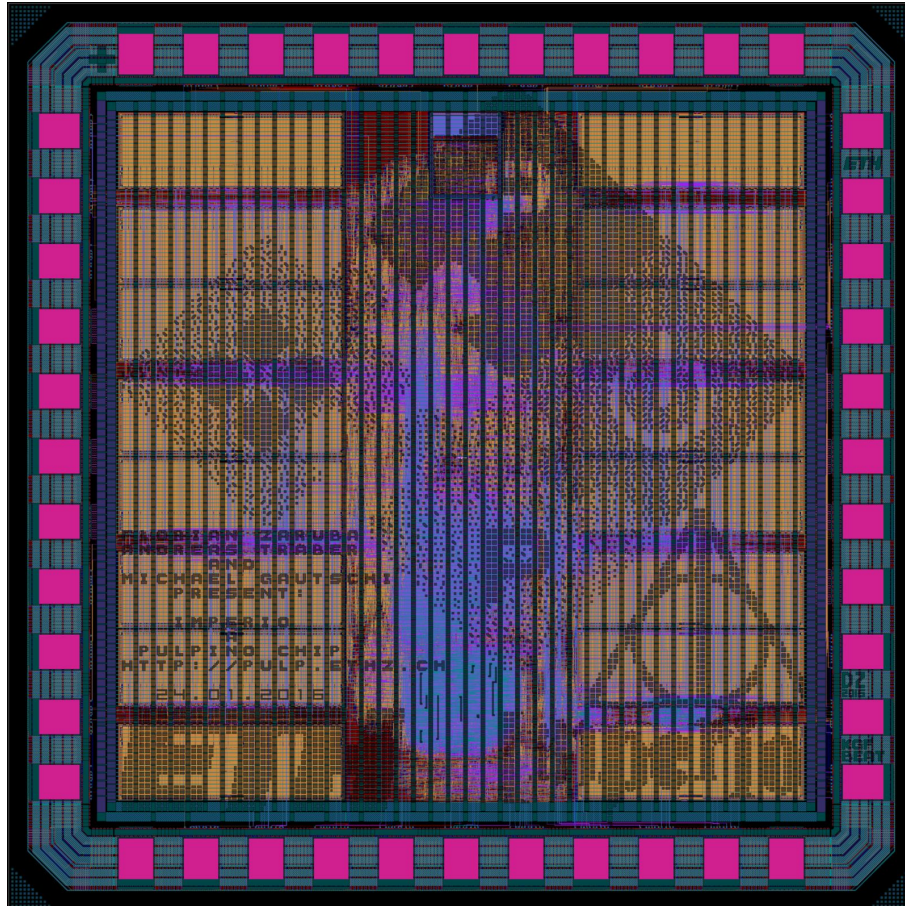


PULPino: Datasheet



Imperio: The first PULPino ASIC

Andreas Traber, Michael Gautschi
{atraber, gautschi}@iis.ee.ethz.ch

November 17, 2016

Contents

1	Overview	4
2	Memory Map	5
2.1	Interrupt Vector Table	7
3	CPU Core	8
4	Advanced Debug Unit	9
5	Peripherals	10
5.1	UART	10
5.2	GPIO	11
5.2.1	PADDIR (Pad Direction)	11
5.2.2	PADIN (Input Values)	11
5.2.3	PADOUT (Output Values)	11
5.2.4	INTEN (Interrupt Enable)	12
5.2.5	INTTYPE0 (Interrupt Type 0)	12
5.2.6	INTTYPE1 (Interrupt Type 1)	12
5.2.7	INTSTATUS (Interrupt Status)	13
5.2.8	PADCFG0-7 (Pad Configuration Registers 0-7)	13
5.3	SPI Master	14
5.3.1	STATUS (Status Register)	14
5.3.2	CLKDIV (Clock Divider)	15
5.3.3	SPICMD (SPI Command)	15
5.3.4	SPIADR (SPI Address)	15
5.3.5	SPILEN (SPI Transfer Length)	16
5.3.6	SPIDUM (SPI Dummy Cycles)	16
5.3.7	TXFIFO (SPI Transmit FIFO)	16
5.3.8	RXFIFO (SPI Receive FIFO)	17
5.3.9	INTCFG (Interrupt Configuration)	17
5.4	I ² C	18
5.4.1	CPR (Clock Prescale Register)	18
5.4.2	CTRL (Control Register)	18
5.4.3	TX (Transmit Register)	19
5.4.4	RX (Receive Register)	19
5.4.5	CMD (Command Register)	19
5.4.6	STATUS (Status Register)	20
5.5	Timer	21
5.5.1	TIMER (Current Timer Value)	21
5.5.2	CTRL (Timer Control)	21

