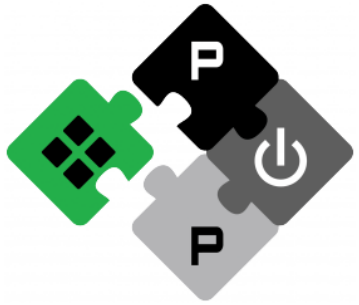


PULP PLATFORM
Open Source Hardware, the way it should be!

A Deep Dive into HW/SW Development with PULP

Robert Balas <balasr@iis.ee.ethz.ch>
Manuel Eggimann <meggimann@iis.ee.ethz.ch>

 http://pulp-platform.org  @pulp_platform  https://www.youtube.com/pulp_platform

Introduction Round

Robert Balas

Manuel Eggimann

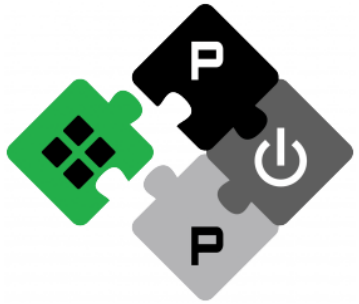
Introduction – Organization of this Training

Day 1

- PULPissimo SoC Architecture
- Software Environment
- RTL Development Flow
- RTL Simulation/Debug Environment

Day 2

- FPGA Port
- PULP IP Landscape
- Hands-on Full-stack IP Integration Exercise
- PULPIssimo Memory Layout Modification



PULP PLATFORM

Open Source Hardware, the way it should be!

SystemVerilog Atrocities

Never Import Into Scope of Compilation Unit

```
import pkg_some_other_ip::*;

module top
  #(
    parameter OUT_WIDTH
  )(
    input logic [IP_BITWIDTH-1:0] data_i;
    output logic [OUT_WIDTH-1:0]  data_o ;
    );
endmodule
```

Never Import Into Scope of Compilation Unit

```
module top
  import pkg_some_other_ip::IP_BITWIDTH;
   #(
     parameter OUT_WIDTH
   )(
     input logic [IP_BITWIDTH-1:0] data_i;
     output logic [OUT_WIDTH-1:0]  data_o ;
     );
endmodule
```

Parameters/Constants

```
module my_ip
   #(
     parameter WIDTH = 32,
     parameter BE_WIDTH = WIDTH/8 //Could be changed
   )(
     input logic [WIDTH-1:0]    data_i;
     input logic [BE_WIDTH-1:0] be_i;
     );
 parameter MY_CONSTANT = 42; //I'm not a constant :-(
endmodule
```

X

X

If you want a (derived) constant, use localparam

Paramters/Constants

```
module my_ip
   #(
     parameter WIDTH = 32,
     localparam BE_WIDTH = WIDTH/8 //Cannot be changed
   )(
     input logic [WIDTH-1:0]     data_i;
     input logic [BE_WIDTH-1:0] be_i;
     );
 localparam MY_CONSTANT = 42; //I'm a constant :-)
endmodule
```

Elaboration SystemTasks (supported since SV-2012)

```
module my_ip
  #(
    parameter NR_CORES = 32
    )(
      input logic [WIDTH-1:0]    data_i;
      input logic [BE_WIDTH-1:0] be_i;
      );
endmodule // my_ip
```

Elaboration SystemTasks (supported since SV-2012)

```
module my_ip
  #(
    parameter NR_CORES = 32 //Must be power of 2!
    )(
      input logic [WIDTH-1:0]    data_i;
      input logic [BE_WIDTH-1:0] be_i;
      );
endmodule // my_ip
```

better

Elaboration SystemTasks (supported since SV-2012)

```
module my_ip
  #(
    parameter NR_CORES = 32 //Must be power of 2!
    )(
      input logic [WIDTH-1:0]    data_i;
      input logic [BE_WIDTH-1:0] be_i;
      );
   if (NR_CORES == 0 || (NR_CORES & (NR_CORES-1)) != 0)
     $error("NR_CORES must");
endmodule // my_ip
```

Even better

Includes

```
`include "mac_ms.svh"

module my_ip
   (input logic  clk_i);

endmodule : my_ip
```

Don’t use generic names!

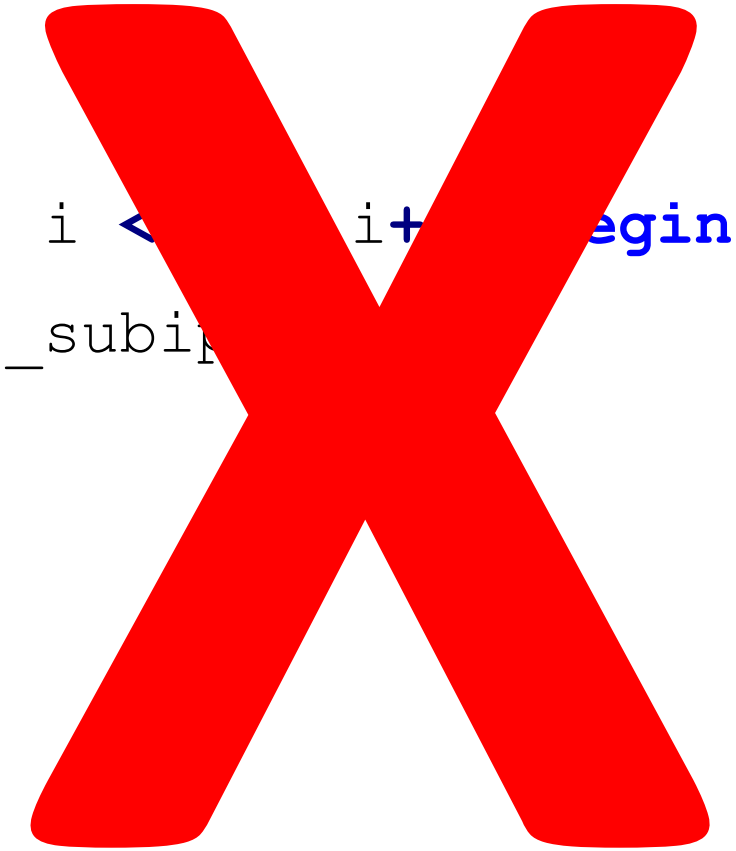
Includes

```
`include "my_ip_macros.svh"
module my_ip
   (input logic  clk_i);
endmodule : my_ip
```

Prefix all header file names and defines with to avoid naming colisions and redefinitions

Generate Statements

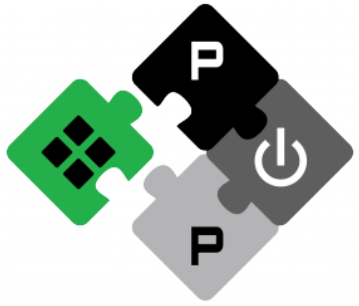
```
genvar i;
generate
  for ( i = 0; i <        i+    egin
    my_subip i_subi
  end
endgenerate
```



Generate Statements

```
for (genvar i = 0; i < 10; i++) begin :gen_sub_ips
  my_subip i_subip…
end
```

- Don’t use generate regions. They are redundant in SystemVerilog (and Verilog 2005).
- Always label your generate blocks. Otherwise the hierarchical name is toole-dependent!

