PULPissimo: Datasheet

The PULP team
pulp-info@list.ee.ethz.ch

May 9, 2020

Contents

1 Overview

PULPissimo is a 32 bit RI5CY single-core System-on-a-Chip. PULPissimo is the second version of the PULPino system and it can be extended with the multi-core cluster of the PULP project.

Differently from the simpler PULPino system, PULPissimo uses a more complex memory subsystem, an autonoumous I/O subsystem which uses the uDMA, new peripherals (eg the camera interface) and a new SDK.

Figure 1.1 shows a simplified block diagram of the SoC. As for PULPino, PULPissimo can be configured at design time to use either the RISC-V or zero-riscy. The peripherals are connected to the uDMA which transfers the date to the memory subsystem efficiently. The JTAG and the AXI plug have also access to the SoC. The AXI plug can be used to extend the microcontroller with a multi-core cluster or an accelerator. As for PULPino, the advanced debug unit is used to access to system and core registers, memories and memory-mapped IO via JTAG. A logarithmic interconnect allows to link the core and the uDMA to the memory banks simultaneously.

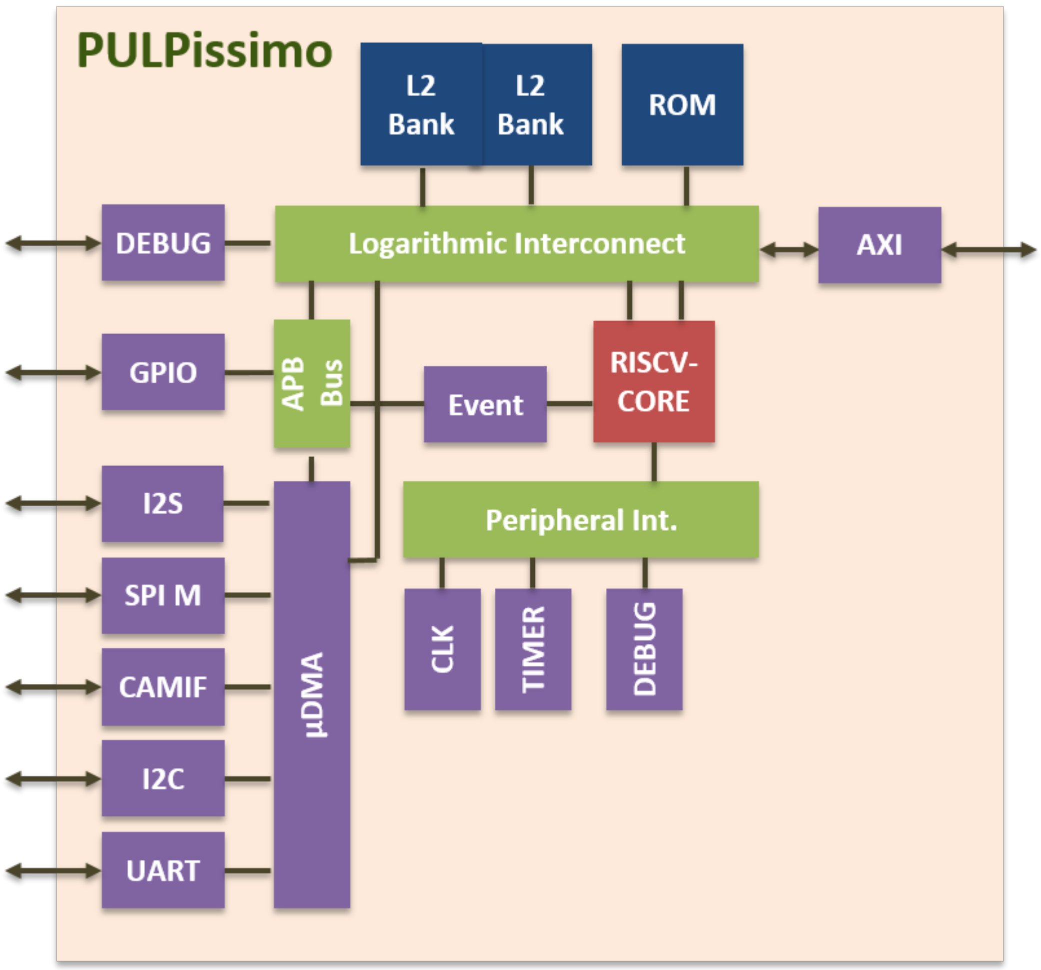


Figure 1.1: PULPissimo Overview.

PULPissimo is mainly targeted at RTL simulation and ASICs. The FPGA versions has not yet been implemented.

2 Memory Map

Figure 2.1 shows the default memory-map of PULPissimo, whereas Please, consult the uDMA documentation for the peripherals attached to the uDMA memory-map of configuration.

