Caribou Boreal

A unified, modular and configurable firmware

Y. Otarid (CERN)

Caribou Developers Meeting – 14.05.2024

Legacy structure and revision objectives

One Git project per device
→ *peary-firmware-DEVICE*

One Git project for common IPs
→ *peary-firmware-iprepo*

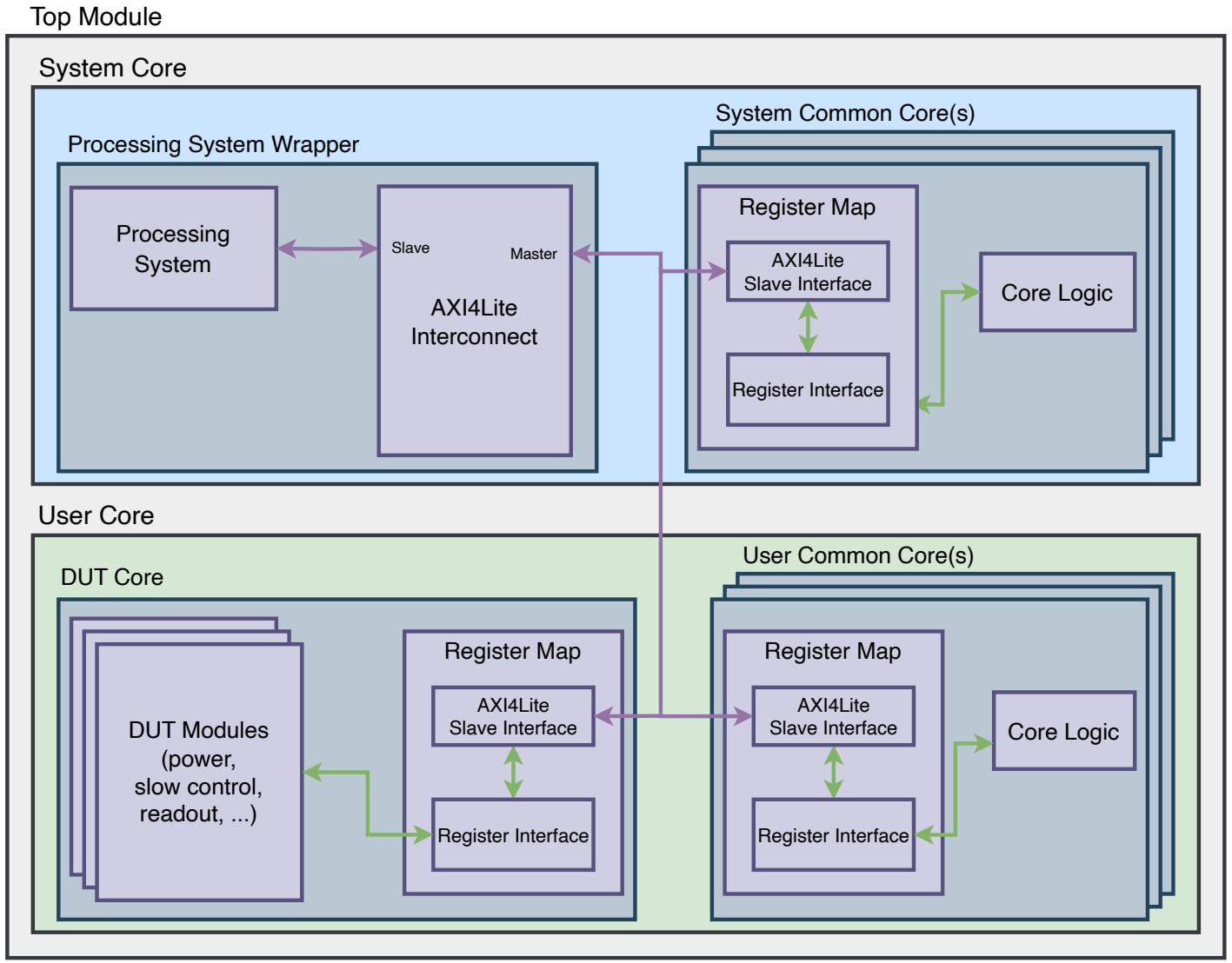
Caveats → Tedious development workflow:
- Block design recreated and reconfigured for every project
- Common IPs copy-pasted to new project and often modified locally
- Vivado version inconsistent across projects
- No HDL simulation / verification
- No CI/CD workflow
- No documentation