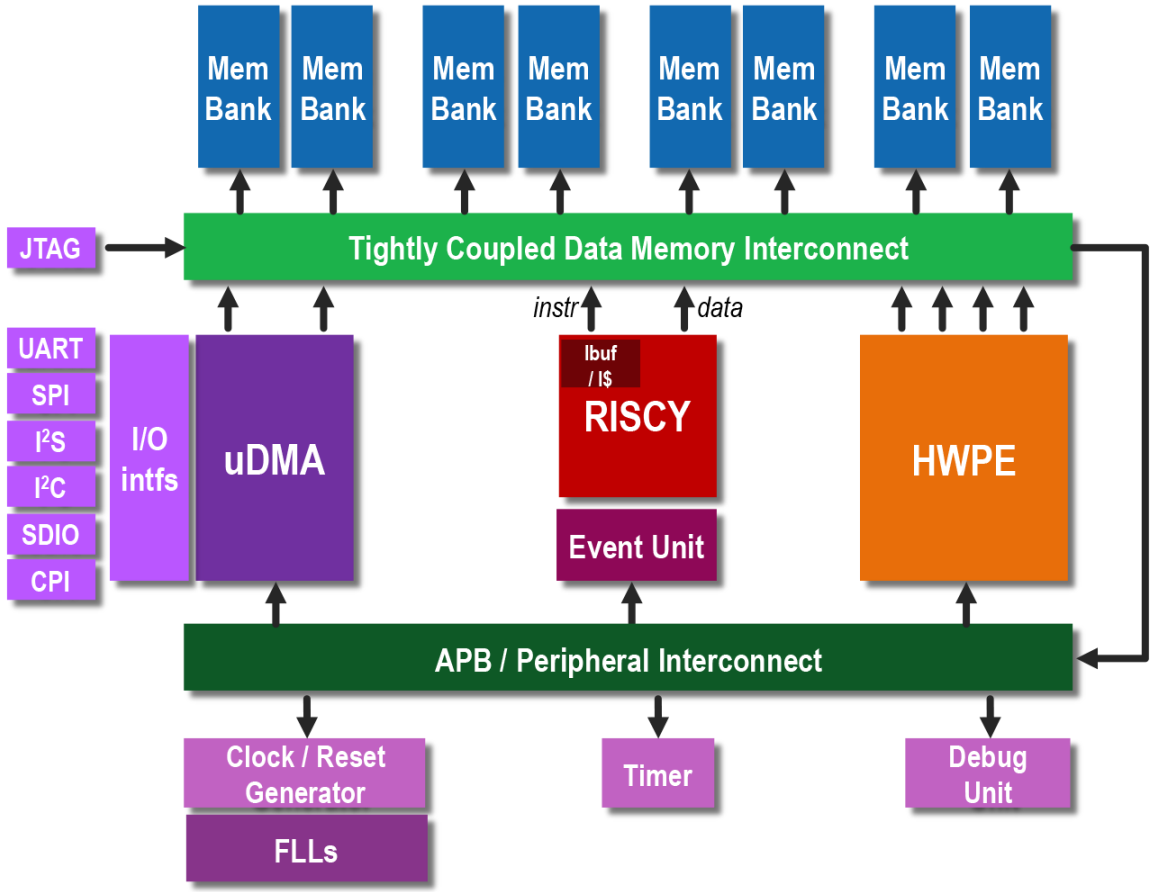


PULP PLATFORM

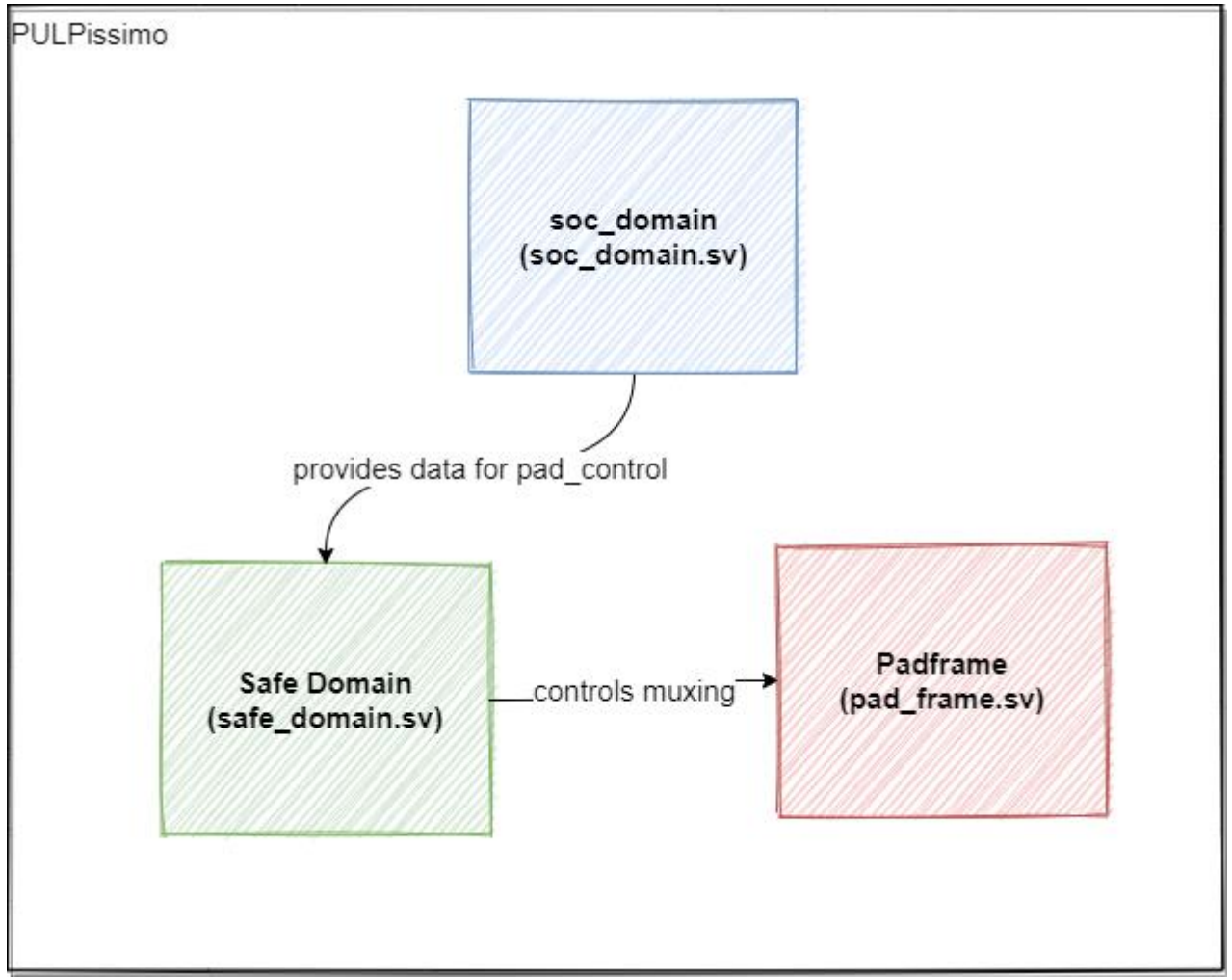
Open Source Hardware, the way it should be!

An Overview on PULPissimo/ PULP SoC

PULPissimo


Mem Bank
Mem Bank
Mem Bank
Mem Bank
Mem Bank
Mem Bank
Mem Bank
Mem Bank
JTAG
Tightly Coupled Data Memory Interconnect
instr
data
UART
SPI
I²S
I²C
SDIO
CPI
I/O intfs
uDMA
Ibuf / I$
RISCY
Event Unit
HWPE
APB / Peripheral Interconnect
Clock / Reset Generator
FLLs
Timer
Debug Unit

PULPissimo


PULPissimo
soc_domain
(soc_domain.sv)
provides data for pad_control
Safe Domain
(safe_domain.sv)
controls muxing
Padframe
(pad_frame.sv)

PULPissimo Special Toplevel Signals

Signal Name	Description
pad_xtal_in	32 kHz input clock (no internal crystal oscillator IP). Connects to internal FLLS (ASIC) or clock managers (FPGA port).
pad_reset_n	Active-low asynchronous reset (internally synchronized)
pad_bootsel	Affects boot behavior according to program in bootrom.
pad_jtag_xxx	Debug Access port for bus access and core debugging (single step, SW breakpoints)

- Be careful about parameters! They are not always supposed to be changed or do not work anymore.
- There is a lot of dead code (remainings from tapeout specific fixes and legacy code).

Padframe

- Contains technology independent wrappers of IO pads
- Signals for each IO pad:

Direction (padframe perspective)	Name	Description
Input	oe_<padname>_i	Active high output driver enable
Output	in_<padname>_o	Logic signal from pad to SoC
Input	out_<padname>_i	Logic signal from SoC to pad
Inout	pad_<padname>	The actual pad signal that is connected to the toplevel
Input	pad_cfg_i	Additional config signals for pad (e.g. pulldown enable)

Safe Domain

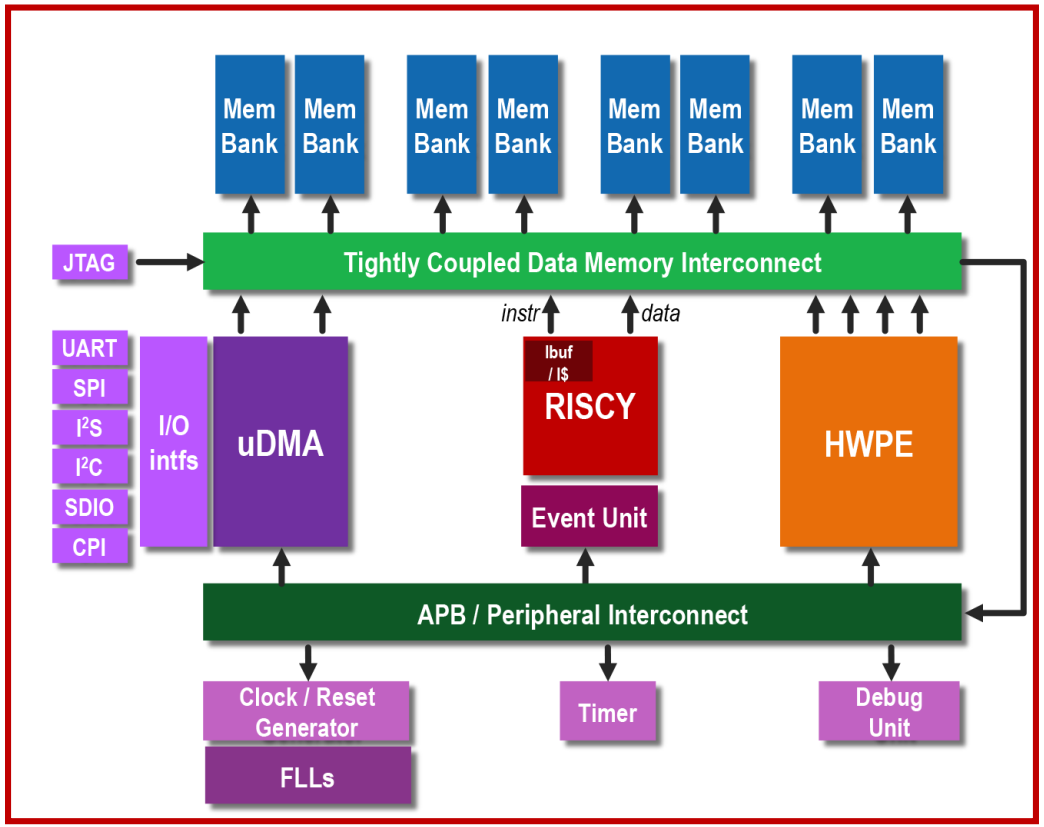
- Contains logic that must not be power gated
- Lives in a separate module for simplified power intent specification in CPF or UPF
- Modules in PULPissimo:
  - pad_control: Multiplexes functionalities of io pads between (e.g. spi sck or gpio)
  - Rst_gen: Synchronizes the reset signal to reference clock. Only used for modules within safe domain that are directly clock with ref_clk.

soc_domain/pulp_soc

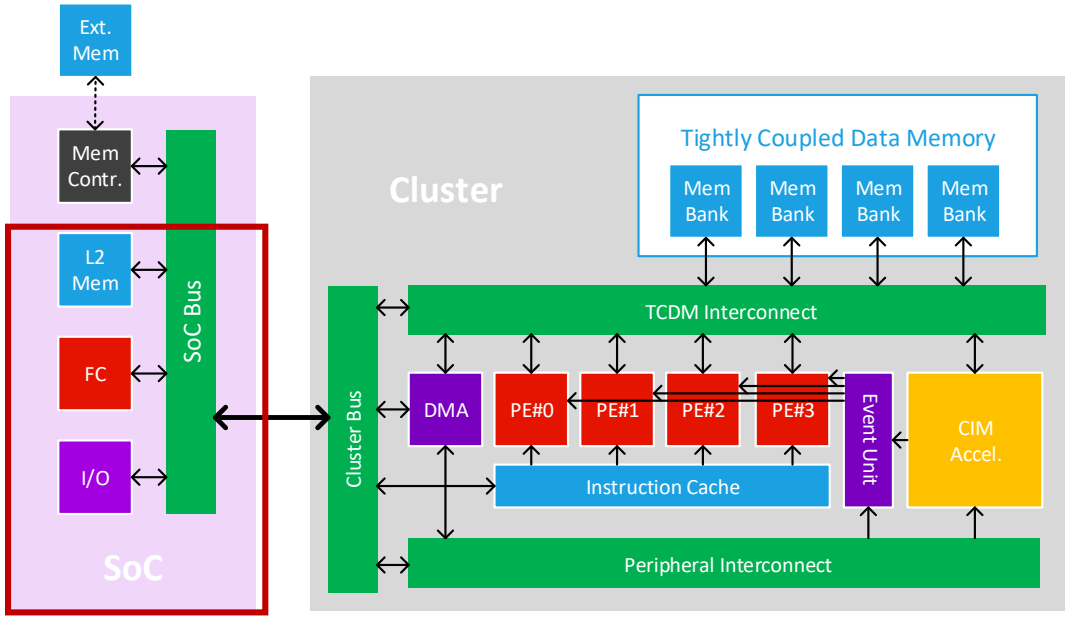
- Wraps the actual heart of the SoC; The pulp_soc IP.
- Pulp_soc was designed to be the main soc fabric of all our larger 32-bit PULP chips
- Contains many signals that are only used when there is an additional multi-core cluster present