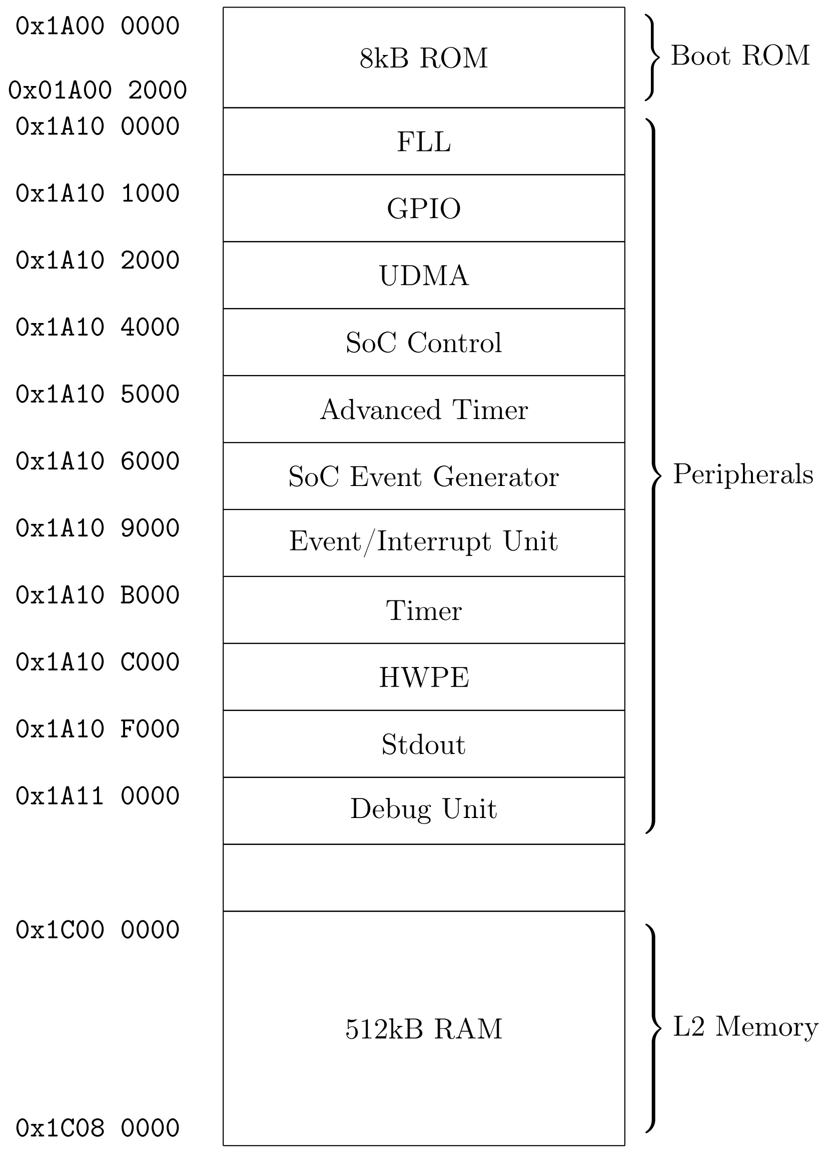


Figure 2.1: PULPissimo memory-map.

3 CPU Core

PULPissimo supports both the RISC-V and the zero-riscy RI5CY core. The two cores have the same external interfaces and are thus plug-compatible. Figure 3.1 and 3.2 show the two cores architectures.

For debugging purposes, all core registers have been memory mapped which allows to them to be accessed over the logaritmic-interconnect subsystem. The debug unit inside the core handles the request over this bus and reads/sets the core registers and/or halts the core.

The core supports performance counters. Those are mainly used for counting core internal events like stalls, but it is possible to count core-external events as well. For this purpose there is the `ext_perf_counters_i` port where arbitrary events can be attached. The core then increases its internal performance counter for this event type every time a logic high is seen on this port.

