included a ”demo application” (PL design & demo executables)
- With more boards coming up, there’s redundancy and risk of code duplication
- To avoid redundancy and code duplication, we refactored the BSP layers into a new structure



New TechLab BSP Structure

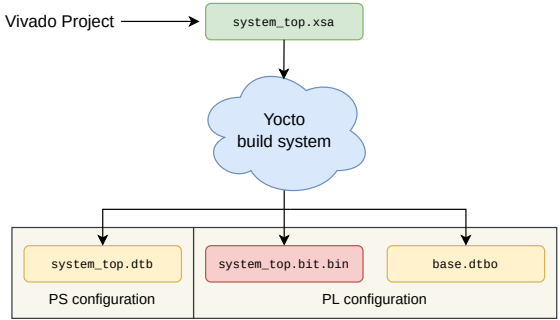
- In the new structure, BSPs for all boards are integrated in one single `meta-techlab-bsp` layer
  - Less redundancy
  - Easier maintenance
  - For product maintenance like migration to a newer Vivado version, only one layer has to be touched
- Demo code / bitstreams have been moved into a separate `meta-techlab-demo` layer
  - Demo layer serves as a starting point for user applications



Support for multiple PL designs per image

Conventional flow with a single PL design

- Vivado project artifact `.xsa` contains configuration for both PS and PL
- Yocto/Petalinux build system converts it to config files:
  - For PS configuration: Device tree blob `.dtb` which is loaded by U-Boot and passed to the Linux kernel
  - For PL configuration: Bitstream `.bit.bin` and device tree overlay `.dtbo` which describes the IP cores in the PL to the Linux system
  - PL configuration is written to the rootfs and loaded at runtime from userspace.

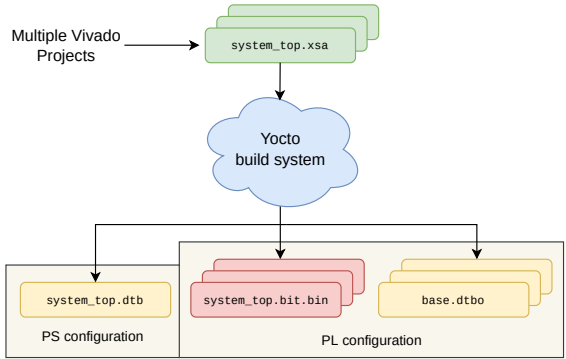


Added support for bundling multiple designs in one image

In the new structure, the application layer can provide multiple PL **variants**

- For each variant, a separate `.xsa` file is put in the application layer
- The build system creates a set of `.bin` and `.dtbo` for each `.xsa` file
- The application layer provides a user-defined init script that determines which of the PL variants to load.

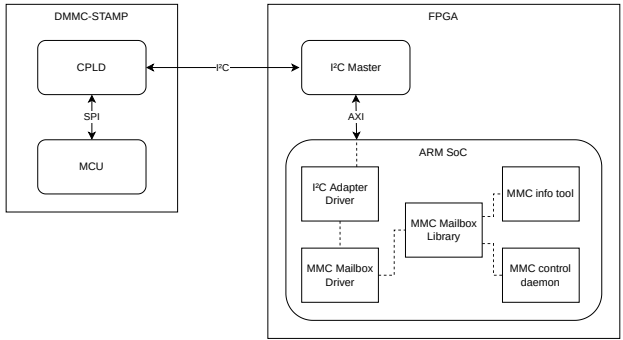
Examples: DAMC-FMC2ZUP / -FMC1Z7IO demo image (SoC variant determined from the chip ID); GigEVision (1G or 10G design, depending on a configuration file)



Data interface between MMC and SoC

MMC data interface (mailbox)

- ”EEPROM-style” (`at24`) I2C interface
- Memory map containing MicroTCA management data
  - MMC status, MMC sensors, AMC slot number
  - FRU status & FRU information for AMC, RTM, FMCs, ...