Objective → Smooth development workflow:
- One Git project for all devices
- Block design and common IPs configured and ready in the project
- Unified Vivado version
- Reinforced HDL simulation / verification
- Streamlined CI/CD workflow
- Documented code

Caribou Peary Firmware(s)

Caribou Boreal Firmware (link)

Boreal firmware architecture



Legend

- AXI4Lite interface bus
- Core interface bus

Processing System wrapper:

- Exported HDL from block design including Processing System and Reset IP

System common core(s):

- Common cores/IPs related to the system (Zynq board, CaR board) features

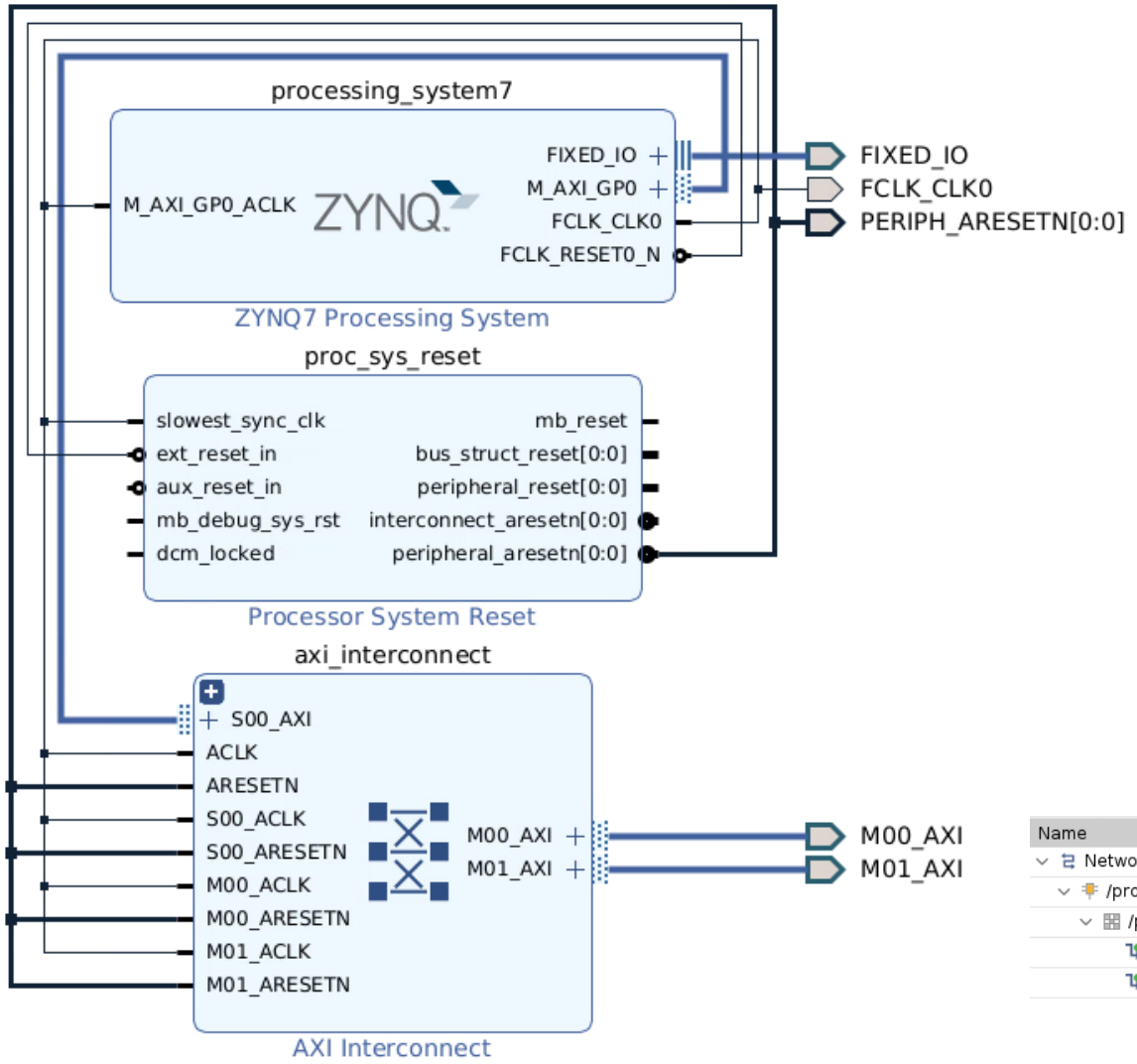
User common core(s):

