

fwk: An Open-Source, Modular Framework for FPGA Project

Gianmarco Ricci Çağıl Gümüş
Huairou, Beijing, China, 15.09.2026

HELMHOLTZ



Outline

Introduction

How does fwk work?

Tutorial

Conclusions

Why does fwk exist?

- > Expected life-time of FPGA projects at DESY: >20 years
- > Constant HW/FW/SW/ Update.
- > "FPGA Framework" was born out of **necessity**.



View of the E-XFEL LINAC

fwk's Goal: Increase **Collaboration** among Scientific institutes

fwk is used in:



What is fwk?

- > fwk [1] it's an **open-source, FPGA Framework**.
- > Automates the creation and build processes of electronic design automation (EDA) tools.
- > **Modules** are encapsulated, shareable pieces of code.

What is fwk?

- > fwk [1] it's an **open-source, FPGA Framework**.
- > Automates the creation and build processes of electronic design automation (EDA) tools.
- > **Modules** are encapsulated, shareable pieces of code.

fwk integrates a set of tools:



Figure: Victorinox Swiss Army Knife “Classic”, adapted from [2].

Supported Tools



Outline

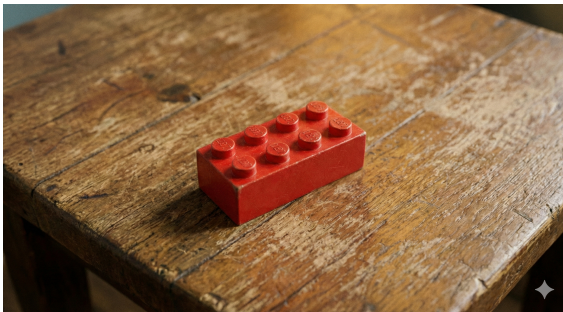
Introduction

How does fwk work?

Tutorial

Conclusions

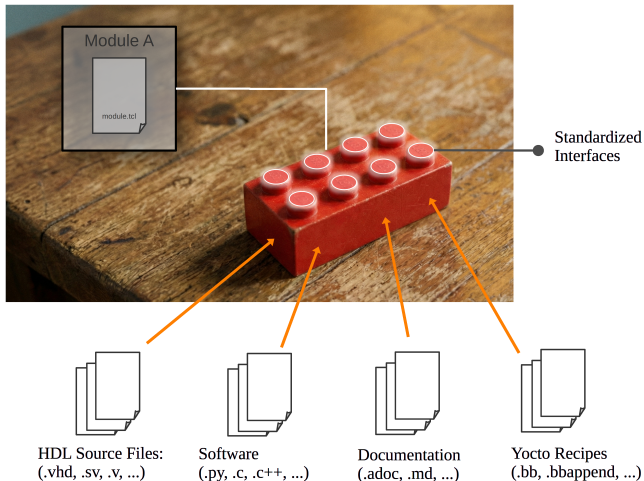
The building block: the "module"

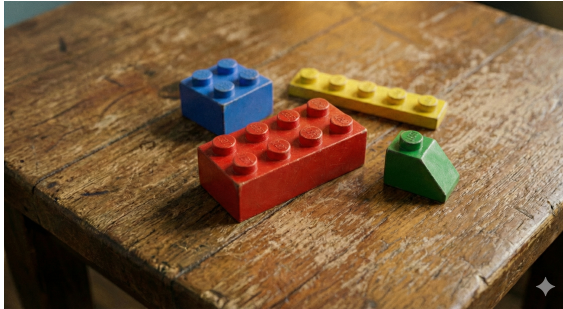


Generated by Gemini 3

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

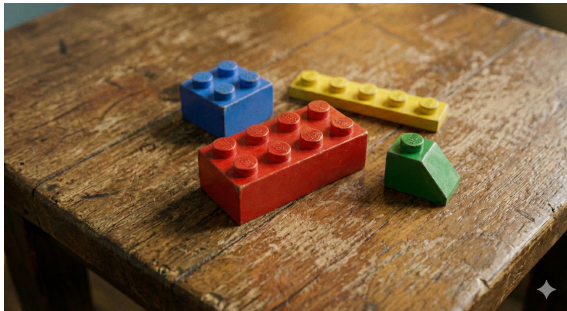
The building block: the "module"