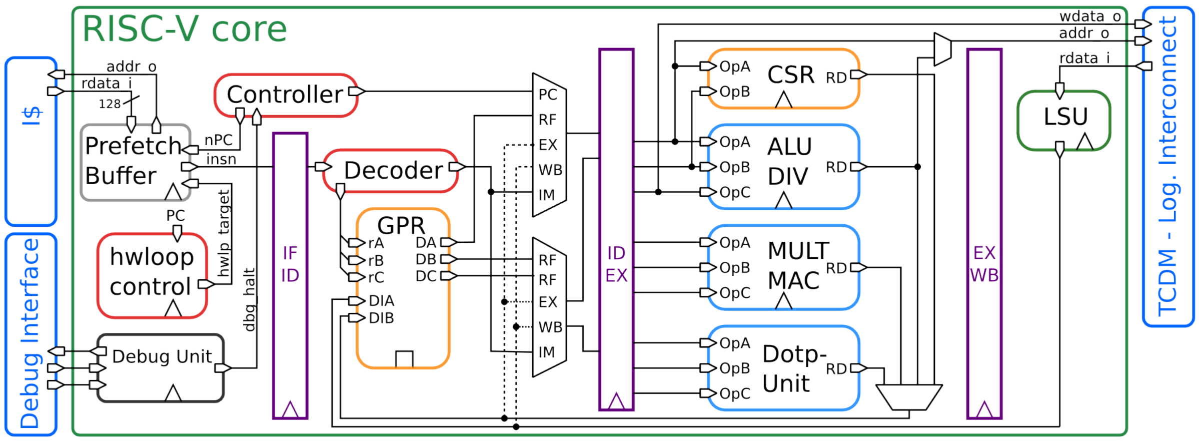


Figure 3.1: RISCY core overview

Take a look at the cores documentation for more details.

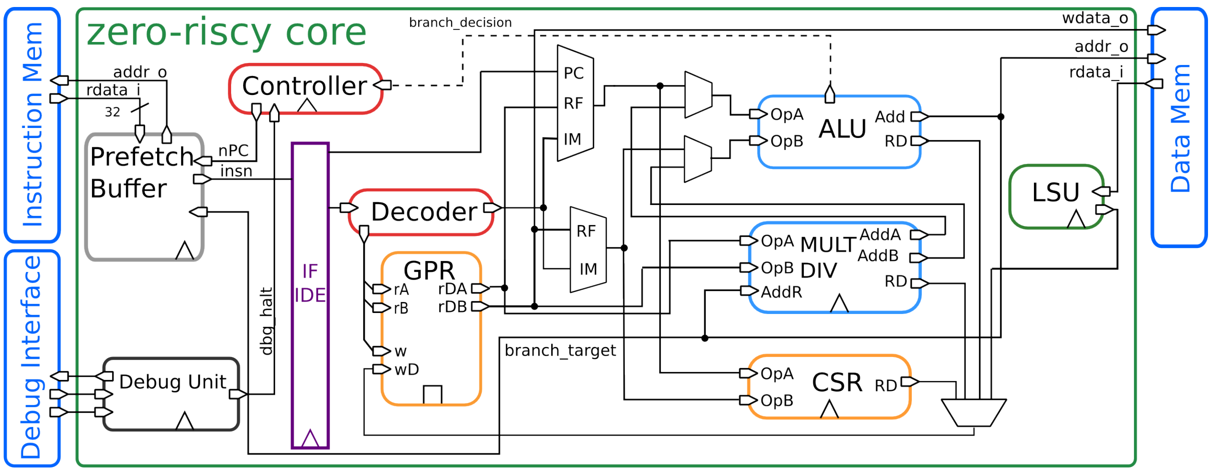


Figure 3.2: zero-riscy core overview

4 Debug Module for External Debug Support

The debug module in PULPissimo is compliant with the RISC-V External Debug Support specification $v1.13.1$. For more details please take a look at the documentation in the debug module folder and consult the RISC-V External Debug Support specification.

5 Peripherals

Most of the peripherals in PULPissimo are connected to the uDMA subsystem which efficiently handles all the data-transfers autonoumsly. The uDMA must be programmed by the core via memory-mapped read and write operations to receive commands.

See the uDMA documentation for more details under the uDMA repository.

The GPIO, timers, event unit and event generator, debug and the FLLs are not connected to the uDMA instead but to the APB bus. Following a brief overview about these units is given.

5.1 FLL

PULPissimo containts 3 FLLs. One FLL is meant for generating the clock for the peripheral domain, one for the core domain (core, memories, event unit etc) and one is meant for the cluster. The latter is not used.

All the FLLs can be bypassed by writing to the JTAG register before the reset signal is asserted. See Section 5.3 for more details about the bypass register.

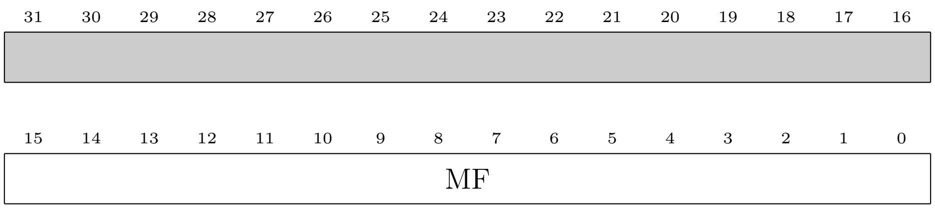
5.1.1 SoC FLL registers

Name	Address	Size	Type	Access	Default	Description
STATUS	`0x1A100000`	32	Status	R	`0x00000000`	FLL status register
CFG1	`0x1A100004`	32	Config	R/W	`0x00000000`	FLL configuration 1 register
CFG2	`0x1A100008`	32	Config	R/W	`0x00000000`	FLL configuration 2 register
INTEG	`0x1A10000C`	32	Config	R/W	`0x00000000`	FLL integrator configuration register.