- Common cores/IPs related to external or user functionalities

DUT core:

- User core including device-specific control and readout logic

Processing System Wrapper

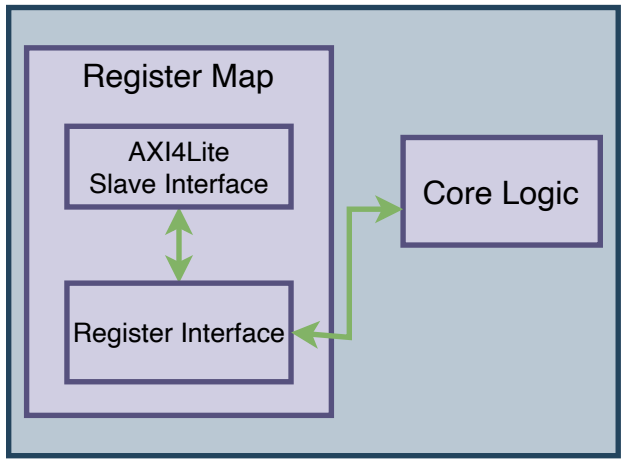


- Pre-configured for the Caribou Zynq board
- One AXI4Lite interface per block (system/user core) with a unique address
- Exported as a VHDL file
- To be modified and extended according to the system evolution and features extension

Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/processing_system7					
/processing_system7/Data (32 address bits : 0x40000000 [1G])					
/M00_AXI	M00_AXI	Reg	0x43C3_0000	8K	0x43C3_1FFF
/M01_AXI	M01_AXI	Reg	0x43C1_0000	64K	0x43C1_FFFF

Core base design

- Register map accessible via AXI interface, with configurable number of registers
- Core logic connected to register map and to the core IO interface
- Example: Blink LED core




```
entity blink_led is
  generic
  (
    N_LED   : integer := 4;
    CLK_DIV : natural := 100000000
  );
  port
  (
    -- Clock signal
    clk_i           : in std_logic;
    -- Reset signal
    rst_i           : in std_logic;
    -- LED driving signal
    led_o           : out std_logic_vector(N_LED - 1 downto 0);

    -- AXI4Lite slave interface
    axi4lite_bus_io : inout axi4lite_bus_type
  );
end blink_led;
```

DUT Core IOs

AXI4Lite bus

```
entity reg_map_if is
  generic
  (
    -- Number of registers
    REG_N : integer := 16
  );
  port
  (
    -- Register data IN
    data_in_i       : in array_2d(REG_N - 1 downto 0)(AXI_DATA_WIDTH - 1 downto 0);
    -- Register data OUT
    data_out_o      : out array_2d(REG_N - 1 downto 0)(AXI_DATA_WIDTH - 1 downto 0);

    -- AXI4Lite slave interface
    axi4lite_bus_io : inout axi4lite_bus_type
  );
end reg_map_if;
```

DUT Core **Register Map**

Project configuration

Configuration file

```
--Board type constant
type board_type is (ZC706, ZCU102);
constant BOARD : board_type := ZC706;
```