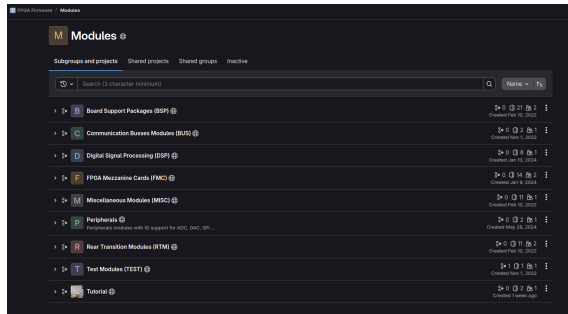
Generated by Gemini 3

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



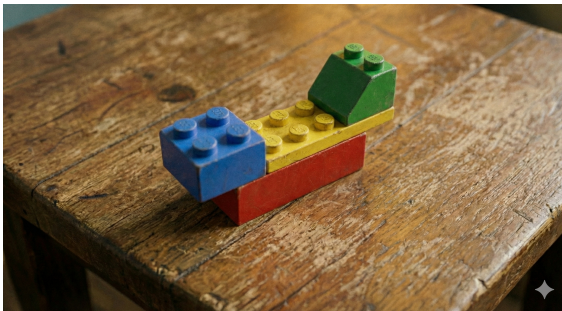
Generated by Gemini 3

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



Visit gitlab.desy.de/fpgafw/mod to see all DESY-Modules.

Many modules make a "project"



Generated by Gemini 3

fwk assembles these modules together

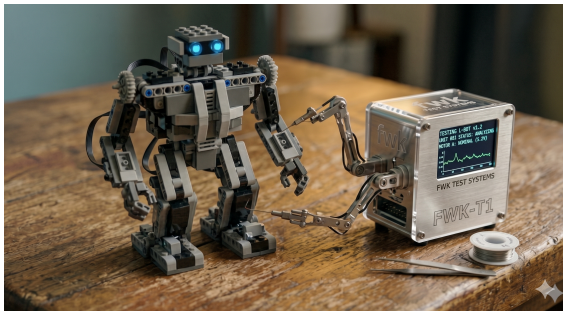
Build of a project



Generated by Gemini 3

And finally, builds it.

Test of a project



Generated by Gemini 3

It can even perform test/simulation on the assembled project.

Outline

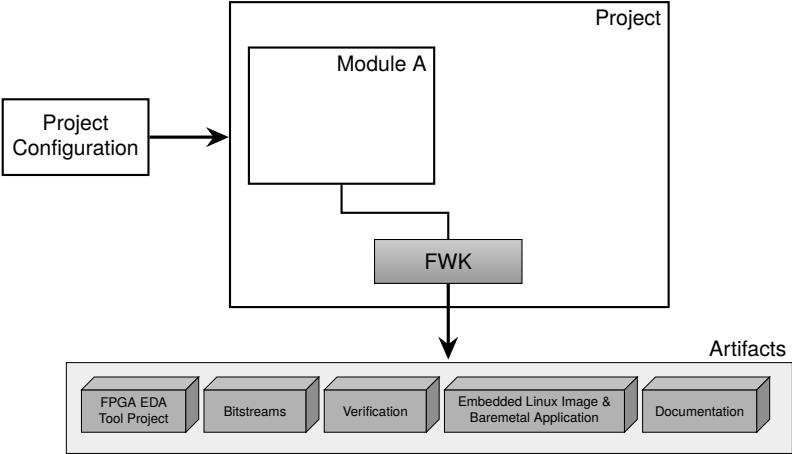
Introduction

How does fwk work?