Table 5.1: SoC FLL register table

5.1.2 STATUS

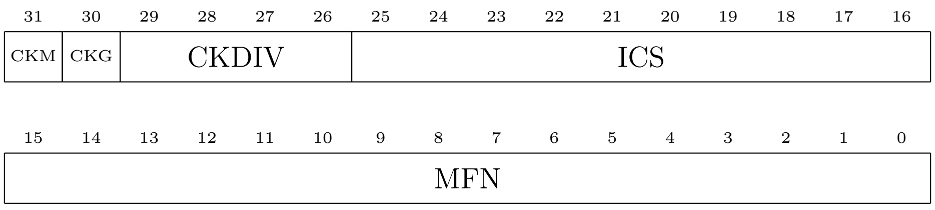
Address: `0x1A10_0000`
Reset Value: `0x0000_0000`



`Bit 15-0` **MF** *(R)* Current DCO multiplication factor value bitfield

5.1.3 CFG1

Address: `0x1A10_0004`
Reset Value: `0x0000_0000`



`Bit 31` **CKM** *(R/W)* FLL operation mode configuration bitfield

- 0b0: standalone
- 0b1: normal

`Bit 30` **CKG** *(R/W)* FLL output clock divider configuration

- 0b0: not gated
- 0b1: gated

`Bit 29-26` **CKDIV** *(R/W)* FLL output clock divider configuration

`Bit 25-16` **ICS** *(R/W)* DCO input code in standalone

`Bit 15-0` **MFN** *(R/W)* Target clock multiplication factor in normal mode

5.1.4 CFG2

Address: `0x1A10_0008`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DITH	OL	CKSEL	▨	LT											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCKL						UCKL						LG			

`Bit 31` **DITH** *(R/W)* Dithering activation

`Bit 30` **CKM** *(R/W)* Open loop when locked

- 0b0: disabled
- 0b1: enabled

`Bit 29` **CKSEL** *(R/W)* Configuration clock selection in standalone mode

- 0b0: DCO clock
- 0b1: Reference clock

`Bit 27-16` **LT** *(R/W)* Lock tolerance configuration. It is the margin around the multiplication factor within which the output clock is considered stable.

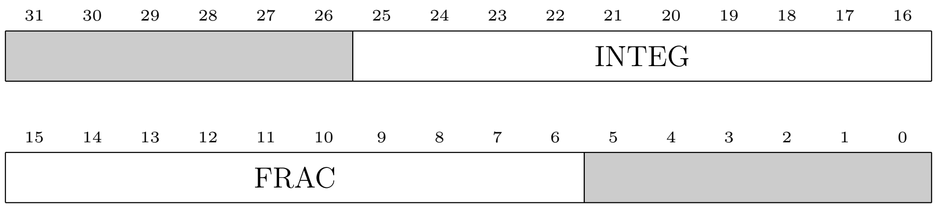
`Bit 15-10` **SCKL** *(R/W)* Number of stable REFCLK cycles until LOCK assert in normal mode. Uppper 6 bits of LOCK assert counter target in standalone mode.

`Bit 9-4` **UCKL** *(R/W)* Number of unstable REFCLK cycles until LOCK de-assert in normal mode. Lower 6 bits of LOCK assert counter target in standalone mode.

`Bit 3-0` **LG** *(R/W)* FLL loop gain setting

5.1.5 INTEG

Address: `0x1A10_000C`
Reset Value: `0x0000_0000`



`Bit 25-16` **INTEG** *(R/W)* Integer part of integrator state bitfield. It corresponds to DCO unit bits.

`Bit 15-6` **FRAC** *(R/W)* Fractional part of integrator state bitfield. It corresponds to dither unit input.

5.2 GPIO

Table 5.2: GPIO Signals

Signal	Direction	Description
`gpio_in[31:0]`	**input**	Transmit Data
`gpio_out[31:0]`	**output**	Receive Data
`gpio_dir[31:0]`	**output**	Request to Send
`gpio_padcfg[5:0][31:0]`	**output**	Pad Configuration
`interrupt`	**output**	Interrupt (Rise or Fall or Level)

5.2.1 PADDIR (Pad Direction)

Address: `0x1A10_1000`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	PADDIR

`Bit 31:0` **PADDIR**: Pad Direction.
Control the direction of each of the GPIO pads. A value of `1` means it is configured as an output, while `0` configures it as an input.

5.2.2 PADIN (Input Values)

Address: `0x1A10_1004`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	PADIN

`Bit 31:0` **PADIN**: Input Values.

