

fwk Tutorial

14th MicroTCA Workshop

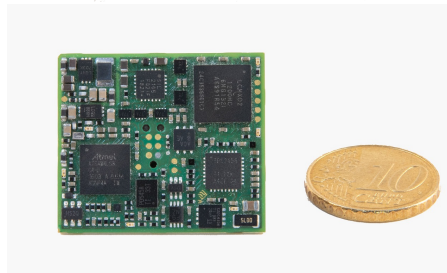
Çağıl Gümüş

Hamburg, 02.12.2025

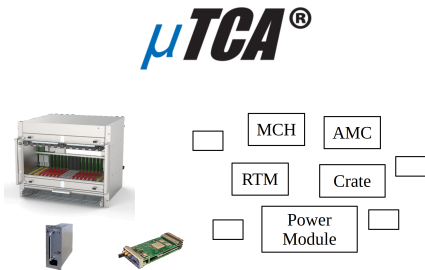
What is MicroTCA?



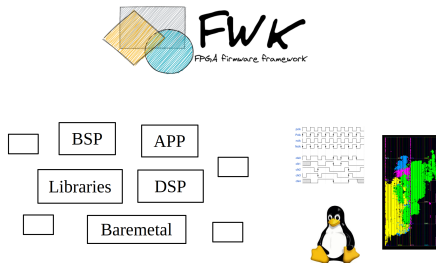
MicroTCA is an abstraction done on hardware.



MicroTCA and fwk are abstractions at a different level



Hardware Abstraction

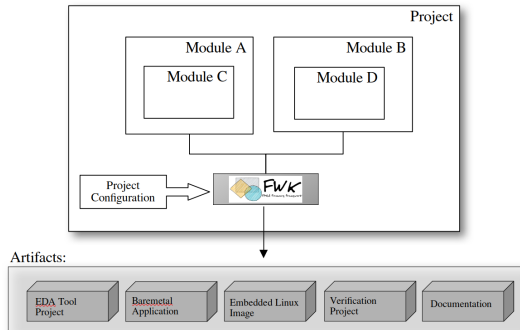


Firmware Abstraction

What is fwk?



What is fwk?



- > It is an **open-source, FPGA Framework**. Written in Tcl-language (>11k lines)
- > Automates the creation and build processes of EDA Tools
- > Divides the codebase into smaller chunks of IPs (Modules)

Supported Tools



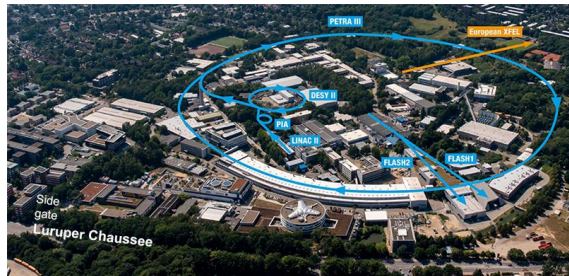
Why does fwk exist?



Why does fwk exist?

fwk started from the MSK Group at DESY.
This group focuses on:

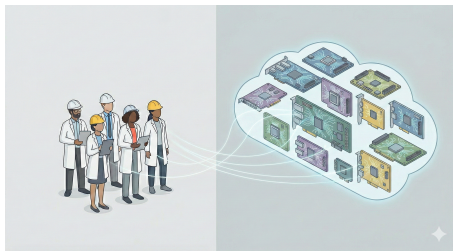
- Research and Development
- Maintenance
- Collaboration: Public/Private Sector



Why does fwk exist?

Resource Problem:

- > # FPGA Bitstreams + Verification: >100
- > # Baremetal Projects: >15
- > # Embedded Linux Images: >7
- > # developers: 8



Generated by Gemini 3

Solution: Increase **Reusability** with **Abstraction**

Where is fwk used?



Where is fwk used?

Biggest fwk project: European-XFEL, Hamburg, Germany

- > Low-Level-RF (LLRF) Systems
- > Beam Diagnostics
- > Timing and Synchronization
- > DAQ
- > Motion Control



View of the E-XFEL Linac

fwk is used in:



sck cen



ICEYE



How does fwk work?