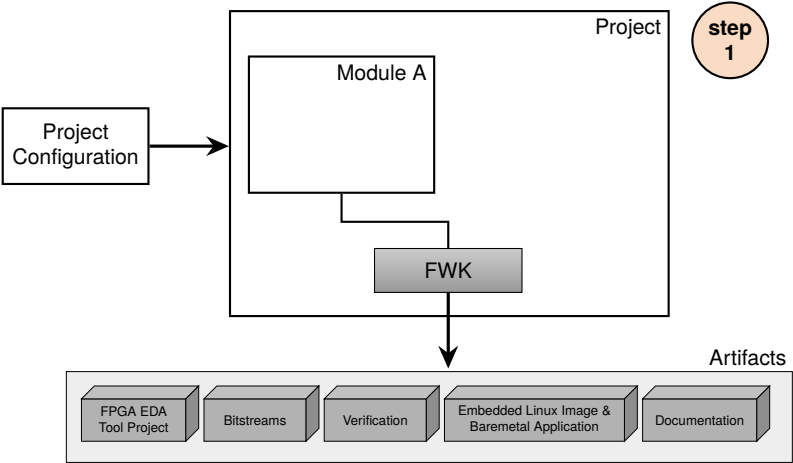
Tutorial

Conclusions

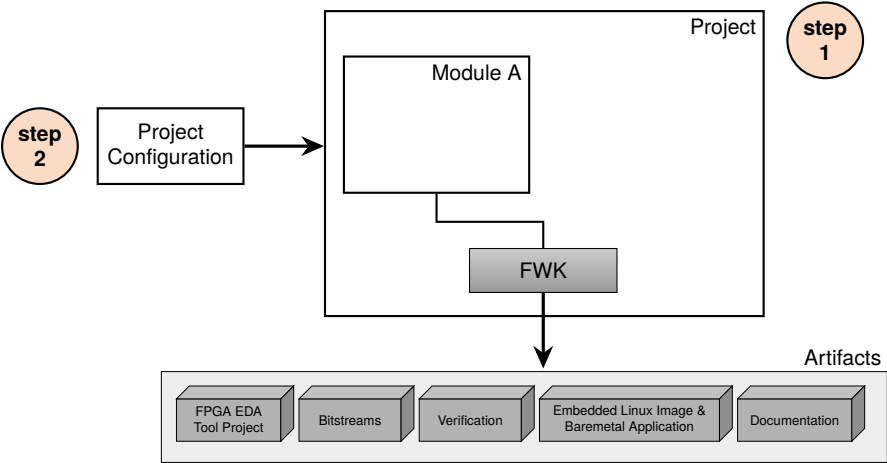
Example of a full fwk project



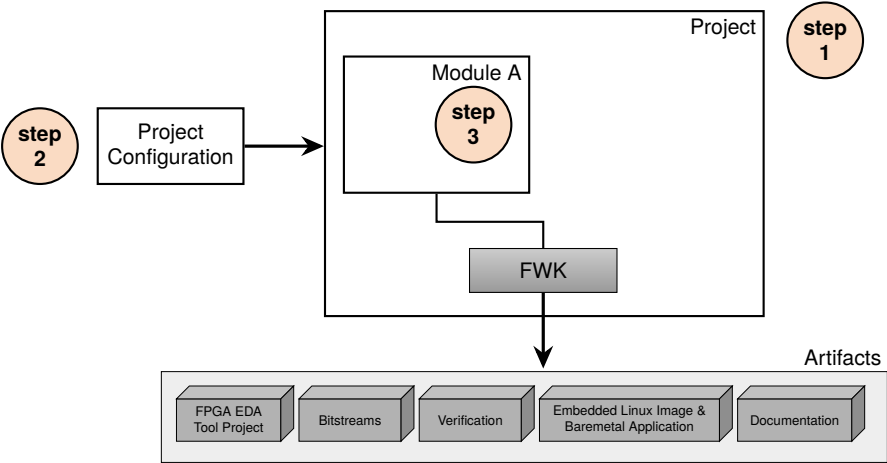
Example of a full fwk project



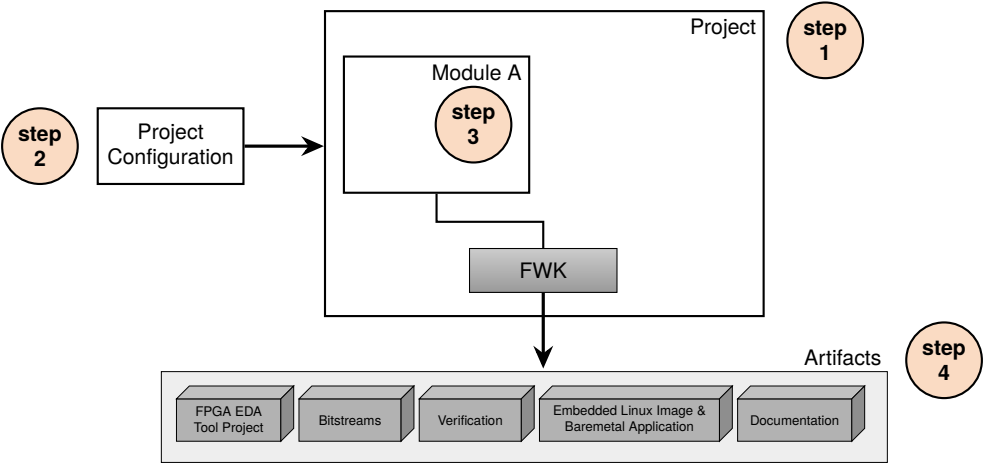
Example of a full fwk project



Example of a full fwk project



Example of a full fwk project



Step 1 - Environment Setup

```
$ mkdir tutorial
$ cd tutorial

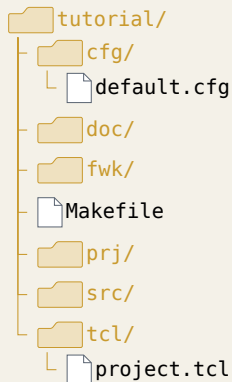
# Clone the framework
$ git clone \
  https://gitlab.desy.de/fpgafw/fwkw.git

# Init framework
$ cd fwkw/
$ git submodule update --init --recursive
$ cd ..
```



Step 1 - Project initialization

```
$ ./fwk/scr/init_fwprj.sh tutorial
$ make env # Install the necessary packages
```



Step 2 - Project configuration

Modify project configuration:

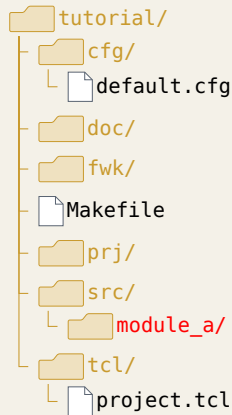
```
./cfg/default.cfg
```

```
1 ProjectName=tutorial
2 ProjectConf=default