soc_domain/pulp_soc

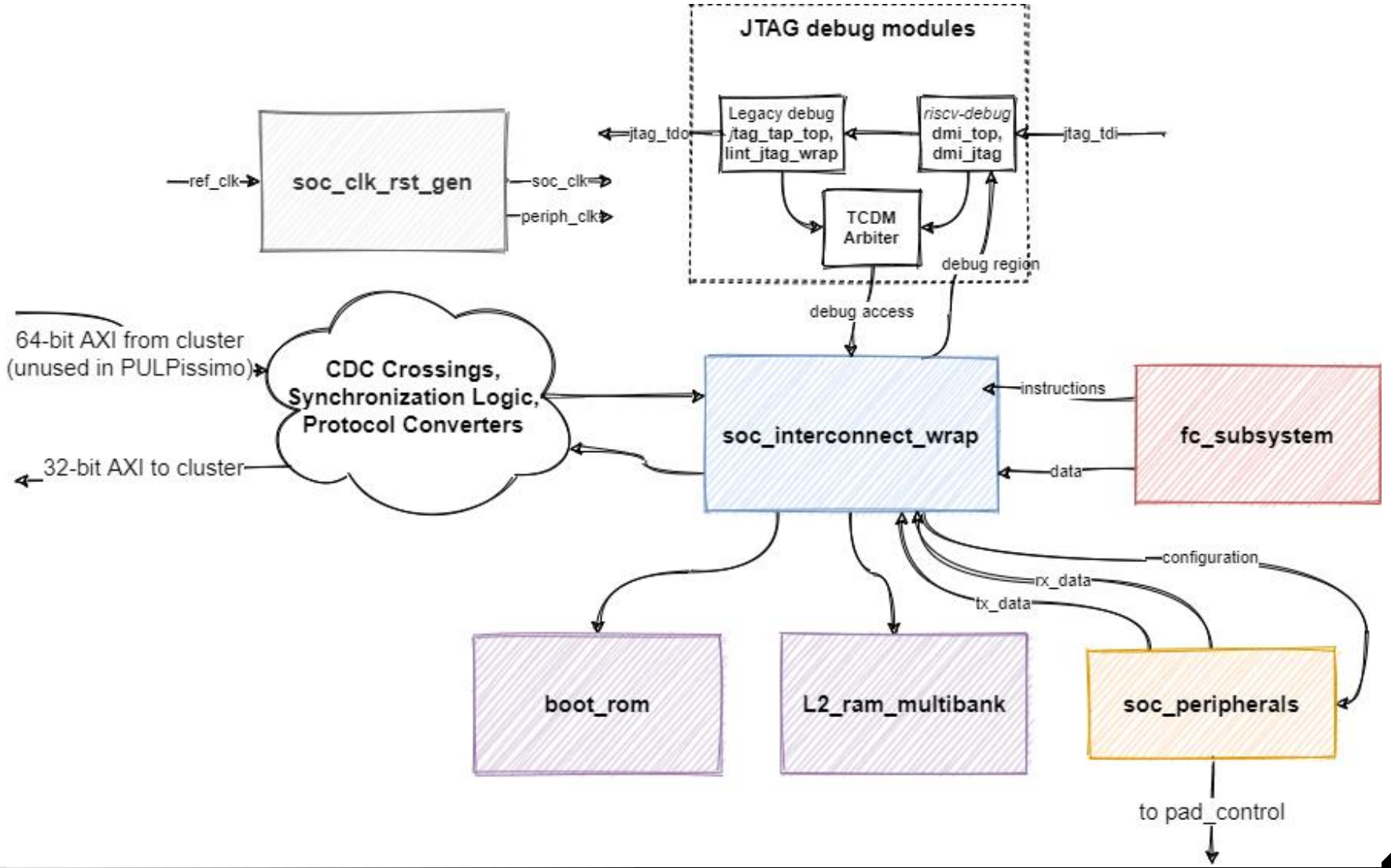
PULPissimo



Multicore PULP



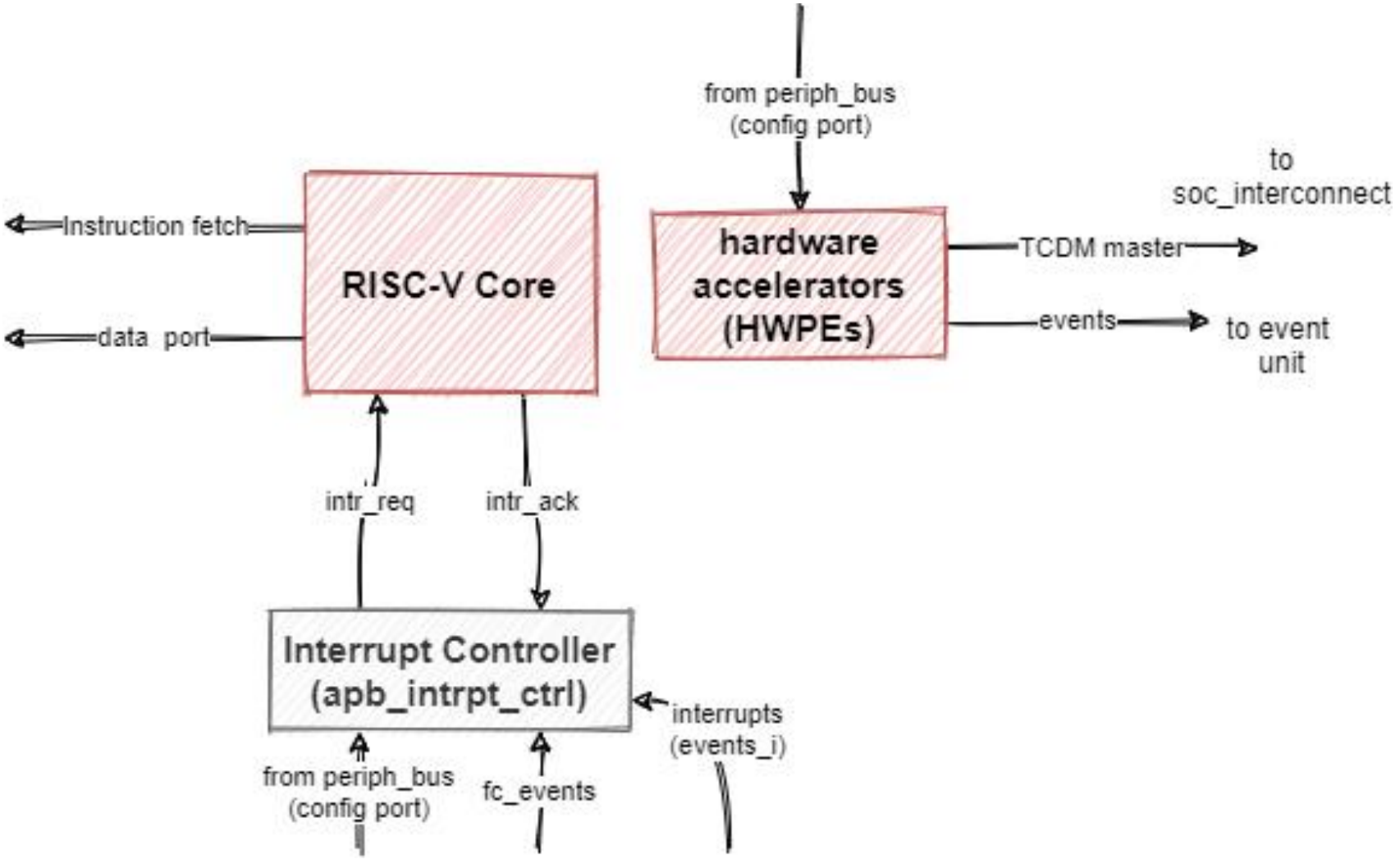
PULP SoC Schematic Overview



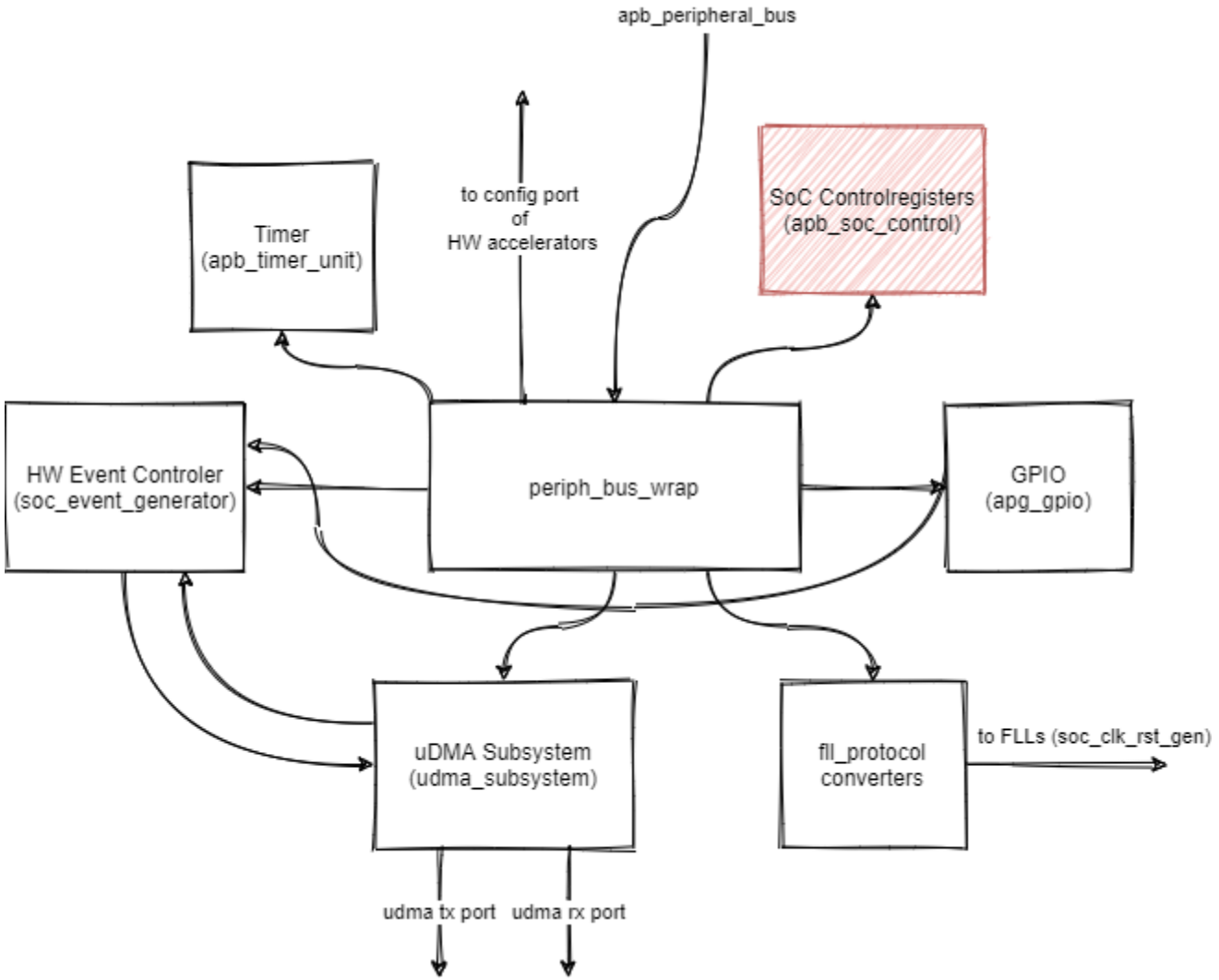
PULPissimo Clock Domains

Clock Name	Description	Usage
ref_clk_i	Signal from directly taken from pad_xtal_in.	Connects to internal FLLs/PLLs
slow_clk_i	In ASIC version, identical to ref_clk_i. For FPGA version, passes through glitch free clock divider since certain boards (e.g. Genesys2) have very fast external oscillators. This one must always be 32kHz	• Timers • Directly used as interrupt source
periph_clk_i	One of the two fast clock. Generated by internal FLL/PLL from ref_clk_i.	• Drives IO facing peripherals (e.g. UART, SPI, I2C)
soc_clk_i	Fastest clock in the system. Generated by second internal FLL/PLL from ref_clk_i.	• Drives everything else (Core, memory, interconnect) in the SoC.

FC Subsystem



SoC Peripherals



APB SoC Control

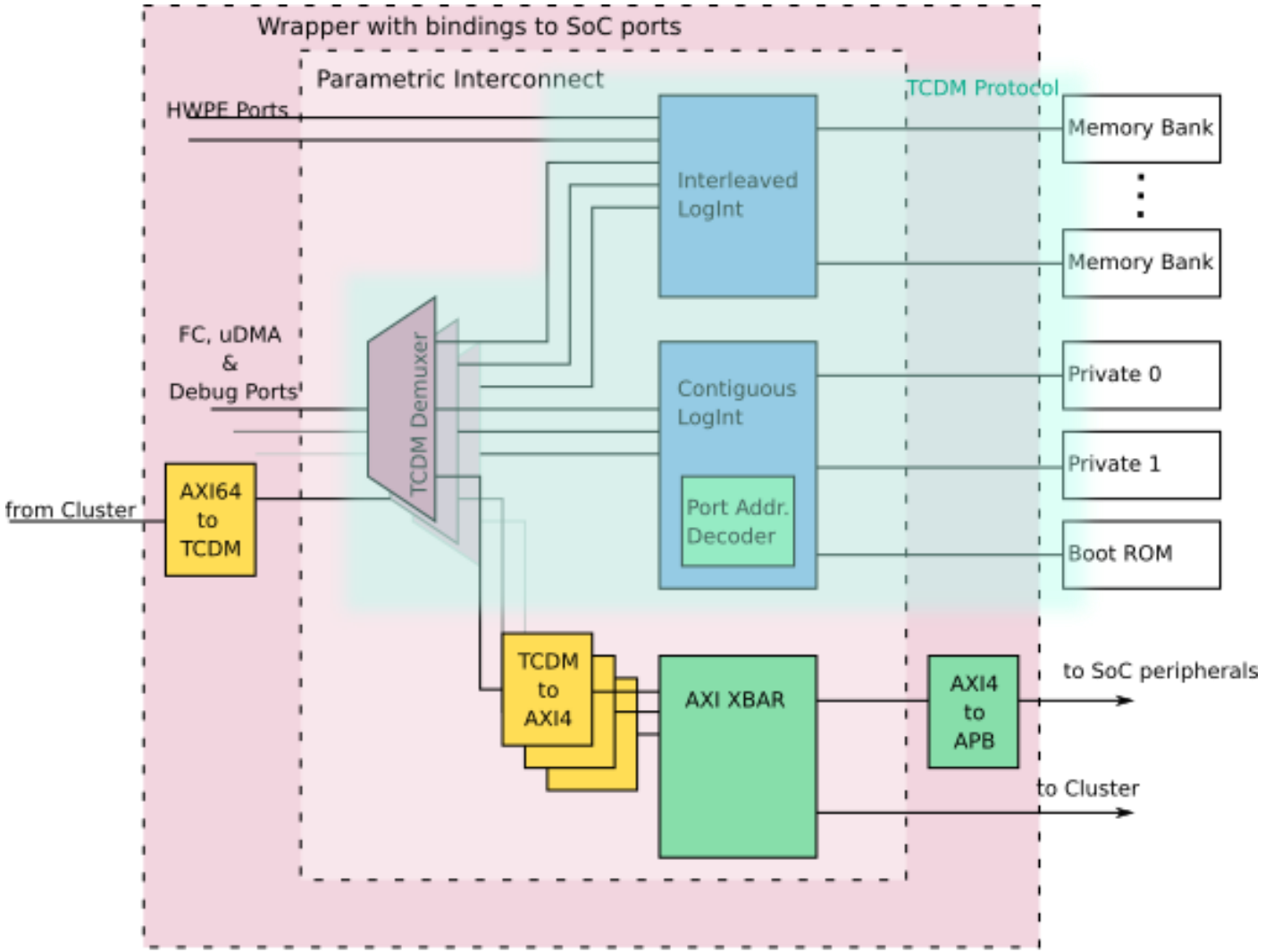
- APB Register File with Global configuration signals for SoC