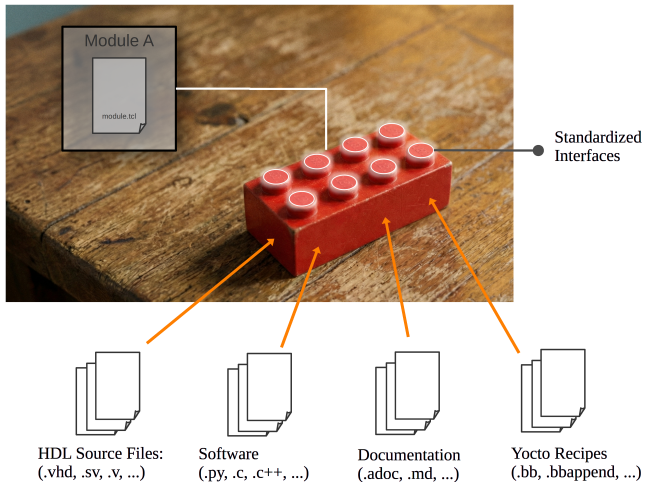
Module



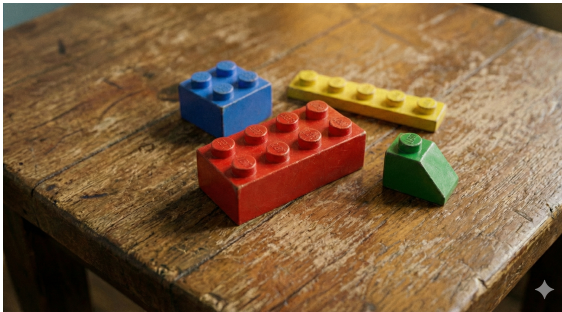
Generated by Gemini 3

Module is a container for an IP. It is the building block of fwk.

Module



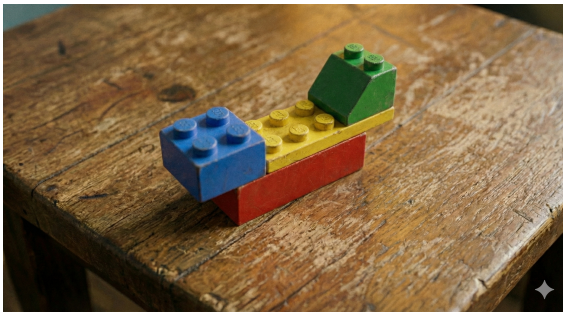
Types of Modules



Generated by Gemini 3

There are many types of modules: BSP, RTM, FMC, APP, etc.

Modules using Modules



Generated by Gemini 3

Modules can encapsulate other modules.