5.2.3 PADOUT (Output Values)

Address: `0x1A10_1008`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	PADOUT

`Bit 31:0` **PADOUT**: Output Values.

5.2.4 INTEN (Interrupt Enable)

Address: `0x1A10_100C`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	IT	INTEN

`Bit 31:0` **INTEN**: Interrupt Enable.
Interrupt enable per input bit. INTTYPE0 and INTTYPE1 control the interrupt triggering behavior.

There are four triggers available

- `INTTYPE0 = 0, INTTYPE1 = 0`: Level 1
- `INTTYPE0 = 1, INTTYPE1 = 0`: Level 0
- `INTTYPE0 = 0, INTTYPE1 = 1`: Rise
- `INTTYPE0 = 1, INTTYPE1 = 1`: Fall

5.2.5 INTTYPE0 (Interrupt Type 0)

Address: `0x1A10_1010`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	T0	INTTYPE0

`Bit 31:0` **INTTYPE0**: Interrupt Type 0.
Controls the interrupt trigger behavior together with INTTYPE1. Use INTEN to enable interrupts first.

5.2.6 INTTYPE1 (Interrupt Type 1)

Address: `0x1A10_1014`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1	INTTYPE1

`Bit 31:0` **INTTYPE1**: Interrupt Type 1.
Controls the interrupt trigger behavior together with INTTYPE0. Use INTEN to enable interrupts first.

5.2.7 INTSTATUS (Interrupt Status)

Address: `0x1A10_1018`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	INTSTATUS

`Bit 31:0` **INTSTATUS**: Interrupt Status.
Contains interrupt status per GPIO line. The status register is cleared when read. Similarly the `interrupt` line is high while a bit is set in interrupt status and will be deasserted when the status register is read.

5.2.8 GPIOEN (GPIO Enable)

Address: `0x1A10_101C`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	GPIOEN

`Bit 31:0` **GPIOEN**: GPIO Enable.
Contains the enable bit per GPIO line.

5.2.9 PADCFG0-7 (Pad Configuration Registers 0-7)

Address: `0x1A10_1020 - 0x1A10_103C`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	PADCFG0-7

`Bit 31:0` **PADCFG0-7**: Pad Configuration Registers.
The pad configuration registers control various aspects of the pads that are typically used in ASICs, e.g. drive strength, Schmitt-Triggers, Slew Rate, etc. Since those configuration parameters depend on the exact pads used, each implementation is free to use the PADCFG0-7 registers in every way it wants and also leave them unconnected, if unneeded.

Writing to the PADOUTSET address (0x1A10_1040), the content of the PADOUT register is updated with its content "ored" with the write data.

Writing to the PADOUTCLR address (0x1A10_1044), the content of the PADOUT register is updated with its content "anded" with the inverted write data.

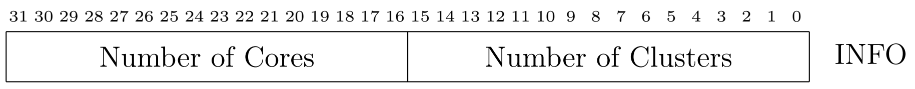
5.3 SoC Control

PULPissimo features a small and simple APB peripheral which provides information about the platform and provides the means for pad muxing on the ASIC.

The following registers can be accessed.

5.3.1 Info

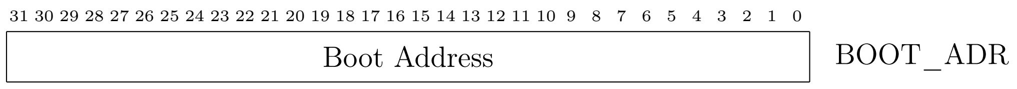
Address: `0x1A10_4000`
Reset Value: `0x0000_0000`



`Bit 31:0` **Info**: This register holds the number of clusters and the number of cores in the each cluster. It is a read-only register.

5.3.2 Boot Address

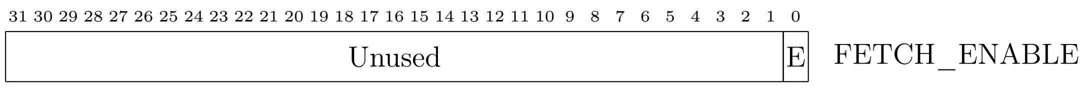
Address: `0x1A10_4004`
Reset Value: `0x1A10_0000`



`Bit 31:0` **Boot Address**: This register holds the boot address.

5.3.3 Fetch Enable

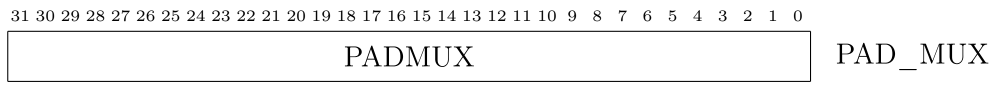
Address: `0x1A10_4008`
Reset Value: `0x0000_0001`



`Bit 31:0` **Fetch Enable**: This register contains the value of the fetch enable signal of the core.