Regname	Description
Boot Address	Contains the boot address of the core.
Fetch Enable	Enables instruction fetching in the core. By default controlled with an external signal (default 1)
Padmux	Signals used by pad_control to multiplex between dual usage of pads (GPIO or peripheral)
Pad Configuration	Controls the special pad control signals when the pad is in GPIO mode
JTAG Register	

L2 RAM Multibank

- Contains Wrappers for SRAM (or block memory) macros
- Internal address conversion
  - Address bit truncation
  - Offset subtraction if necessary and
  - conversion to 32-bit word addressing (wordwidth of SRAMs is 32-bit, core takes care of misaligned load/stores in hw)
- Protocol converter
  - assign gnt = request
  - r_valid delayed by one cycle

SoC Interconnect

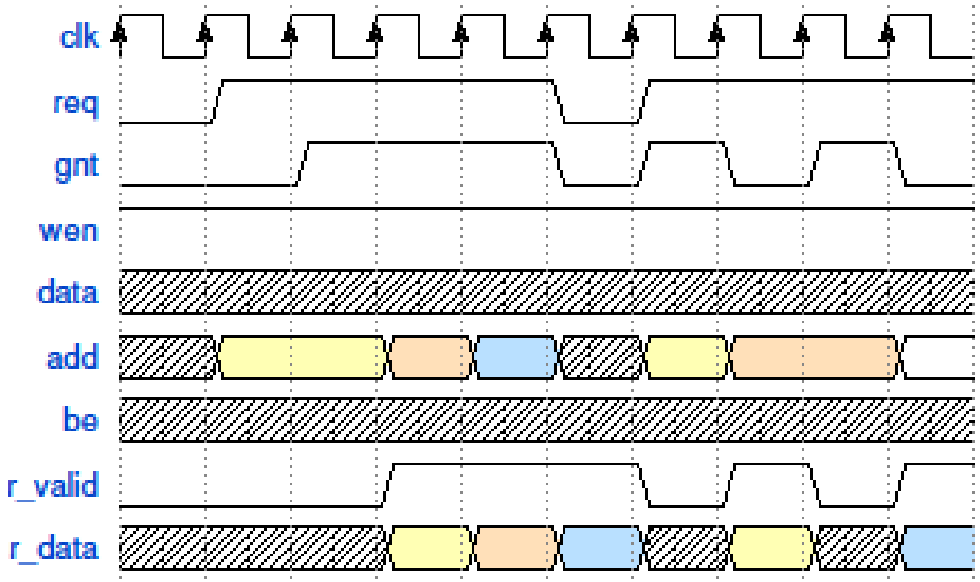


TCDM Protocol

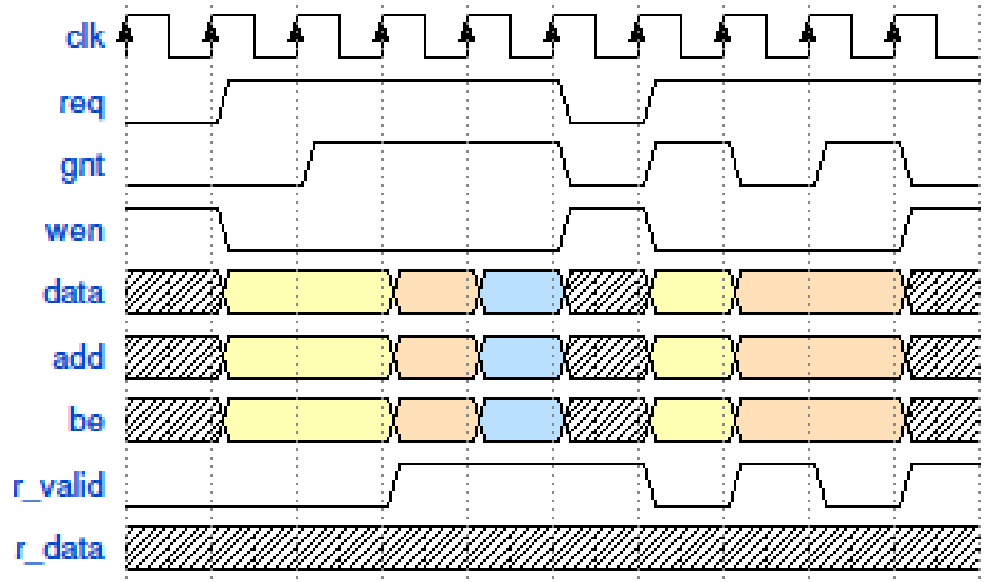
- Single cycle latency protocol
- Used for communication between core and memories
- Does not allow multiple outstanding transactions!
- Req must not depend on gnt, but gnt typicallcy does combinationally depend on req.

TCDM Protocol

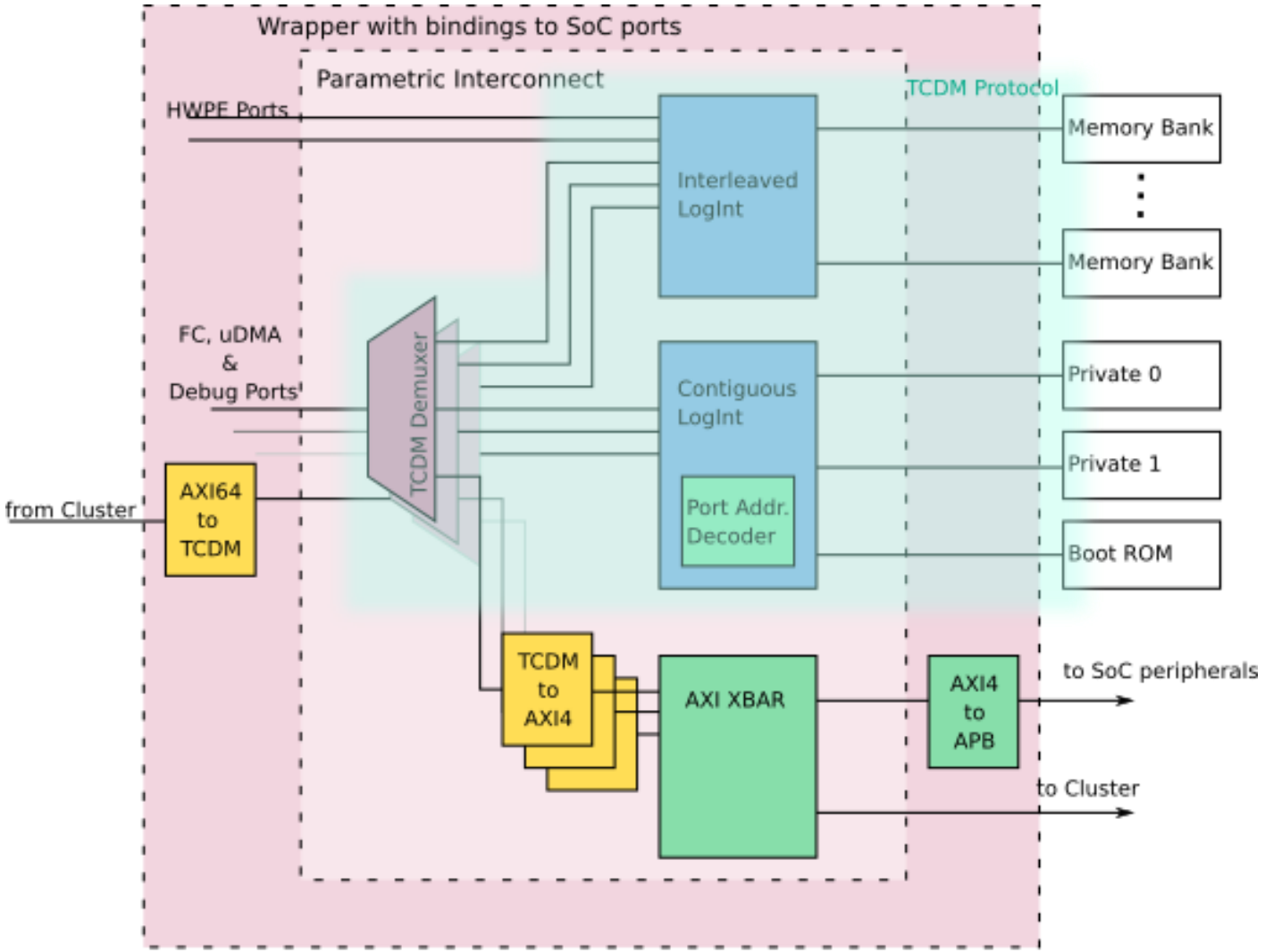
Read Transaction

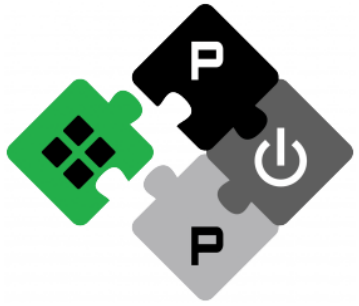


Write Transaction



SoC Interconnect





Training resumes at 13:00

Update: Resuming at 13:15 (food didn’t arrive in time:-)