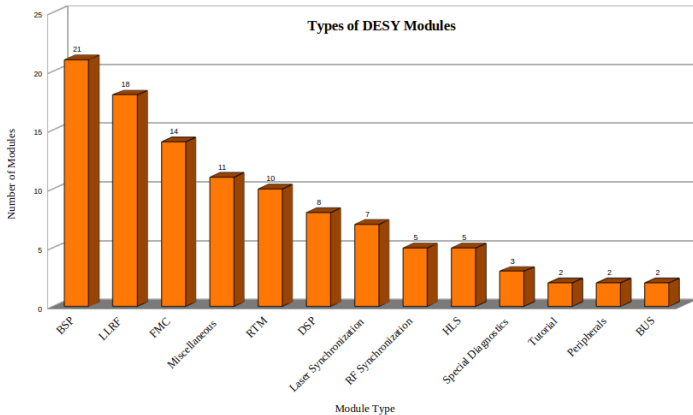
Assembled Project



Generated by Gemini 3

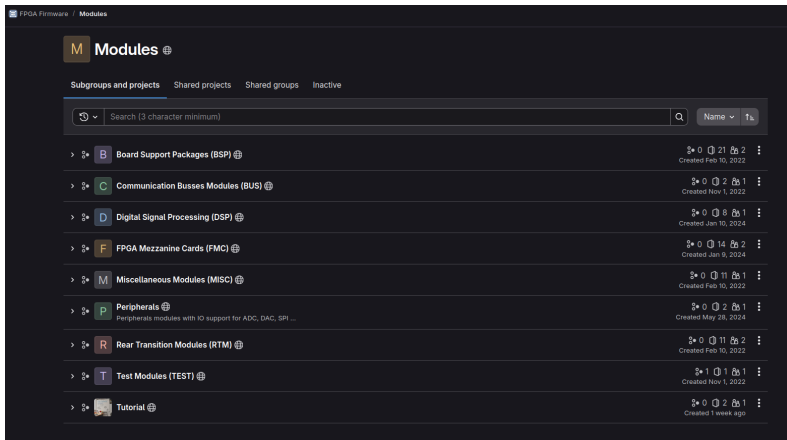
Final assembled project.

DESY Modules



Total in-house Modules: 108 (25 open-sourced under CERN OHWL- W2)

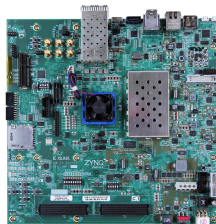
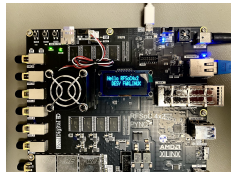
DESY Modules



Visit our Gitlab Page to see all DESY-Modules

fwk on Evaluation Boards

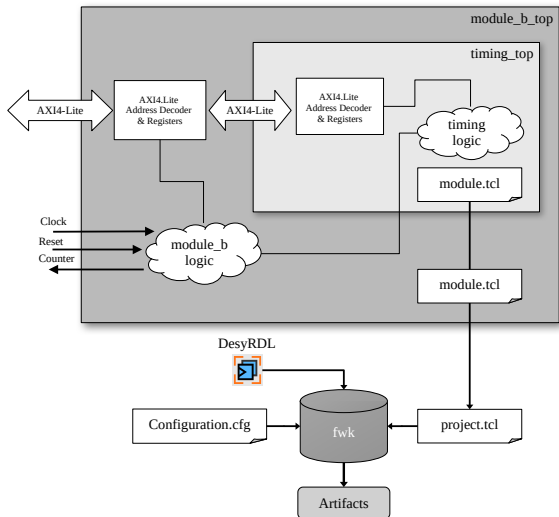
- > fwk can support various evaluation boards from AMD. Available:
- > ZCU208, ZCU102, RFSoc4x2
- > Universidad Politécnica de Madrid has created an fwk example design for Kria260!
- > GitHub Link: I2A2 Instrumentation R&D Group



Tools: DesyRDL



- > Modules are often uses lots of registers.
- > One module address space can be inside another module
- > Use meta-programming (Jinja2) to generate boilerplate VHDL code for address decoder and register logic



Future



Milestones & Future

Milestone 1.0.0 Reached:

- > Complete overhaul of the Tcl API
- > Documentation written from scratch (w/ the help of LLMs)
- > Evolved into a community project with >20 contributors.
- > External Facilities and Companies started to join the cause!

Outlook:

- > MSK Group of DESY will continue maintain and develop for many years to come.
- > 2.0.0 Planning has already started...

*Special thanks to **Lukasz Butkowski** for his great contribution to 1.0.0 efforts*

fwk Tutorial



fwk Documentation and Tutorials

FPGA Firmware Documentation (Public)

fwk Documentation

FWK

Home Page

Installation

What is an fwk Project?

What is an fwk Module?

Configuration Mechanism

How fwk works under the hood?