Supported SoC boards

```
-- Project type constant
type project_type is (BLINK_LED);
constant PROJECT            : project_type := BLINK_LED;
```

Supported devices

```
-- AXI4Lite interface address width
-- ZC706 = 32 ; ZCU102 = 40
constant AXI_ADDR_WIDTH     : integer      := 32;
-- AXI4Lite interface data width
constant AXI_DATA_WIDTH     : integer      := 32;
-- AXI4Lite interface address LSB
constant AXI_ADDR_LSB       : integer      := (AXI_DATA_WIDTH/AXI_ADDR_WIDTH) + 1;
-- AXI4Lite data bytes
constant AXI_DATA_BYTES     : integer      := ((AXI_DATA_WIDTH - 1)/8) + 1;
-- AXI4Lite register address width
constant AXI_REG_ADDR_WIDTH : integer      := AXI_ADDR_WIDTH - AXI_ADDR_LSB;
```

AXI Interface Configuration

```
end package config;
```

Automatic DUT core instantiation

User Core

```
architecture behavioral of zc706_usr_core is

begin

  -- LED blinking block instantiation
  blink_led_gen : if PROJECT = BLINK_LED generate
    blink_led_inst : entity work.blink_led
```

Automatic constraints selection

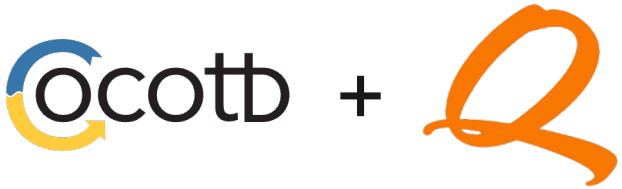
```
# Check condition
if {$project == "BLINK_LED"} {
  # ZC706 constraints
  if {$board == "ZC706"} {
}

# ZCU102 constraints
if {$board == "ZCU102"} {
```

XDC file

Simulation

- Cocotb, an open source coroutine-based cosimulation testbench environment for verifying VHDL and SystemVerilog RTL using Python.

- Questa simulator, but any other simulator can be used



Example cocotb successful simulation of register map block

```
 9980.00ns INFO     cocotb.reg_map_if               ***********************************************************
 9980.00ns INFO     cocotb.reg_map_if               Reading data from register 15 at address 0x43c1003c
 9980.00ns INFO     cocotb.axi4lite_bus_io.None     Read start addr: 0x43c1003c prot: 2 length: 4
10010.00ns INFO     cocotb.axi4lite_bus_io.None     Read complete addr: 0x43c1003c prot: 2 resp: 0 data: ec df 24 27
10300.00ns INFO     cocotb.reg_map_if                      Data-in word = 0x2724dfec
10300.00ns INFO     cocotb.reg_map_if                      Data-out word = 0x2724dfec
10300.00ns INFO     cocotb.regression               test_reg_map_if passed
10300.00ns INFO     cocotb.regression               ****************************************************************************************
                                                    ** TEST                          STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
                                                    ****************************************************************************************
                                                    ** test_reg_map_if.test_reg_map_if   PASS       10300.00           0.08     121366.38  **
                                                    ****************************************************************************************
                                                    ** TESTS=1 PASS=1 FAIL=0 SKIP=0                10300.00           0.10     106281.05  **
                                                    ****************************************************************************************
```