PULP-SDK vs PULP-RUNTIME

PULP-SDK

- Fully-featured SDK
- Drivers
- Complex
- FPGA support

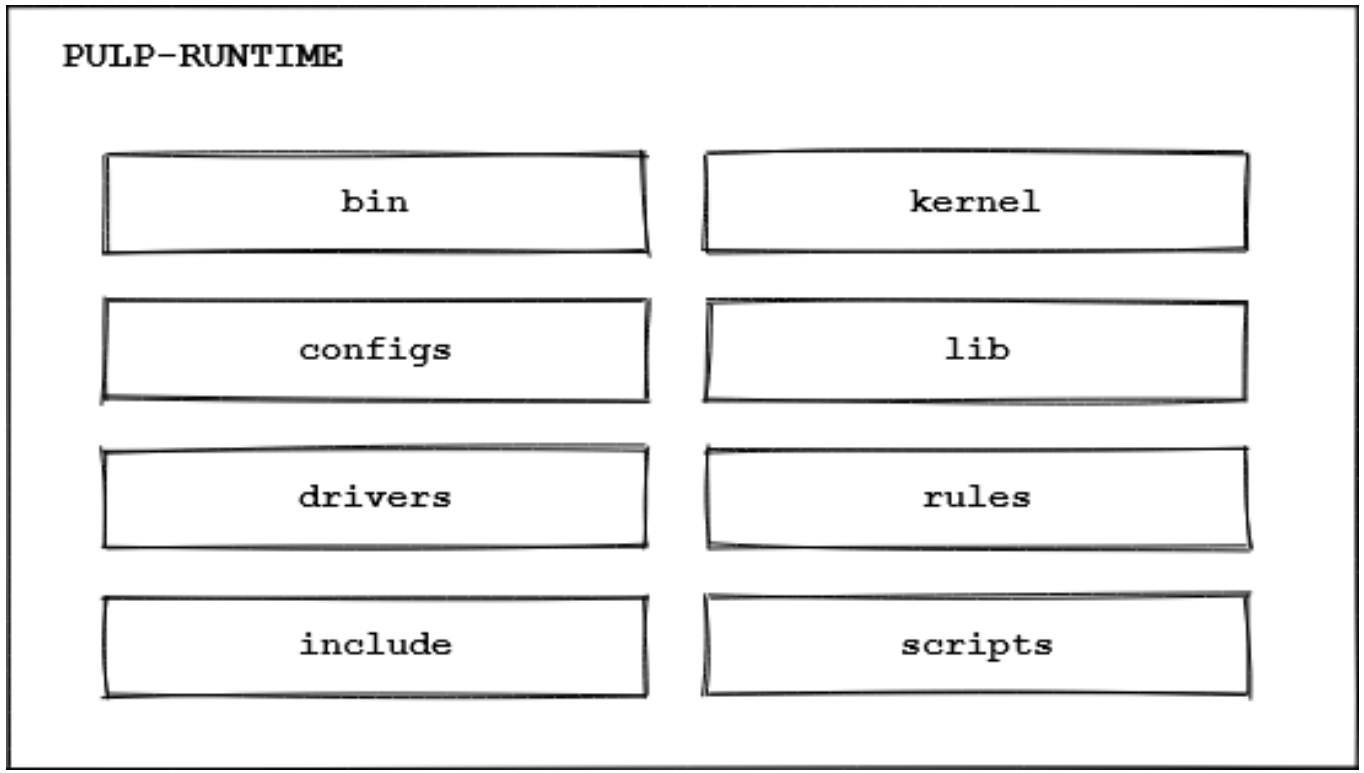
PULP-RUNTIME

- Minimal bare-metal runtime
- Boot-to-main
- Only uart driver
- FPGA support
- Active

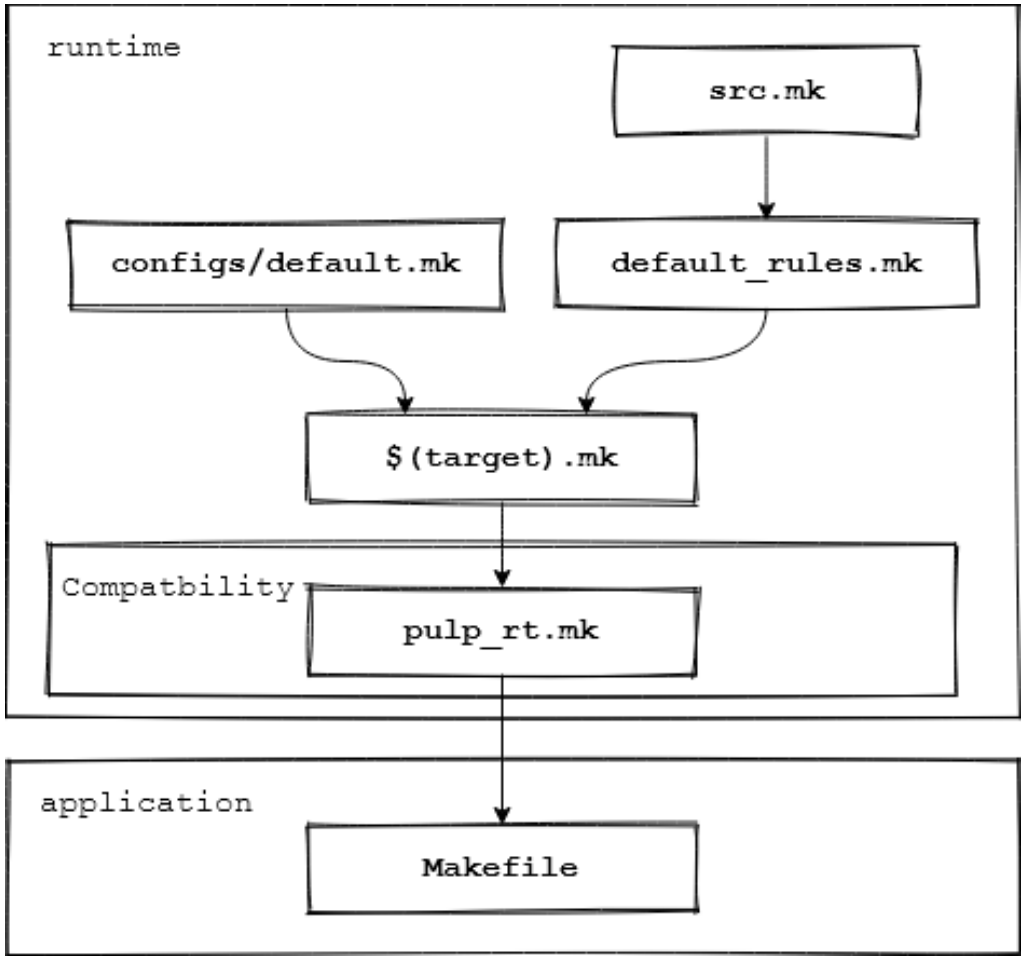
PULP-RUNTIME

- crt0.S to main() with minimal initialization
- Only uart driver available
- HAL available

PULP-RUNTIME - Overview



PULP-RUNTIME – Build Flow



PULP-RUNTIME – Hello World - Setup

- **Directory structure**
  - training/pulpissimo (v6.0.0) git clone https://www.github.com/pulp-platform/pulpissimo
  - training/pulp-runtime (v0.0.6) git clone https://www.github.com/pulp-platform/pulp-runtime
  - training/sw git clone https://www.github.com**/pulp-training/sw**
- **Compiler (pulp-gcc)**
  - https://github.com/pulp-platform/pulp-riscv-gnu-toolchain

PULP-RUNTIME – Hello World - Demonstration

1. Simulator location

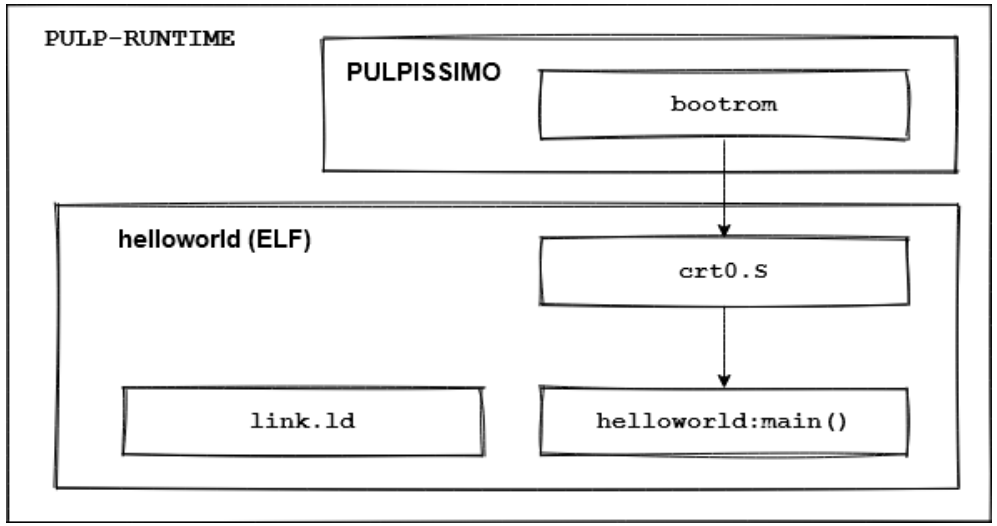
$ source setup/vsim.sh

2. Configuration

$ source configs/pulpissimo.sh

3. Compiler

$ export PULP_RISCV_GCC_TOOLCHAIN=...



PULP-RUNTIME – FPGA specific

1. ARCHI_FPGA_FREQUENCY
2. ARCHI_FPGA_FC_FREQUENCY
3. configs/fpgas/pulpissimo/*.sh

PULP-RUNTIME – Trivial Driver

1. We add a trivial driver to the pulp-runtime

```
$ cd sw/runtime-trivial-driver
```

PULP-RUNTIME – Trivial Driver - Exercise

1. Pass the second test in runtime-trivial-driver. Put the required macro and function in a separate .c/.h file.

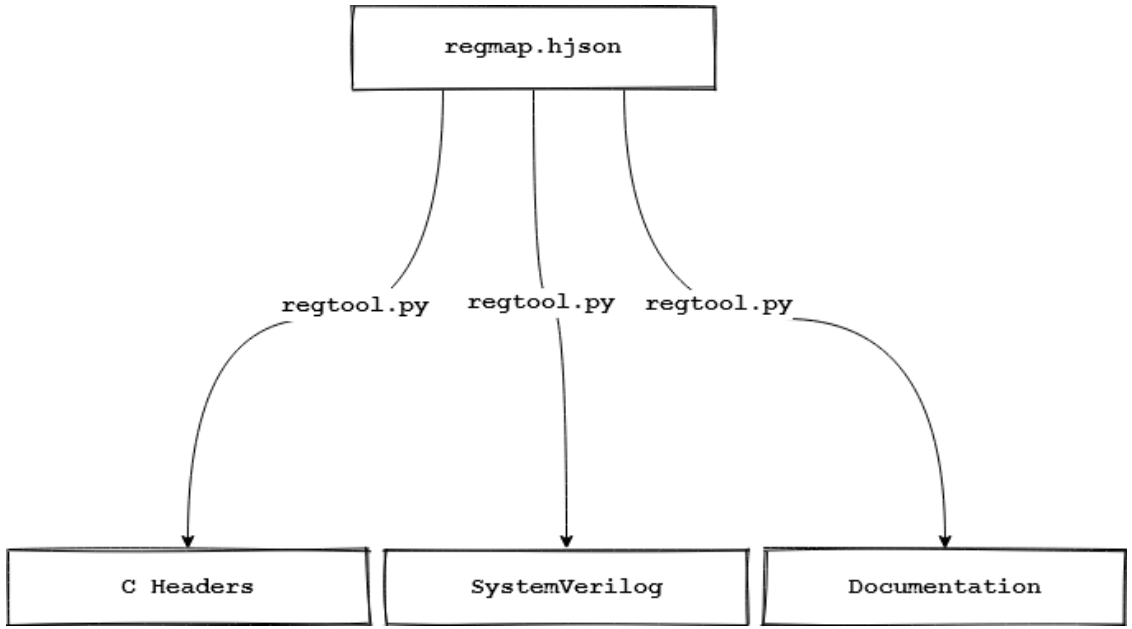
$ cd sw/runtime-trivial-driver

Exercise Time

- Join a breakout room of your choice
- In case you need help or have a question, visit: https://bit.ly/37nnI8P and enqueue yourself
- If you cannot use Zoom to share your screen or have issues with it: https://fisch.ddns.net/call/mhq9864w
- Consult https://fisch.ddns.net/sites/pulp_training for SoC schematics and FAQ (hopefully we have time to update it adhoc)

PULP-RUNTIME - Reggen

- Open-Source
- Used in tapeout
- Single source of truth
- Easier hw/sw co-design
- Lowrisc IP supported



PULP-RUNTIME – Reggen – PULP patches

- **We at ETH added some patches**
  - Tilelink is rather complicated and we don't use it
  - Add support for register_interface (simple protocol to access register)
  - Lots of protocol converters (AXI, APB, TCDM (partial)) and CDC
  - «reg» keyword to hjson

PULP-RUNTIME – Reggen - Demonstration

1. We show how the hjson description looks like
2. We generate a header file and SystemVerilog code from it
3. We use it a small program