The mailbox function is abstracted in a MMC Mailbox Library which provides a high-level API for applications

MMC data interface examples

Checking the mounted FMC model as part of sanity checks

Full path is: FMC EEPROM → MMC FRU parser → MMC mailbox → `libmmcmb` → software application

```
root@ZUP-0555 ~ head -n34 /var/log/gigev-ctrl.log |cut -d' ' -f1-2,4-
Starting gigev-ctrl for variant '10G'
[2022-12-05 16:02:36.917519] [info]    MicroTCA Tech Lab GigE Vision controller
[2022-12-05 16:02:36.917607] [info]    Software version: 1.2.1-33-950336c4
[2022-12-05 16:02:36.917631] [info]    Using boost version: 1.71.0
[2022-12-05 16:02:36.917655] [debug]   Log: debugging level enabled
[2022-12-05 16:02:36.917696] [info]    HwAccessorAarch64: /dev/mem opened for 0xa0000000
[2022-12-05 16:02:36.917744] [debug]   HwAccessorAarch64: Memory mapped at address 0xffff80a71000
[2022-12-05 16:02:36.917782] [info]    HwAccessorAarch64: /dev/mem opened for 0xaa000000
[2022-12-05 16:02:36.917816] [debug]   HwAccessorAarch64: Memory mapped at address 0xffff80671000
[2022-12-05 16:02:36.917838] [info]    Machine: DAMC-FMC2ZUP 1 channel, 10 GBit
[2022-12-05 16:02:36.917859] [info]    Machine: Firmware/Software identifiers are matching!
[2022-12-05 16:02:36.917892] [info]    Machine: got unique information:
[2022-12-05 16:02:36.917909] [info]      Implementation:   DIPC-7050 GigE Vision Stack
[2022-12-05 16:02:36.917925] [info]      Manufacturer:     MicroTCA Technology Lab, 2019
[2022-12-05 16:02:36.917942] [info]      IP Version:       Revision 01.01.00
[2022-12-05 16:02:36.917958] [info]    Machine: Firmware/Software version are matching!
[2022-12-05 16:02:36.948935] [info]    FMC1 is a CAENELS FMC-4SFP+
[2022-12-05 16:02:36.948997] [info]    DIP Switch Status: 0x03
[2022-12-05 16:02:36.949652] [info]    Fmc4SfpConfig: Set PLL to 156250000 Hz
[2022-12-05 16:02:36.962469] [info]    ClockMonitor: FMC1 GBT CLK0 = 100.009 MHz
[2022-12-05 16:02:36.962519] [info]    ClockMonitor: GIGEV PCS_PMA = 0 MHz
[2022-12-05 16:02:36.962538] [warning] ClockMonitor: FMC1 / PCS_PMA clock mismatch or zero
[2022-12-05 16:02:41.963824] [info]    ClockMonitor: FMC1 GBT CLK0 = 156.265 MHz
[2022-12-05 16:02:41.963873] [info]    ClockMonitor: GIGEV PCS_PMA = 156.264 MHz
[2022-12-05 16:02:41.963892] [info]    ClockMonitor: PCS_PMA OK
[2022-12-05 16:02:41.964633] [info]    Starting machine: DAMC-FMC2ZUP 1 channel, 10 GBit
[2022-12-05 16:02:41.990879] [info]    SFPDiagnostic: SFP transceiver module detect on channel: 0
[2022-12-05 16:02:41.990906] [info]                   Vendor: FS
[2022-12-05 16:02:41.990926] [info]                   Product: SFP-10G-T
[2022-12-05 16:02:41.990943] [info]                   Rev.:
[2022-12-05 16:02:41.990961] [info]                   S/N: G1904009153
[2022-12-05 16:02:41.990978] [info]                   Date: 190507
[2022-12-05 16:02:41.991003] [info]                   Identifier: SFP transceiver
[2022-12-05 16:02:41.991035] [info]                   Connector: LC
```