3 # Entry point script that builds the module hierarchy.
4 ProjectTcl=tcl/project.tcl
5 # Tool used in this project.
6 ToolType=vivado
```

The cfg/default.cfg file dictates how fwk builds the project.

Step 3 - Adding a module

```
$ git clone \  
https://gitlab.desy.de/fpgafw/mod/\  
tutorial/module_a \  
src/module_a
```



Step 4 - Building the project

```
$ make project gui
```

or

```
$ make project build
```

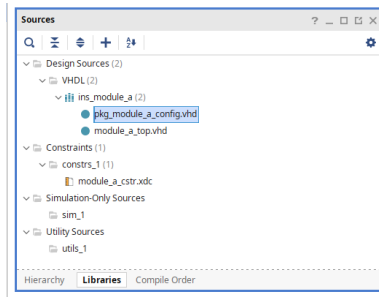


Figure: Vivado window with source files imported from module_a, from [3].

Outline

Introduction

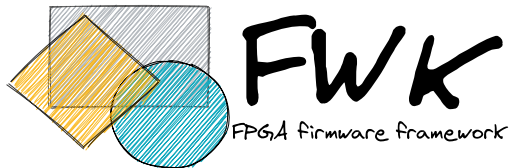
How does fwk work?

Tutorial

Conclusions

Conclusions

- > fwk automates the creation and build processes for FPGA projects.
- > **Main purpose:** standardize the FPGA firmware project structure and build process.
- > Please visit:
 - [fwk documentation](#).
 - [Tutorials](#) with pre-made [projects](#).
 - And many more [open source projects](#).



Thank you!