$ cd sw/runtime-reggen

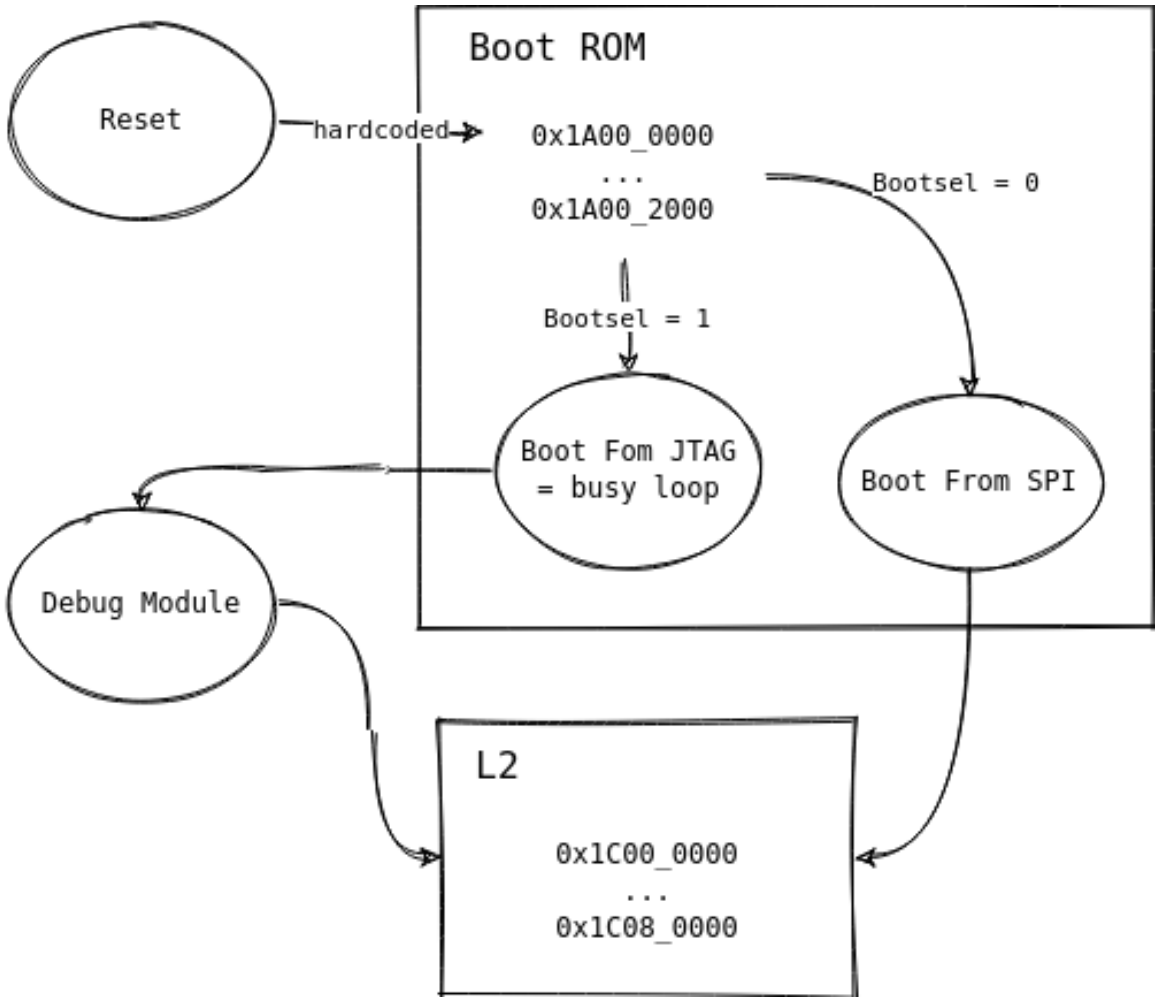
PULP-RUNTIME – Reggen - Exercise

1. Integrate the generated header file into pulp-runtime
2. Try to generate documentation from the hjson description
3. Explore the --help options

PULPissimo – Booting

1. Boot procedure
2. Introduction to Linkerscripts
3. Boot code, compile and link

PULPissimo – Boot procedure

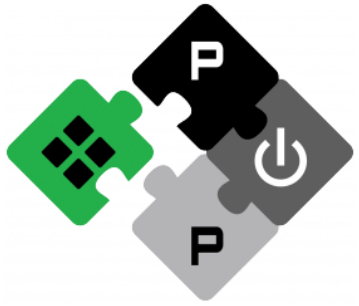


PULPissimo – Introduction to Linkerscripts

- **Compiler groups instructions and data in sections**
  - .text = instructions
  - .data = initialized variables
  - .bss = zero initialized variables
  - .rodata = read only data
- **Linkerscript = Set of rules on how to map sections to memory**

PULPissimo – Bootcode - Demonstration

$ cd boot_code/

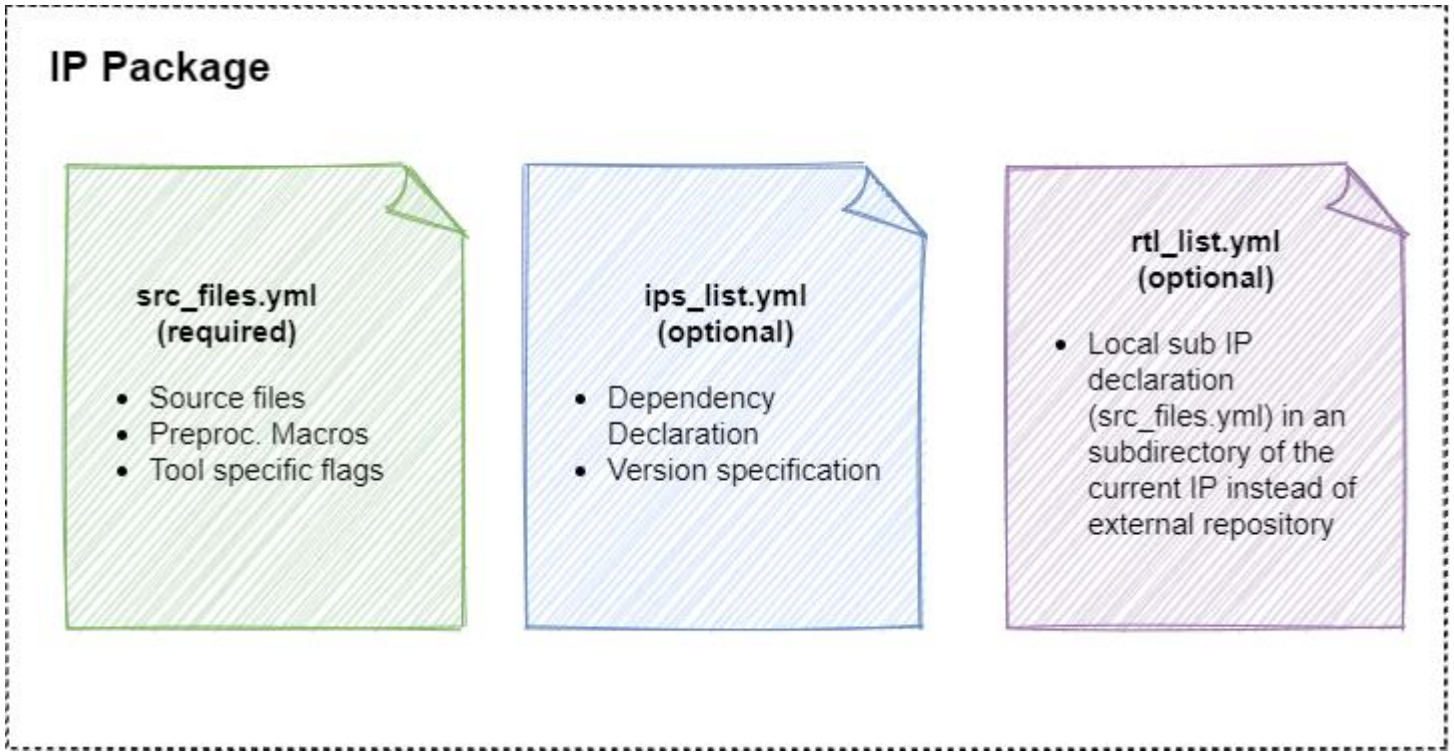


RTL Development Flow / Tools

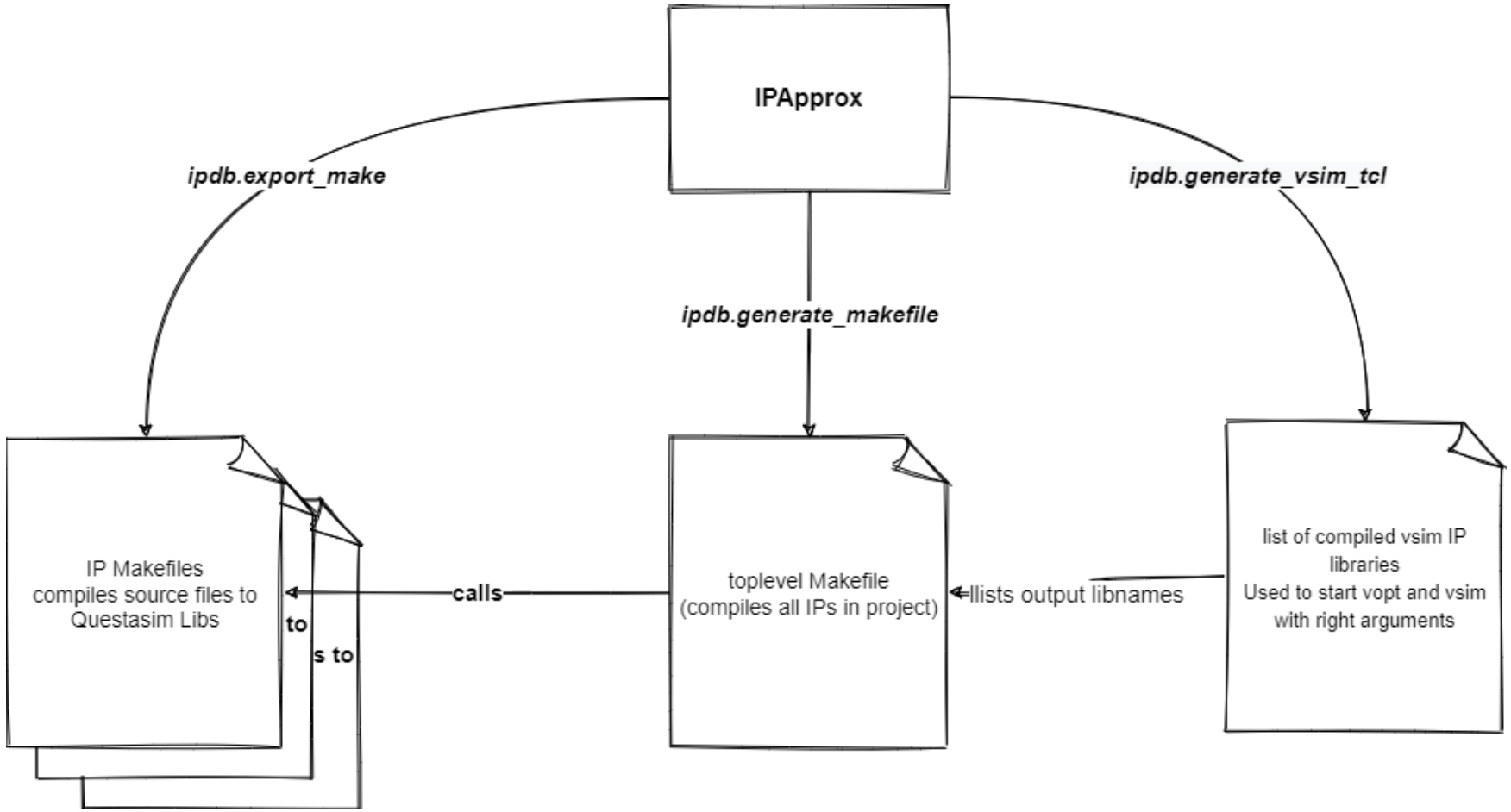
IP Dependency Management (IPApprox)