Reading MMC sensors from a shell

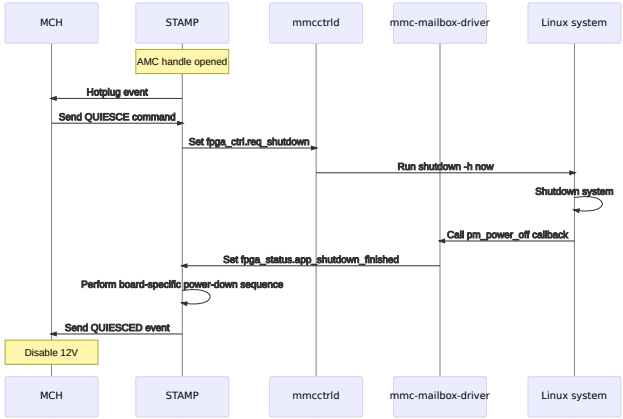
`mmcinfo` is the command line tool for the MMC mailbox

```
root@ZUP-0555 ~ mmcinfo sensors
MMC sensors
-----------
STAMP Temp    : 34.3125
AMC MP 3V3    : 3.38565
AMC PP 12V    : 12.4858
I_RTM MP 3V3  : 0
I_RTM PP 12V  : 0
CPLD Done     : 1
RTM MP 3V3 P  : 0
RTM PP 12V P  : 0
RTM Fault     : 0
RTM Temp.1    : nan
PGood_A       : 1
PGood_B       : 1
FPGA1 Init    : 1
FPGA1 Done    : 1
FPGA2 Init    : 1
FPGA2 Done    : 1
Inlet Temp    : 38.3125
Outlet Temp   : 38.8125
LTM4630 Temp  : 39.8741
LTM4650 Temp  : 41.4658
LTM4633_F Te  : 44.8637
LTM4633_R Te  : 46.1143
ZUP IC Temp   : 47
S7 IC Temp    : 44
IMON_AVTT     : 0.580708
IMON_AVTTY    : 0.473748
IMON_AVCC     : 0.498168
IMON_AVCCY    : 0.263492
Vcore         : 0.720093
VCC_Vadj      : 1.80127
VCC_1V2       : 1.19946
FMC-4SFP+ PG  : 1
```

Shutdown of a Linux system on a AMC

A Linux system should be properly shut down before being switched off, to prevent filesystem corruption.

- When handle is opened, `mmcctrld` daemon triggers system shutdown
- MMC waits for ”shutdown finished” signaling from Linux kernel `pm_power_off()`



Support for versioning of Yocto images

Regular Yocto manifest

The sources of a Yocto system are determined by its ”manifest” XML.

While actively working on a Yocto system, it’s practical to have the manifest pointing at development branches.

File: **.repo/manifests/techlab-internal.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>

    <!-- Common repositories -->
    <include name="xilinx.xml"/>

    <!-- Special repositories -->
    <remote name="msktechvcs"  fetch="ssh://git@msktechvcs.desy.de/techlab/software/yocto-layers/"/>

    <project remote="msktechvcs" revision="rel-v2020.2" name="meta-techlab-utils" path="sources/meta-techlab-utils"/>
    <project remote="msktechvcs" revision="rel-v2020.2" name="meta-techlab-bsp"   path="sources/meta-techlab-bsp"/>
    <project remote="msktechvcs" revision="rel-v2020.2" name="meta-techlab-demo"  path="sources/meta-techlab-demo"/>