Contact

Deutsches Elektronen-
Synchrotron DESY

www.desy.de

Gianmarco Ricci

 0009-0004-9894-433X

gianmarco.ricci@desy.de

<https://gitlab.desy.de/gianmarco.ricci/>

Çağıl Gümüş

 0000-0003-2176-0212

cagil.guemues@desy.de

<https://cagil-gumus.github.io/>

References

[1] DESY.

fwk - firmware framework.

<https://gitlab.desy.de/fpgafw/fwk>, 2026.

Accessed: 2026-08-31.

[2] Victorinox.

Victorinox swiss army knife “classic”.

<https://www.victorinox.com/en-MX/Products/Swiss-Army-Knife%E2%84%A2-and-Tools/Essentials/Classic-SD-Classic-Colors/p/0.6223.G>, 2026.

Accessed: 2026-08-31.

[3] Gianmarco Ricci.

fwk basic tutorial.

https://fpgafw.pages.desy.de/docs-pub/fw/main/tutorial_basic.html, 2026.

Accessed: 2026-08-31.

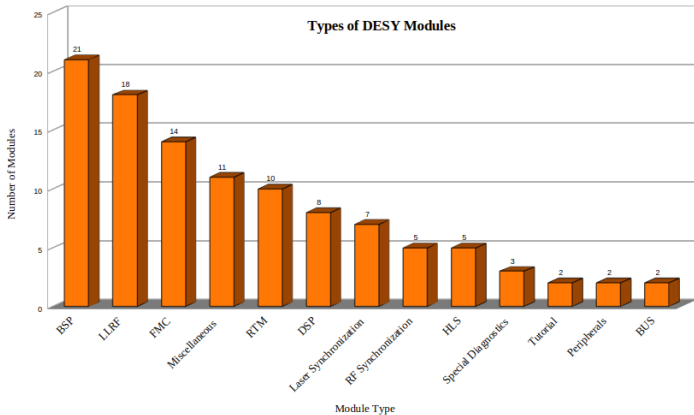
Outline

Backup

What is a Board Support Package (BSP) module in fwk World?

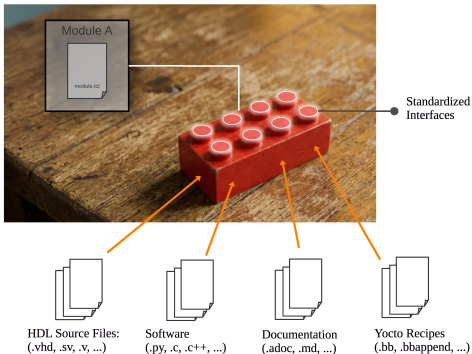
Tools: DesyRDL

DESY Modules



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

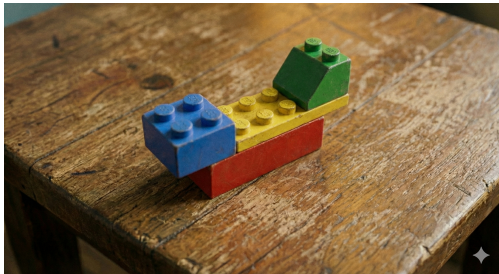
Module



Generated by Gemini 3 edited by a human

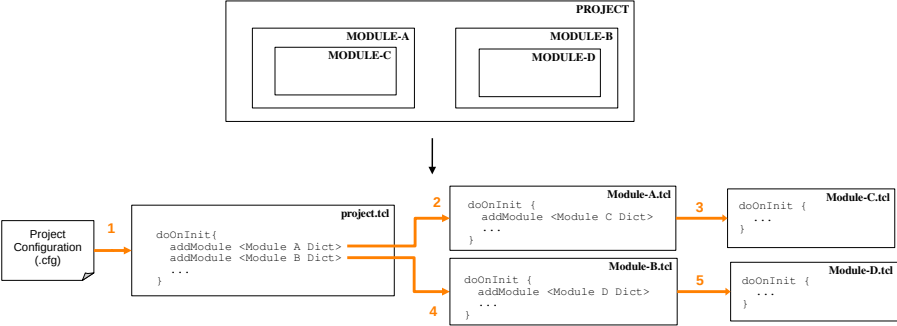
```
my_module/  
|-- cstr/      # Constraints (XDC, PDC, ...)  
|-- doc/       # Documentation (AsciiDoc)  
|-- hdl/       # Synthesizable sources  
|-- ip/        # Vendor IPs (XCI, TCL)  
|-- sim/       # Testbenches, waveforms  
|-- sw/        # Drivers (C, C++, H)  
|-- tcl/       # Fwk integration scripts  
`-- module.tcl
```