- Transitively resolves Dependencies between IPs
- Automatically checks out sub IP repositories
- Manages tool and target specific file sets
- Generates analyzes and elaborate scripts for simulation, ASIC & FPGA Synthesis
- Called by two python scripts (in PULPissimo they are called update_ips & generate_scripts)

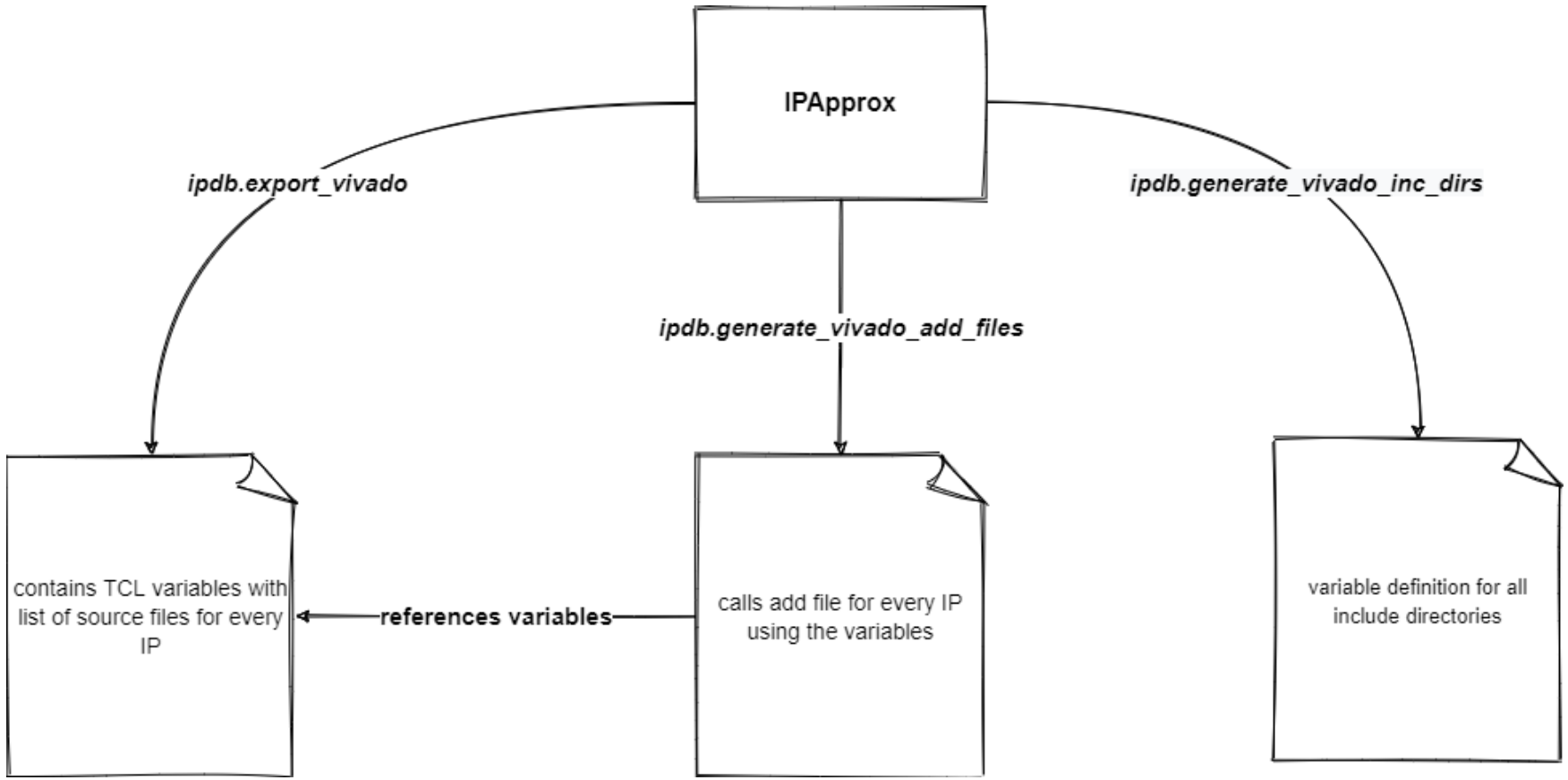
IPApprox


IP Package
src_files.yml
(required)
Source files
Preproc. Macros
Tool specific flags
ips_list.yml
(optional)
Dependency Declaration
Version specification
rtl_list.yml
(optional)
Local sub IP declaration (src_files.yml) in an subdirectory of the current IP instead of external repository

IPApprox Questasim Output



IPApprox FPGA Output



IPApprox Development Flow

Toplevel Modifications

1. If modification is in the toplevel, just update src_files.yml in RTL directory.

Independent Sub-IP Modifications

1. modify the IP in the checked out IPs directory on a new feature branch
2. Change the version in the dependent IPs to the new feature branch
3. If you add new dependencies, you have to commit and push the changes to the ips_list.yml
4. Run update-ips to resolve newly added dependencies and generate the new tcl files for simulation and synthesis
5. Once your changes are stable, commit and tag them and change version in all dependent packages to the new commit/release tag

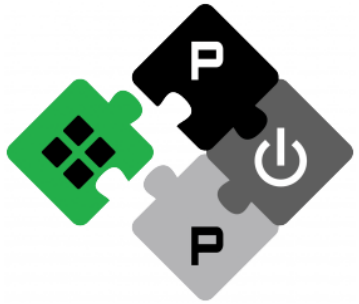
IPApprox Exercise – Integration of a Dummy VIP

Time to try it yourself:

- In this little exercise you are going to practice the IP integration flow using IPApprox
- Source files and Exercise description on: https://github.com/pulp-training/dummy_vip

IP Dependency Management (Bender)

- Transition planed in the next couple of weeks
- Written in rust
- Better Documentation
- More stable dependency resolution and conflict management
- (Yet) not flexible enough for subrepo flow used for pulp_soc

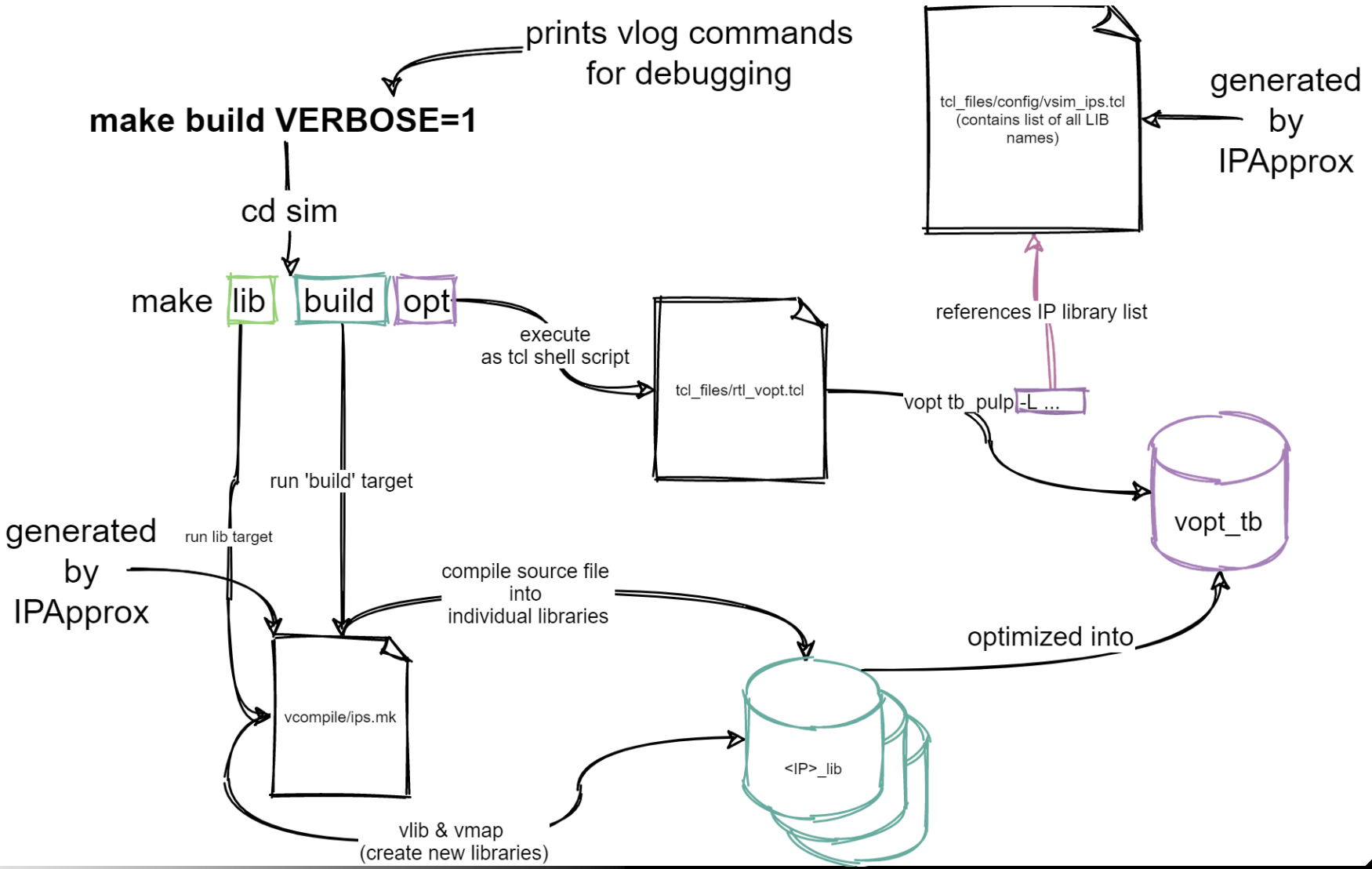


PULP PLATFORM

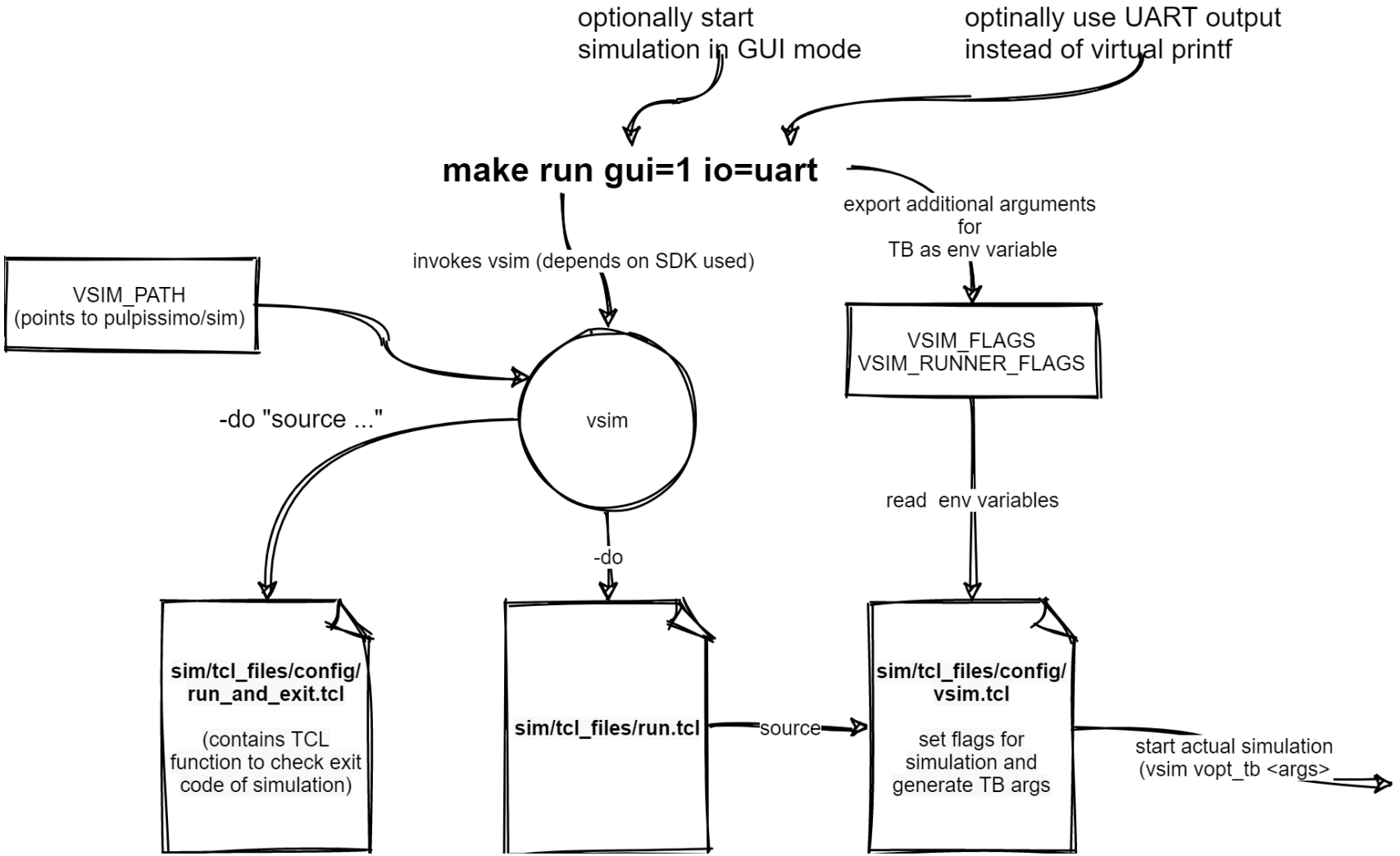
Open Source Hardware, the way it should be!