Boreal Manager

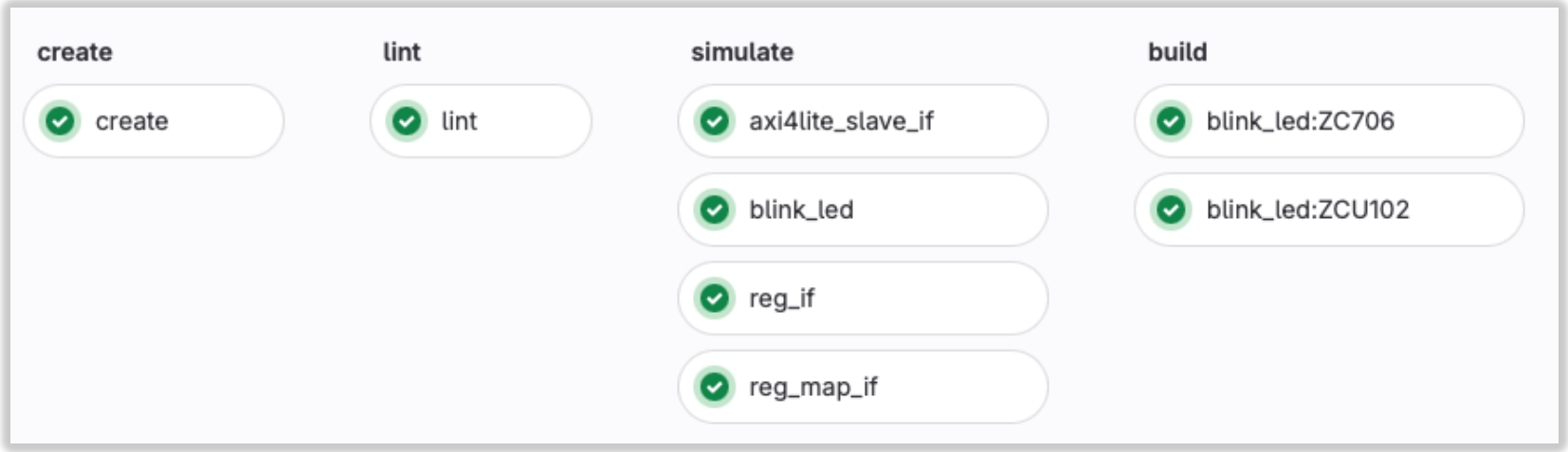
A python script that allows the user to:

- **Create the Vivado project**
  - Execute the TCL project creation script
- **Configure the Vivado project**
  - Choose the board and device
- **Build the Vivado project**
  - Synthesis + Implementation + bitstream generation