Project



```
my_project/
|-- .venv/           # (Python Virtual Environment)
|-- .gitignore/      # (Git Ignore)
|-- .gitmodules/     # (Git Submodules)
|-- cfg/             # (Project Configurations)
|-- prj/             # (Project Build Directory)
|-- doc/             # (Project Documentation)
|-- fwk/             # (fwk)
|-- Makefile/        # (Makefile, fwk entry point)
|-- out/             # (Artifacts, autogenerated)
|-- src/             # (`Sources` directory)
|   |-- hw/          # (Gateway sources go here)
|       |-- my_module/ #
|   |-- sw/          # (Software sources go here)
|-- tcl/             # (Project fwk tcl script)
```


Hierarchial Tree Construction



Step 3 - Integration of the module in the project

./tcl/project.tcl

```
1 proc doOnInit {} {  
2     # Add module_a to this project  
3     addModule [dict create InstanceName ins_module_a File  
4         "path_to/module_a/tcl/module.tcl"]  
5 }  
6 proc doOnCreate {} {  
7     # Set Vivado related project settings  
8     switch $::fwk::ToolType {  
9         vivado {  
10             set_property part xcku040-ffva1156-1-c [current_project]  
11             set_property target_language VHDL [current_project]  
12             set_property default_lib xil_defaultlib [current_project]  
13         }  
14     }  
15 }
```

Module adding another Module

This snippet demonstrates how to initialize a module configuration using Tcl dictionaries.

```
proc doOnInit {} {  
    set module_a_config \  
        [dict create \  
            CFG_PARAMETER1 0x55667788 \  
            CFG_PARAMETER2 "Apple" \  
            CFG_PARAMETER3 {1 2 56} \  
        ]  
  
    addModule [dict create InstanceName "module-a" \  
        File "../src/hw/module-a/tcl/main.tcl" \  
        Config $module_a_config]  
}
```

fwk Module <=> IP Conversation

Handling IP **Consumption** and **Production** within `::fwk::src`

1. Consuming IPs

Integrating existing black boxes.

- > Adds pre-existing, tool-generated IP files (e.g., Xilinx .xci) to the current project fileset.
- > Establishes the IP as a synthesizable black box within your HDL.

Associated API:

```
addGenIPSources {args}
```

2. Producing IPs

Packaging custom RTL as a new IP.

- > Wraps your custom **fwk module** into a reusable vendor IP asset.
- > Automates dependency resolution, address map generation, and packaging.

Associated API:

```
packageIp {ns}  
addFwkIp {InstanceName}
```

fwk Module API Summary

Available Procedures in `::fwk::src::<InstanceName>`

> Configuration & Metadata

- setName, setTop, setConfig, setInfo

> Sources & Structure

- addSource, addTemplate, addModule, addGenIPSources

> Verification & Testing

- addTest, addModuleTests

> Address Space Management

- addAddressSpace, addAddressSpaceToTop

> Code Generation & Versioning

- addConfigAsHdl, parseVhdlConfigFile
- getVersion, genModVerFile

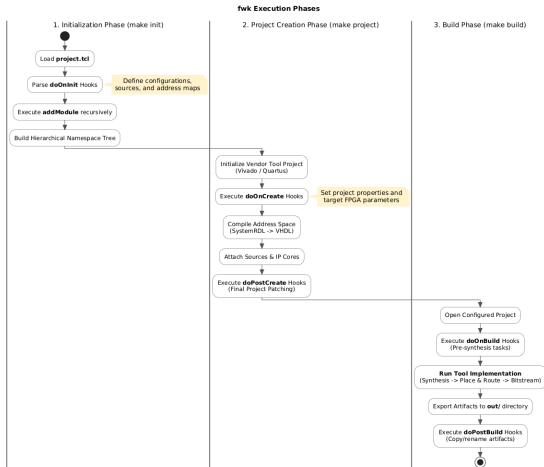
fwk Process Stages

- > **1. Init Phase** (*Top* → *Down*)
 - doOnInit: Define sources and sub-modules.
- > **2. Creation Phase** (*Bottom* → *Up*)
 - doOnCreate: Configure tool settings.
 - doPostCreate: Final project patches.
- > **3. Build Phase** (*Bottom* → *Up*)
 - doOnBuild: Pre-synthesis (e.g., version files).
 - doPostBuild: Post-implementation artifacts.