5.3.4 PAD Mux

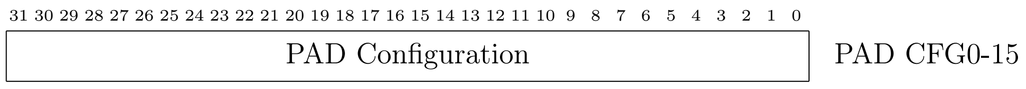
Address: `0x1A10_4010 - 0x1A10_401C`
Reset Value: `0x0000_0000`



`Bit 31:0` **PADMUX**: The content of these registers can be used to multiplex pads when targeting an ASIC. The first register (0x1A10_4010) can be used to sets the mux (2 bit select) from pin 0 (bits [1:0]) to 15 (bits [31:30]). The second register (0x1A10_4014) can be used to sets the mux (2 bit select) from pin 16 (bits [1:0]) to 31 (bits [31:30]). The third register (0x1A10_4018) can be used to sets the mux (2 bit select) from pin 32 (bits [1:0]) to 47 (bits [31:30]). The forth register (0x1A10_401C) can be used to sets the mux (2 bit select) from pin 48 (bits [1:0]) to 63 (bits [31:30]).

5.3.5 PAD Configuration

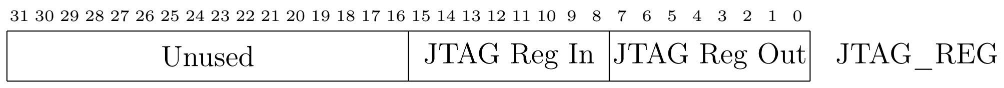
Address: `0x1A10_4020 - 0x1A10_405C`
Reset Value: `0x0000_0000`



`Bit 31:0` **PAD CFG0-15**: These 16 registers can be used for ASIC targets to configure pads, e.g. pull up, pull down values.

5.3.6 JTAG Register

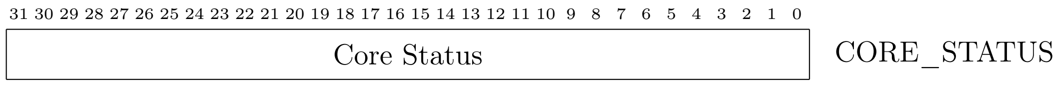
Address: `0x1A10_4074`
Reset Value: `0x0000_0000`



`Bit 31:0` **JTAG Register**: This register contains the value of the input from the JTAG and can be used to write 8bit in the JTAG output register for system-to-JTAG communications.

5.3.7 Core Status

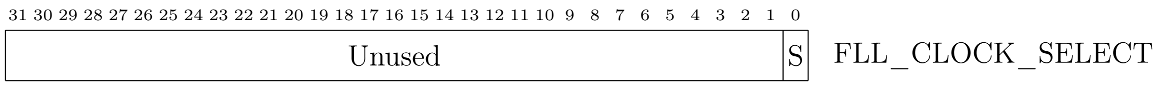
Address: `0x1A10_40A0 and 0x1A10_40C0`
Reset Value: `0x0000_0001`



`Bit 31:0` **Core Status**: These 2 registers contain the status of the system for testing/verification purposes like End Of Computation. The 0x1A10_40C0 register is read-only.

5.3.8 FLL Clock Select

Address: `0x1A10_40C8`
Reset Value: `0x0000_0000`



`Bit 31:0` **FLL Clock Select**: This register contains whether the system clock is coming from the FLL or the FLL is bypassed. It is a read-only register by the core but it can be written via JTAG.

5.4 Event/Interrupt Controller

PULPissimo features a lightweight event and interrupt controller which supports vectorized interrupts and events of up to 32 lines. It contains a FIFO of events from the peripherals or SW events. When an interrupt is ready and it is enabled (not masked), the unit sends the 5-bit ID to the core and the interrupt request line is raised up. If the core takes the interrupt, it replies with the ID of the interrupt taken and the acknowledge signal. The communication between the interrupt controller and the core is completly asynchronous. Note that the interrupt controller can change the interrupt ID anytime but it must rely on the ID sent by the core to know which interrupt has been taken. This is an important feature that covers the situation where a higher priority interrupt request prevent another one that has been already sent to the core. Depending on the core state and core interrupt enable, the interrupt can be accepted within a couple of clock cycles.

5.4.1 Mask

Address: `0x1A10_9000`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	MASK

`Bit 31:0` **MASK**: This register contains the MASK (interrupt enable) for each of the 32 interrupts or events. Writing to 0x1A10_9004 sets the bits of the MASK register selected. Writing to 0x1A10_9008 clears the bits of the MASK register selected.