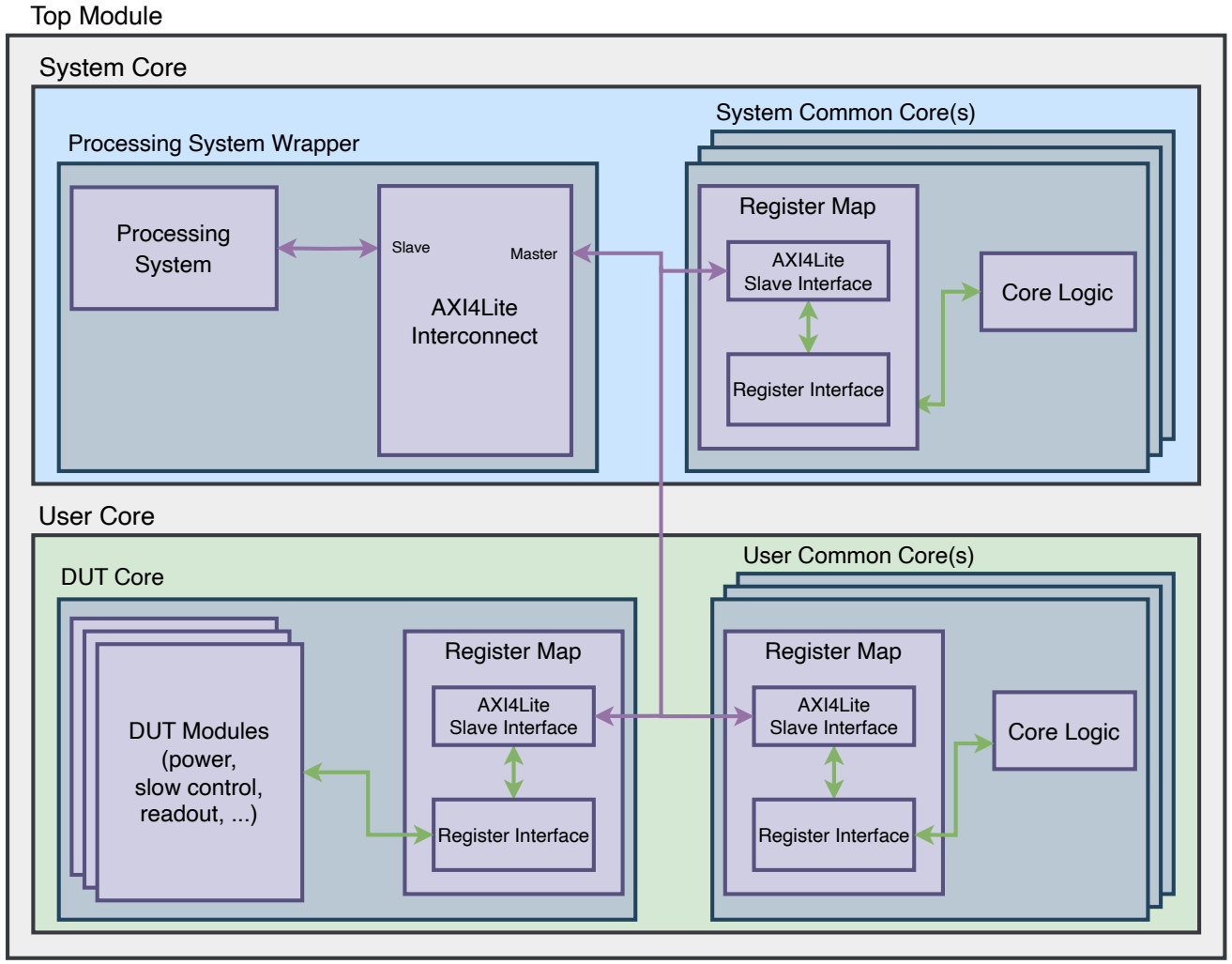
```
Boreal Vivado project manager

optional arguments:
  -h, --help            show this help message and exit
  --create              Create Vivado project
  --configure           Configure Vivado project
  --board {ZC706,ZCU102}
                        Board to be used
  --project {BLINK_LED}
                        Device project to be used
  --synth               Run synthesis
  --impl                Run implementation
```

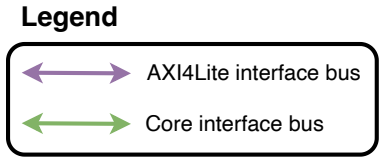
CI/CD workflow

- Single instance of project creation
- Code linting using *vhdl-linter* and *verible-format*
- Cocotb simulation of all cores
- Bitstream generation for all supported devices and for all supported boards



Boreal firmware landscape



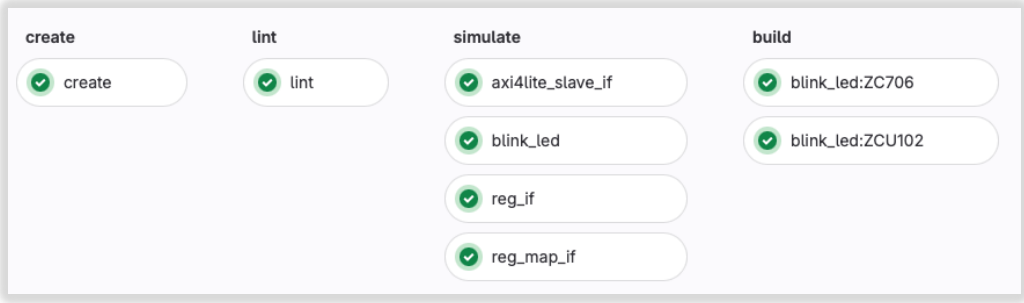


Boreal firmware (link):