Module Hook Procedures

Hook	Triggered	Typical Use Case
doOnInit	init	Resource Definition. Register sources (addSource), test cases, and address spaces. Child modules (addModule) are defined here.
doOnCreate	create (Start)	Project Configuration. Set tool properties (e.g., Vivado set_property) or prepare variables before sources are added.
doPostCreate	create (End)	Finalization. Modify project after all sources/IPs are added. Useful for patching generated files.
doOnBuild	build (Start)	Pre-Build. Generate version files, update dynamic content, or perform pre-synthesis checks.
doPostBuild	build (End)	Artifacts. Copy custom artifacts (e.g., .bin) to out/, run reports, or trigger external scripts.
setSim	simulate	Simulation. Configure environments, set simulator properties, and wrap testbenches.

fwk Process Flow



Project Directory Structure

Each project under the framework follows the defined directory structure below.

```
| .venv          (Python Virtual Environment (auto-generated))
| .gitignore     (Git Ignore)
| .gitmodules    (Git Submodules)
| cfg            (Project Configurations)
| tcl            (Project Tcl goes in here)
| prj           (Project Build/Work Directory (auto-generated))
| doc           (Project Documentation)
| fwk           (fwk)
| Jenkinsfile    (Optional: Jenkinsfile for CI/CD)
| Makefile       (Makefile that just includes fwk entry point)
| out           (Build Artifacts (auto-generated))
| src           (Sources directory)
| --- hw        (Gateware sources go here)
| --- sw        (Software sources go here)
```

Outline

Backup

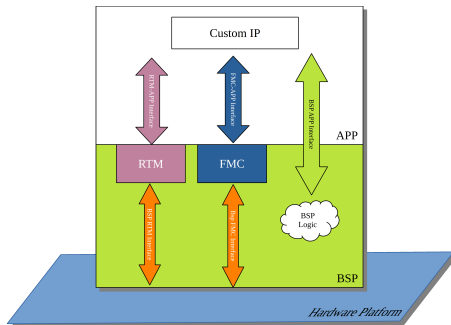
What is a Board Support Package (BSP) module in fwk World?

Tools: DesyRDL

BSPs

What is a Board Support Package (BSP) module in fwk World?

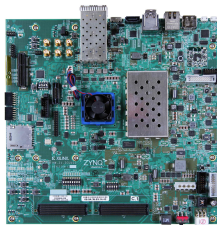
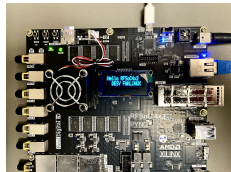
- > BSP Module abstracts the entire underlying hardware
- > Usually carries the PS, PCIe, PL/PS DDR, I2C, SPI etc.
- > Has standardized sockets for FMCs, RTMs
- > The most valuable asset for the fwk ecosystem
- >



BSP Abstracting Underlying Hardware Platform

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



Outline

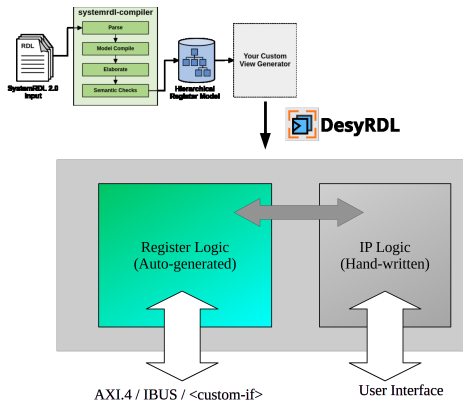
Backup

What is a Board Support Package (BSP) module in fwk World?

Tools: DesyRDL

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



DesyRDL and fwk connection