</manifest>
```

However, as the layers evolve, the resulting Linux image will change.

How can we assign and retrieve versions of a Yocto image?

Pinned Yocto manifest

`repo` has a special mode to create a ”pinned manifest”: `repo manifest -r -o pinned.xml`

File: **.repo/manifests/pinned.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
  <remote name="msktechvcs" fetch="ssh://git@msktechvcs.desy.de/techlab/software/yocto-layers/"/>
  <remote name="xilinx" fetch="https://github.com/Xilinx"/>

  <default sync-j="4"/>

  <project name="meta-browser" path="sources/meta-browser" remote="xilinx" revision="ad13d59c9db4b6979f8c05c200a97475eb94d692" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-clang" path="sources/meta-clang" remote="xilinx" revision="81ba160c95b12b2922f99b60bef25ab37a5e2f0e" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-jupyter" path="sources/meta-jupyter" remote="xilinx" revision="eb9ce0967a6d14707c7b33d487e5fae35088ef8e" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-linaro" path="sources/meta-linaro" remote="xilinx" revision="8fa21d2e7d5102f580d65c925e74c5c4481c7291" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-mingw" path="sources/meta-mingw" remote="xilinx" revision="fa7b4ca913d2c9caf618d7355ae78cbf4c8828277" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-openamp" path="sources/meta-openamp" remote="xilinx" revision="3894fa0f3d60ab2ec2debb4fff7e117d5c8ce35ea" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-openembedded" path="sources/meta-openembedded" remote="xilinx" revision="aa649aeeecab1969f168d6c89753f86208a8fdb9" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-petalinux" path="sources/meta-petalinux" remote="xilinx" revision="ef895be94f61e3f315701b0120497dde556136bf" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-qt5" path="sources/meta-qt5" remote="xilinx" revision="432ad2aa6c3a13253fefc909faba368851d21fb1" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-techlab-bsp" path="sources/meta-techlab-bsp" remote="msktechvcs" revision="51a628ed8ccffa48263c6a926869d2d93cdf9144" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-techlab-demo" path="sources/meta-techlab-demo" remote="msktechvcs" revision="581ccf128ba2f71bdd61a10da268d3aa59cbd7e9" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-techlab-utils" path="sources/meta-techlab-utils" remote="msktechvcs" revision="e2669900dccc8259913e3cfe958039eeccc82d52fe" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-virtualization" path="sources/meta-virtualization" remote="xilinx" revision="e5ee8bf32d6ad39d67ec9c02d52ba39148a31b66" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-vitis-ai" path="sources/meta-vitis-ai" remote="xilinx" revision="e24ef77f2363b142c78484c8ac181998f9386c15" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-xilinx" path="sources/meta-xilinx" remote="xilinx" revision="e32c768b7e30c2f949b0b98e807afe82ac213556" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="meta-xilinx-tools" path="sources/meta-xilinx-tools" remote="xilinx" revision="f2de39704de069af39321f5cc2f34bf02870c4f0" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="poky" path="sources/core" remote="xilinx" revision="a453ccc9c7c8ba61436b5fc3745b26688b972f25" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="yocto-manifests" path="sources/manifest" remote="xilinx" revision="dae6456cd29248ff6e01466e6691c259e7d72c9a" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
  <project name="yocto-scripts" path="." remote="xilinx" revision="7c2e51bc4f315b70707b7d504a469e5f10a9808f" upstream="rel-v2020.2" dest-branch="rel-v2020.2"/>
</manifest>
```

A pinned manifest points at explicit commit hashes for all layers, uniquely IDing all contained source code. This can be checked in / tagged and serve as a version reference for a Yocto image.

Version identification on the target system

A custom `image-buildinfo-mod.bbclass` retrieves version tags & commit hashes from the manifest. Example output before & after tagging a new version:

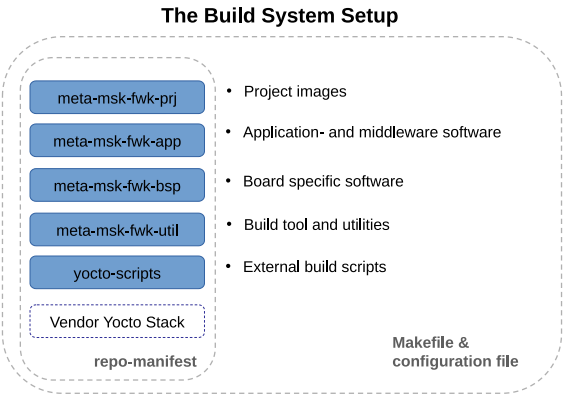
```
root@ZUP-0555 ~ tail -n15 /etc/build
meta-vitis-ai       = HEAD:e24ef77f2363b142c78484c8ac181998f9386c15
meta-techlab-utils  = rel-v2020.2:33ff3e9801a9bd08c0b9391720a80af3da9dbf15
meta-techlab-bsp    = rel-v2020.2:a54d647dddd00b6e86fcc7aecaf2bd7a1beda805
meta-gigevision     = rel-v2020.2:c22e236456271a42189ccf1ff1836878f6418312
workspace           = HEAD:7c2e51bc4f315b70707b7d504a469e5f10a9808f

************************
* Image Version:       *
************************
Manifest: v0.01
Layers: clean
meta-techlab-utils: mismatch: 33ff3e9801a9bd08c0b9391720a80af3da9dbf15, pinned e2669900dcc8259913e3cfe958039eecc82d52fe
meta-techlab-bsp: mismatch: a54d647dddd00b6e86fcc7aecaf2bd7a1beda805, pinned 42a167598c92cc8946c56c59f1645267486e3600

Image version: N/A
```

```
root@ZUP-0555 ~ tail -n15 /etc/build
meta-openamp        = HEAD:3894fa0f3d0ab2ec2debb4fff7e117d5c8ce35ea
meta-jupyter        = HEAD:eb9ce0967a6d14707c7b33d487e5fae35088ef8e
meta-vitis-ai       = HEAD:e24ef77f2363b142c78484c8ac181998f9386c15
meta-techlab-utils  = rel-v2020.2:33ff3e9801a9bd08c0b9391720a80af3da9dbf15
meta-techlab-bsp    = rel-v2020.2:a54d647dddd00b6e86fcc7aecaf2bd7a1beda805
meta-gigevision     = rel-v2020.2:c22e236456271a42189ccf1ff1836878f6418312
workspace           = HEAD:7c2e51bc4f315b70707b7d504a469e5f10a9808f

************************
* Image Version:       *
************************
Manifest: v0.02
Layers: clean

Image version: v0.02
```

Integration into MSK Firmware Framework

Integration into MSK Firmware Framework

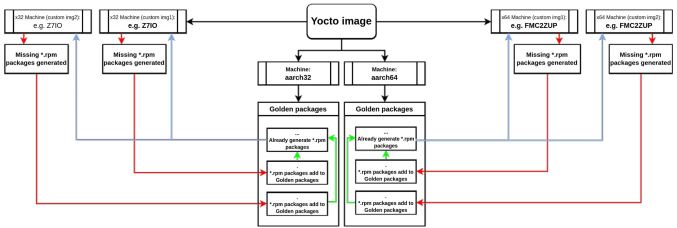
- FWK is a framework used in DESY for FPGA firmware development.
- FWK is based on TCL script and Makefile with config files for FWK projects



"Integration of Yocto and Jupyter Lab into the MSK Firmware Framework for MPSoCs", Seyed Nima Omidsajedi & Michael Randall (DESY, MSK)

Yocto Package Feed

- Yocto projects are typically created using a prebuilt shared state cache plus a number of packages for application development.
- An option for updating Yocto is copying build binaries manually to the target's root filesystem. However, this method is not useful if packages need to be updated regularly. Therefore, we configured an Apache server as a package feed so the target's package management system can install packages directly on the board.