Command Reference (baika)

Tcl API Reference

Address Space

VHDL Library Management

Supported EDA Tools

Test and Verification of a Module

Module Documentation Guide

Template Engine Usage

Tutorials

CHANGELOG

License and Copyrights

Contributions and Development Guide

fwk Documentation / Tcl API Reference

Contents

Main Process Procedures

fwk::createProject

fwk::buildProject

fwk Tcl API Reference

Main Process Procedures

These procedures control the high-level execution flow of the framework.

fwk::createProject

```
proc ::fwk::createProject {}
```

Creates the vendor tool project. This proc executes the following flow:

1. Creates the project build directory (PrjBuildPath).
2. Calls `fwk::tool::cleanProject` to remove old project files.
3. Packages any modules defined in the `::fwk::tclp` namespace (if it exists).
4. Calls `fwk::tool::createProject` to create the empty tool project.
5. Calls `fwk::doOnCreate` (runs `doOnCreate` hook in all modules).
6. Calls `fwk::procTemplatesRec` (processes all templates).
7. Calls `fwk::addr::main` (generates address space files).
8. Calls `fwk::addSourceRec` (runs `addSources` hook in all modules, which in turn calls `fwk::tool::addSources`).
9. Calls `fwk::doPostCreate` (runs `doPostCreate` hook in all modules).
10. Calls `fwk::tool::saveProject` and `fwk::tool::closeProject` (if not in GUI mode).

fwk::buildProject

```
proc ::fwk::buildProject {}
```

Builds the project using the vendor tool:

1. Calls `fwk::tool::openProject`.
2. Creates the out directory.
3. Calls `fwk::addr::main` to regenerate address spaces (ensuring they are up-to-date).
4. Calls `fwk::doOnBuild` (runs `doOnBuild` hook in all modules).
5. Calls `fwk::tool::buildProject` (runs the tool's main synthesis/implementation).
6. Calls `fwk::tool::exportOut` (copies artifacts to the out directory).

FPGA Firmware Documentation (Public)

fwk Documentation

FWK

Home Page

Installation

What is an fwk Project?

What is an fwk Module?

Configuration Mechanism

How fwk works under the hood?

Command Reference (baika)

Tcl API Reference

Address Space

VHDL Library Management

Supported EDA Tools

Test and Verification of a Module

Module Documentation Guide

Template Engine Usage

Tutorials

CHANGELOG

License and Copyrights

Contributions and Development Guide

fwk Documentation / Tutorials / Address Space Tutorial

Contents

Setup and Environment

Inspecting the project.tcl

Inspecting module_ba Tcl script

Inspecting module_ba Register definitions (.rd)

Create and View the Vivado Project

Map Files

Address Space Tutorial

Modules can instantiate sub-modules while mapping the address space of these instances into their own. This tutorial demonstrates how to achieve this hierarchical address mapping using DesyRTL within fwk projects.

Before starting this tutorial, it is recommended to review the [Main Documentation of DesyRTL](#).

In short, DesyRTL generates the compiled address map along with synthesizable HDL files that handle address decoding. This offers a significantly better approach to creating registers, as users only need to define them using an industry-standard language (SystemRTL). fwk ensures that the files generated by DesyRTL are added to the project correctly (e.g., ensuring libraries are set correctly).

Overview of the Project

The diagram illustrates the project structure and workflow. It shows a hierarchy of modules: `module_ba_top` (containing `module_ba` and `module_ba_top`), `module_ba` (containing `module_ba` and `module_ba`), and `module_ba` (containing `module_ba` and `module_ba`). The workflow involves configuration, compilation, and artifact generation.

[fig tutorial-address-space] shows how the entire workspace looks like from the top view.

In this tutorial, we will use a module named `module_ba`, which contains a dependent module named `Timing`.

The `Timing` module originates from Desy and is actively used across many projects. For more information, please refer to the [Timing Module Documentation](#).

Check out the brand new documentation and tutorials for 1.0 release!

Thank you!

Contact

Deutsches Elektronen-
Synchrotron DESY

www.desy.de

Çağıl Gümüş
 0000-0003-2176-0212
MSK
cagil.guemues@desy.de
+49-40-8998-3760
<https://cagil-gumus.github.io/>