5.4.2 Interrupt

Address: `0x1A10_900C`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	INT

`Bit 31:0` **INT**: This register contains the pending interrupts or events. Writing to 0x1A10_9010 sets the bits of the INT register selected. Writing to 0x1A10_9014 clears the bits of the INT register selected.

5.4.3 Int Ack

Address: `0x1A10_9018`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	ACK

`Bit 31:0` **ACK**: This register contains the ACK (interrupt enable) for each of the 32 interrupts or events. Writing to 0x1A10_901C sets the bits of the ACK register selected. Writing to 0x1A10_9020 clears the bits of the ACK register selected.

5.4.4 FIFO Content

Address: `0x1A10_9024`
Reset Value: `0x0000_0000`

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Fifo Data	FIFO_DATA

`Bit 31-0` **FIFO_DATA**: Fifo Content.
This is a read-only register that contain the first valid value of the FIFO.

5.5 SoC Event Generator

PULPissimo Events from peripherals and other sources can be forwarded to the fabric controller, cluster or (back) to certain peripherals, though for PULPissimo we don't have a cluster.

It is the SoC Event Generator's job to control which events are to be forwarded and where to. There are three set of masks available to do this:

FC Masks	Control which events are to be forwarded to the fabric controller
Cluster Masks	Control which events are to be forwarded to the cluster (disabled)
Peripheral Masks	Control which events are to be forwarded to peripherals

5.5.1 SoC Event Generator registers

Name	**Address**	**Size**	**Type**	**Access**	**Default**	**Description**
SW_EVENT	`0x1A106000`	32	Config	W	`0x00000000`	SoC software events trigger register
FC_MASK0	`0x1A106004`	32	Config	R/W	`0xFFFFFFFF`	Events 0-31 dispatch mask to FC
FC_MASK1	`0x1A106008`	32	Config	R/W	`0xFFFFFFFF`	Events 32-63 dispatch mask to FC
FC_MASK2	`0x1A10600C`	32	Config	R/W	`0xFFFFFFFF`	Events 64-95 dispatch mask to FC
FC_MASK3	`0x1A106010`	32	Config	R/W	`0xFFFFFFFF`	Events 96-127 dispatch mask to FC
FC_MASK4	`0x1A106014`	32	Config	R/W	`0xFFFFFFFF`	Events 128-159 dispatch mask to FC
FC_MASK5	`0x1A106018`	32	Config	R/W	`0xFFFFFFFF`	Events 160-191 dispatch mask to FC
FC_MASK6	`0x1A10601C`	32	Config	R/W	`0xFFFFFFFF`	Events 191-223 dispatch mask to FC
FC_MASK7	`0x1A106020`	32	Config	R/W	`0xFFFFFFFF`	Events 224-255 dispatch mask to FC

Table 5.3: SoC Event Generator register table part 1

Name	**Address**	**Size**	**Type**	**Access**	**Default**	**Description**
PR_MASK0	`0x1A106044`	32	Config	R/W	`0xFFFFFFFF`	Events 0-31 dispatch mask to peripherals
PR_MASK1	`0x1A106048`	32	Config	R/W	`0xFFFFFFFF`	Events 32-63 dispatch mask to peripherals
PR_MASK2	`0x1A10604C`	32	Config	R/W	`0xFFFFFFFF`	Events 64-95 dispatch mask to peripherals
PR_MASK3	`0x1A106050`	32	Config	R/W	`0xFFFFFFFF`	Events 96-127 dispatch mask to peripherals
PR_MASK4	`0x1A106054`	32	Config	R/W	`0xFFFFFFFF`	Events 128-159 dispatch mask to peripherals
PR_MASK5	`0x1A106058`	32	Config	R/W	`0xFFFFFFFF`	Events 160-191 dispatch mask to peripherals
PR_MASK6	`0x1A10605C`	32	Config	R/W	`0xFFFFFFFF`	Events 191-223 dispatch mask to peripherals
PR_MASK7	`0x1A106060`	32	Config	R/W	`0xFFFFFFFF`	Events 224-255 dispatch mask to peripherals
ERR0	`0x1A106064`	32	Status	R	`0x00000000`	Events 0-31 event queue overflow
ERR1	`0x1A106068`	32	Status	R	`0x00000000`	Events 32-63 event queue overflow
ERR2	`0x1A10606C`	32	Status	R	`0x00000000`	Events 64-95 event queue overflow
ERR3	`0x1A106070`	32	Status	R	`0x00000000`	Events 96-127 event queue overflow
ERR4	`0x1A106074`	32	Status	R	`0x00000000`	Events 128-159 event queue overflow
ERR5	`0x1A106078`	32	Status	R	`0x00000000`	Events 160-191 event queue overflow
ERR6	`0x1A10607C`	32	Status	R	`0x00000000`	Events 191-223 event queue overflow
ERR7	`0x1A106080`	32	Status	R	`0xFFFFFFFF`	Events 224-255 event queue overflow
TIMER_LO	`0x1A106084`	32	Status	R/W	`0xFFFFFFFF`	Trigger Timer LO of APB Timer with event
TIMER_HI	`0x1A106088`	32	Status	R/W	`0xFFFFFFFF`	Trigger Timer HI of APB Timer with event