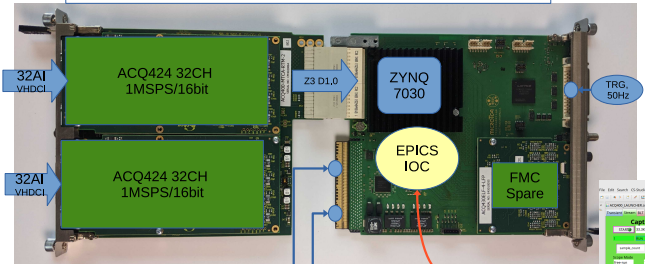


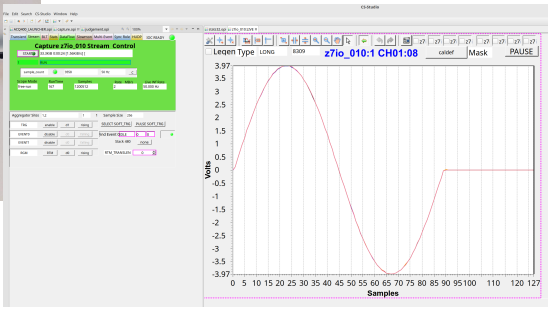
"Integration of Yocto and Jupyter Lab into the MSK Firmware Framework for MPSoCs", Seyed Nima Omidsajedi & Michael Randall (DESY, MSK)

Application examples

DAMC-FMC1Z7IO: 64 channel scope (D-TACQ)

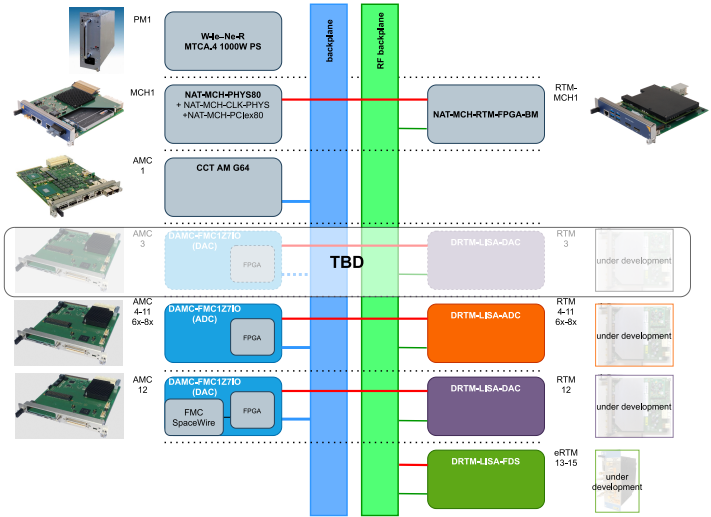
64 Channel Scope, 50Hz trigger, Stats and Mask





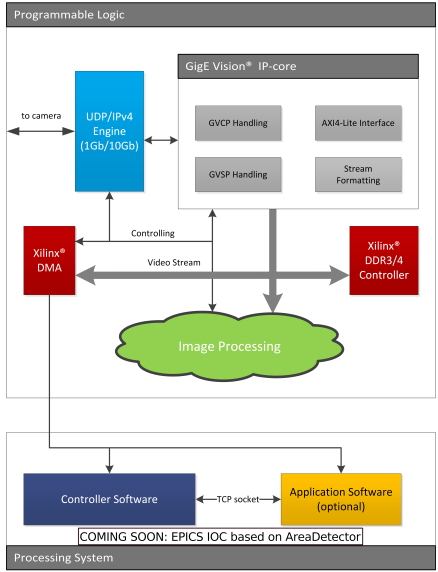
Source: D-TACQ Solutions Ltd

DAMC-FMC1Z7IO: LISA phasemeter (Uni HH)



Source: Universität Hamburg

DAMC-FMC2ZUP: GigEVision (MicroTCA Technology Lab)



Thank you!

Links:

- Support for multiple PL designs per image
  https://github.com/MicroTCA-Tech-Lab/meta-techlab-bsp/tree/rel-v2020.2/recipes-bsp/hdf
  https://github.com/MicroTCA-Tech-Lab/meta-techlab-utils/tree/rel-v2020.2/recipes-bsp/device-tree-from-bd
- MMC-SoC data interface, Linux shutdown through handle pull
  https://github.com/MicroTCA-Tech-Lab/mmc-mailbox
- Yocto image versioning
  https://github.com/MicroTCA-Tech-Lab/meta-techlab-utils/blob/rel-v2020.2/classes/image-buildinfo-mod.bbclass
  https://github.com/MicroTCA-Tech-Lab/meta-techlab-utils/blob/rel-v2020.2/scripts/tag-image.sh