Simulation and Debug Flow

Build Flow of RTL Platform



Simulation Invocation



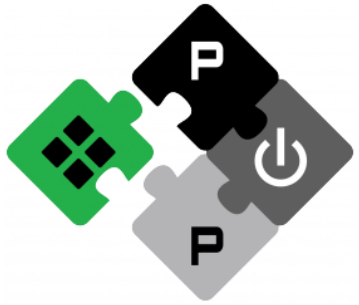
Final Exercise/Homework ☺

- Development of pulp-runtime application (driver interaction)
- Training of RTL debugging skils around PULPissimo
- You find the Exercise Files on: https://github.com/pulp-training/sw/tree/main/configure_fll_debug_rtl
- This is a more involved exercise and requires some code exploration skills. Don't hesitate to ask if you have troubles.

We would appreciate your Feedback!

- Please let us know what you thought of this first training day by filling the feedback form below:

https://bit.ly/2KbDchO



PULP PLATFORM

Open Source Hardware, the way it should be!

PULP Training Day 2

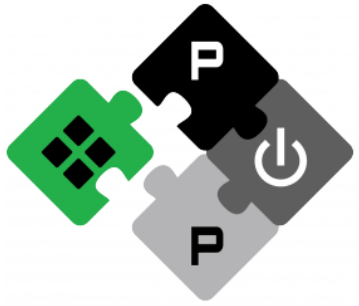
Robert Balas <balasr@iis.ee.ethz.ch>
Manuel Eggimann <meggimann@iis.ee.ethz.ch>

 http://pulp-platform.org  @pulp_platform  https://www.youtube.com/pulp_platform

Programm of Day 2

Day 2

- FPGA Port
- PULP IP Landscape
- PULPIssimo Memory Layout Modification
- Hands-on Full-stack IP Integration Exercise

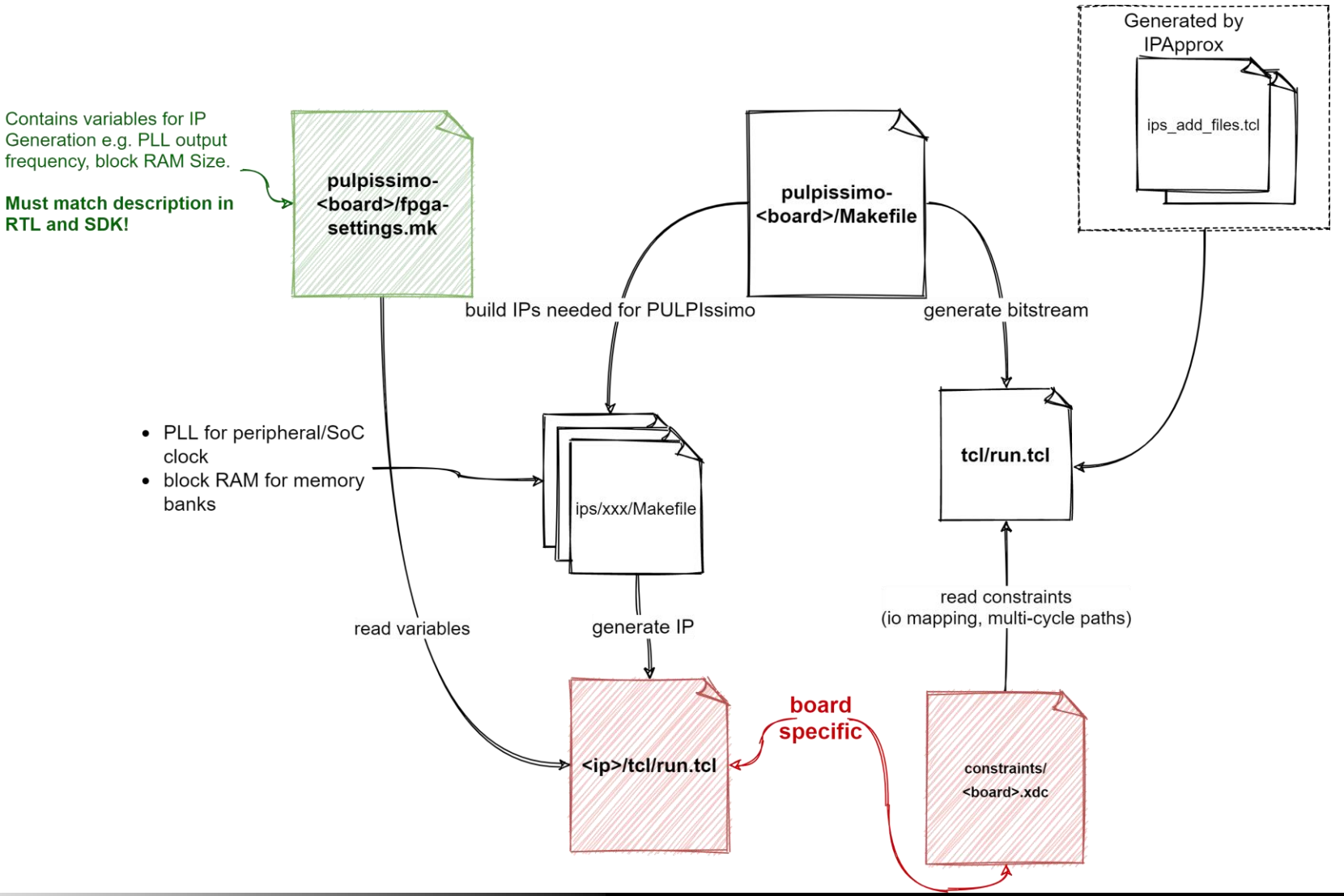


FPGA Port

Folder Structure

- fpga
  - pulpissimo (contains auto generated tcl script form IPApprox)
  - pulpissimo-<fpga-board>
    - rtl (port specific source files, i.e. wrappers for block RAM or clock managers)
    - ips
      - <ips_instantied by wrappers in rtl dir>
        - tcl (contains script to generate IP independent of PULPissimo)

FPGA Bitstream Generation



FPGA Synthesis Flow - Wrappers

Name	Description
xilinx_pulpissimo.sv	Toplevel wrapper of the whole pulpissimo. Converts differential clock input to single ended clock for pad_xtal_in.
pulp_clock_gatinx_xilinx.sv	Dummy clock gating cell. **Must be replaced with a correct implementation for peripherals to work.**
pad_functional_xilinx.sv	Wrapper for IOBUF cells
fpga_bootrom.sv	Dummy implementation. Hardwired to always respond with jal x0,0
fpga_clk_gen.sv	Wrapper for Xilinx clock manager generates soc_clk and periph_clk from reference clock. **Not at-runtime configurable**
fpga_interleaved_ram.sv	Wrapper for blockram macro generated by IP make targets
fpga_private.sv	Same as above
fpga_slow_clk_gen.sv	Certain FPGA boards have extremely high input frequencies (e.g. Genesys2). This wrapper instantiates PLL to slow down to intermediate freq (256*32768Hz) and feeds it to divide by 256 clock divider. Xilinx clock managers cannot go slower than 4.69MHz, that's why.

FPGA Simulation Tipps

- Use ILA Cores on bus signals
- Problems of ILA cores and Genesys2 board (jtag frequency)

Techcells

- Contain technology dependent cells like clock gates for manual instantiation in design
- Must be replaced with tech-specific module implementations that internally instantiate the library cells
- I.e. create a new IP with the replaced modules that depends on techcells_generic (this forces correct compile-order and module override behavior)

Techcells

Generic SRAM

Clk AND

Clk Buffer

Glitch-free Clk Gate

Clk Inverter

Clk Mux (clk bypass)

Clk Xor (freq. doubler)

Programmable Clk Delay (glitch filtering)

Common Cells

- Contains commonly used high-level modules that are independent of technology
- Contains Verilog Macros for uniform declaration of registers
- Contains a couple of commonly used assertion macros

Common Cells

Clk Divider, Rst Synchronizer

CDC Crossings

- Gray FIFO,
- 2phase HS
- Edge Detector
- Serial Synchronizers

Counters

- Delta Counter
- LFSRs

Datapath Elements

- Address Decoder (Heavily used in soc_interconnect)
- ECC Decoder/Encoder
- Gray2Binary/Binary2Gray
- Leading Zero Counter
- Stream (ready-valid pipeline) Building Blocks

Data Structures

- Counting Bloom Filter
- FIFO
- SRAM Behavioral
- Pseudo Least Reccently Used Tree

AXI IPs

Width Converters	Mux/Demux	Protocol Converters (APB, AXI-Lite, Atomics Filter
Fully-Connected XBAR (AXI or AXI-Lite)	Clock Domain Crossings	Pipeline Regs
Burst Splitter	Slave Isolator	Address Rewriter

Exercise Time

- Clone/Pull the latest changes of the exercise repo (there are new changes since yesterday): https://github.com/pulp-training/sw
- Switch to the *memlayout-exercise* and follow the instructions on: https://github.com/pulp-training/sw/tree/main/memlayout-exercise
- Join a breakout room of your choice
- In case you need help or have a question, visit: https://bit.ly/37nnl8P and enqueue yourself
- If you cannot use Zoom to share your screen or have issues with it: https://fisch.ddns.net/call/mhq9864w
- Consult https://fisch.ddns.net/sites/pulp_training for SoC schematics and FAQ (hopefully we have time to update it adhoc)

- ***We will continue at 13:15***

Full-stack AXI IP Integration

1. Write memory map description of IP in HJSON
2. Generate register-file using reggen
3. Develop wrapper that instantiates reg-file, IP, protocol converters and (if at all necessary) additional glue logic
4. Package and register IP using IPApprox
5. Instantiate wrapped IP in pulpissimo, modify soc_interconnect_wrap.sv, soc_mem_map.svh
6. Generate header file and develop driver
7. Test integration in RTL Simulation

Exercise Time

- Clone/Pull the latest changes of the exercise repo (there are new changes since yesterday): https://github.com/pulp-training/sw
- Open the exercise instructions on Github: https://github.com/pulp-training/sw/tree/main/fullstack_ip_integration
- Join a breakout room of your choice
- In case you need help or have a question, visit: https://bit.ly/37nnl8P and enqueue yourself
- If you cannot use Zoom to share your screen or have issues with it: https://fisch.ddns.net/call/mhq9864w
- Consult https://fisch.ddns.net/sites/pulp_training for SoC schematics and FAQ (hopefully we have time to update it adhoc)

- **We will wrap things up at 17:37**