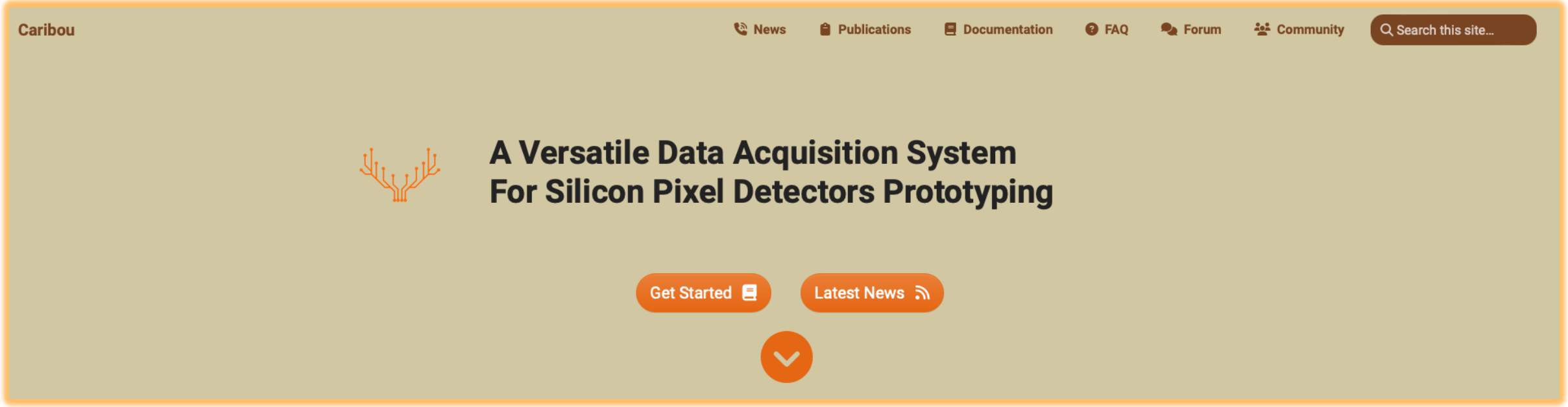
- Unified, modular and configurable architecture

CI/CD workflow:

- linting, simulation, building, deployment



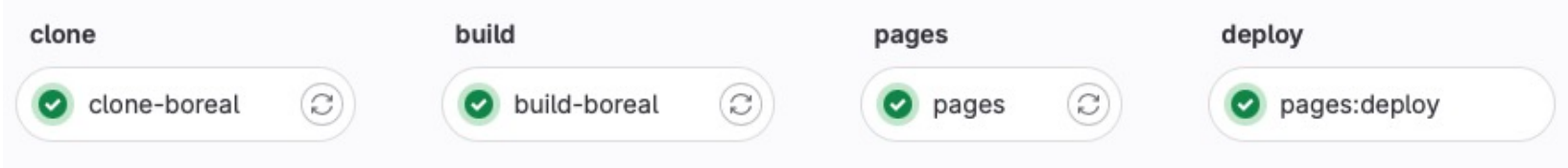
Caribou documentation website



Project website (link)

- Documentation
- Mattermost channel
- Publications
- Forum
- …

Automatic documentation builds and website deployments



Summary – Outlook

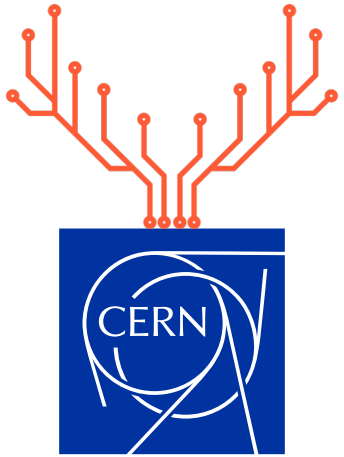
Summary

- New Caribou Boreal firmware architecture
- Unified, modular and configurable design
- Reinforced logic simulation and verification
- Streamlined CI/CD workflow
- Documentation website still in progress
- **Blink LED demonstrator project successfully tested on ZC706 including with Peary software integration**

Outlook

- Support of ZCU102 and test of Blink LED demonstrator project
- Define best documentation deployment workflow

Thank you



Contact

CERN
Younes Otarid
EP R&D
younes.otarid@cern.ch

home.cern