Table 5.4: SoC Event Generator register table part 2

5.5.2 SW_EVENT

Address: `0x1A10_6000`
Reset Value: `0x0000_0000`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	EVENT							

`Bit 7-0` **EVENT** *(W)* Writing a one-hot value into EVENT triggers a SoC software event. 8 software events are available.

5.5.3 FC_MASKX, $X = 0...7$

Address: `0x1A10_6004 + 0x4 *` X
Reset Value: `0xFFFF_FFFF`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FC_MASK															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FC_MASK															

`Bit 31-0` **FC_MASK** *(R/W)* Event Mask to enable/disable event dispatch to FC interrupt controller.

- Setting $bit[i]$ to $0b1$ disables dispatching $event[32 * X + i]$ to FC interrupt controller.
- Setting $bit[i]$ to $0b0$ enables dispatching $event[32 * X + i]$ to FC interrupt controller.

5.5.4 PR_MASKX, $X = 0...7$

Address: `0x1A10_6044 + 0x4 *` X
Reset Value: `0xFFFF_FFFF`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PR_MASK															

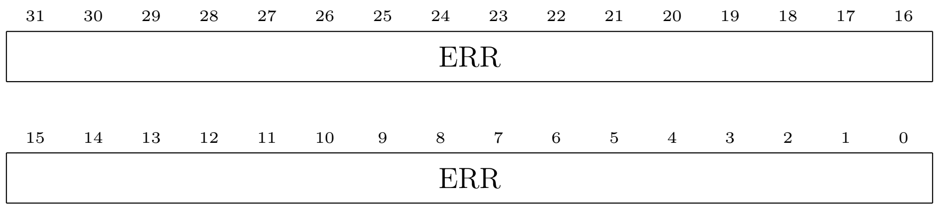
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR_MASK															

`Bit 31-0` **PR_MASK** *(R/W)* Event Mask to enable/disable event dispatch to peripherals.

- Setting $bit[i]$ to $0b1$ disables dispatching $event[32 * X + i]$ to peripherals.
- Setting $bit[i]$ to $0b0$ enables dispatching $event[32 * X + i]$ to peripherals.

5.5.5 ERRX, $X = 0...7$

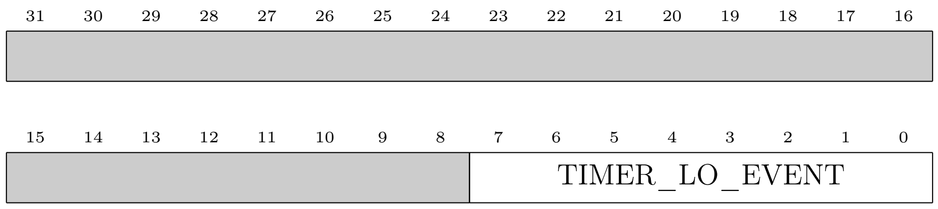
Address: `0x1A10_6064 + 0x4 *` X
Reset Value: `0x0000_0000`



`Bit 31-0` **ERR** *(R/W)* Event queue overflow. Clear after read. Reading $0b1$ at $ERR[i]$ means the event queue of event with id $32 * X + i$ overflowed.

5.5.6 TIMER_LO

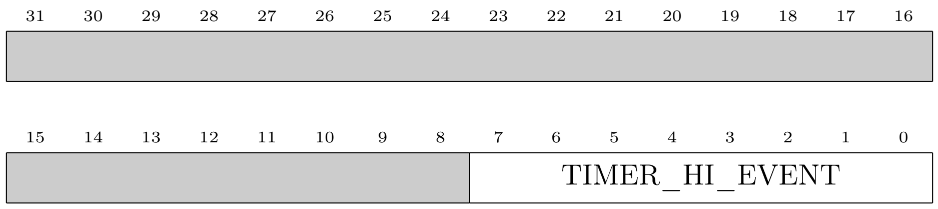
Address: `0x1A10_6084`
Reset Value: `0x0000_0000`



`Bit 7-0` **TIMER_LO_EVENT** *(R/W)* Trigger and start APB Timer LO by the event with id that equals TIMER_LO_EVENT

5.5.7 TIMER_HI

Address: `0x1A10_6088`
Reset Value: `0x0000_0000`



`Bit 7-0` **TIMER_HI_EVENT** *(R/W)* Trigger and start APB Timer HI by the event with id that equals TIMER_HI_EVENT