5.5.3	CMP (Timer Compare)	21
5.6	Event Unit	22
5.6.1	IER (Interrupt Enable)	22
5.6.2	IPR (Interrupt Pending)	23
5.6.3	ISP (Interrupt Set Pending)	23
5.6.4	ICP (Interrupt Clear Pending)	23
5.6.5	EER (Event Enable)	24
5.6.6	EPR (Event Pending)	24
5.6.7	ESP (Event Set Pending)	24
5.6.8	ECP (Event Clear Pending)	24
5.6.9	SCR (Sleep Control)	25
5.6.10	SSR (Sleep Status)	25
5.7	SoC Control	26
5.7.1	PAD Mux	26
5.7.2	CLK Gate	26
5.7.3	Boot Address	26
5.7.4	Info	27
5.7.5	Status	27
5.7.6	PAD Configuration	27
5.8	Debug Port	28
6	SPI Slave	29

1 Overview

PULPINO is a single-core System-on-a-Chip built for the OpenRISC OR10N core and the RISC-V RI5CY core. PULPINO reuses most components from its bigger brother PULP. It uses separate single-port data and instruction RAMs. It includes a boot ROM that contains a boot loader that can load a program via SPI from an external flash device.

Figure 1.1 shows a block diagram of the SoC. The SoC uses a AXI as its main interconnect with a bridge to APB for simple peripherals. Both the AXI and the APB buses feature 32 bit wide data channels. For debugging purposes the SoC includes an advanced debug unit which enables access to core registers, the two RAMs and memory-mapped IO via JTAG. Both RAMs are connected to the AXI bus via bus adapters.

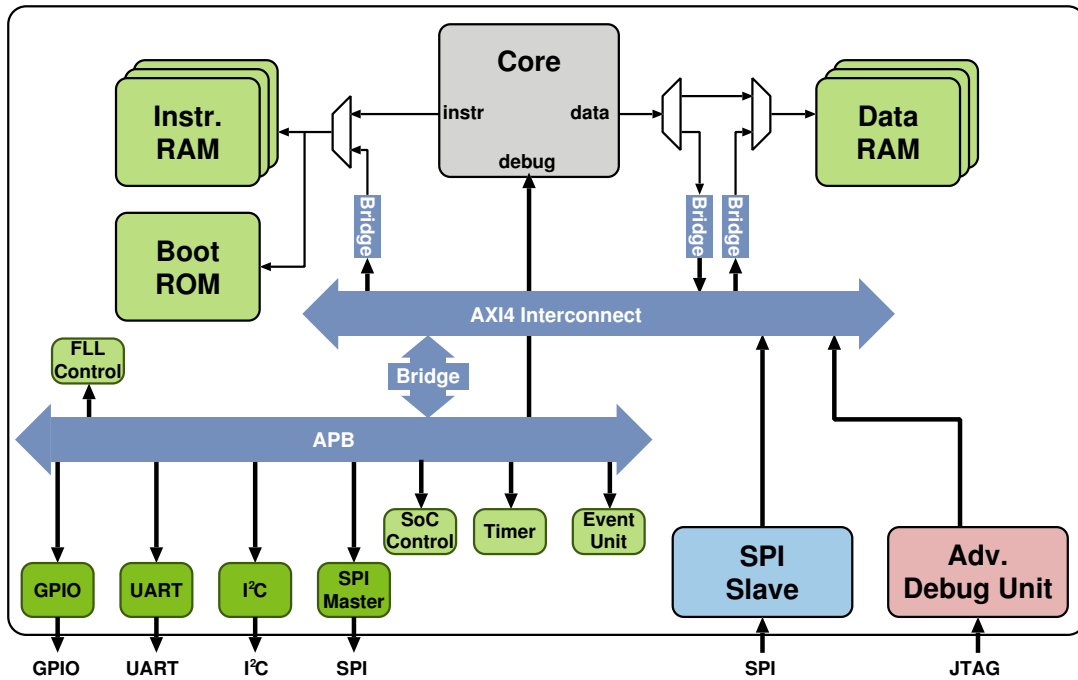


Figure 1.1: PULPINO Overview.

PULPINO is mainly targeted at RTL simulation and ASICs, although there is also an FPGA version. The FPGA version is not specifically optimal in terms of performance as we mainly use it as an emulation platform rather than a standalone platform.

2 Memory Map

Figure 2.1 shows the default memory-map of PULPino assuming 32 kB of data and instruction RAM. This can be changed in the PULPino top-level SystemVerilog file.

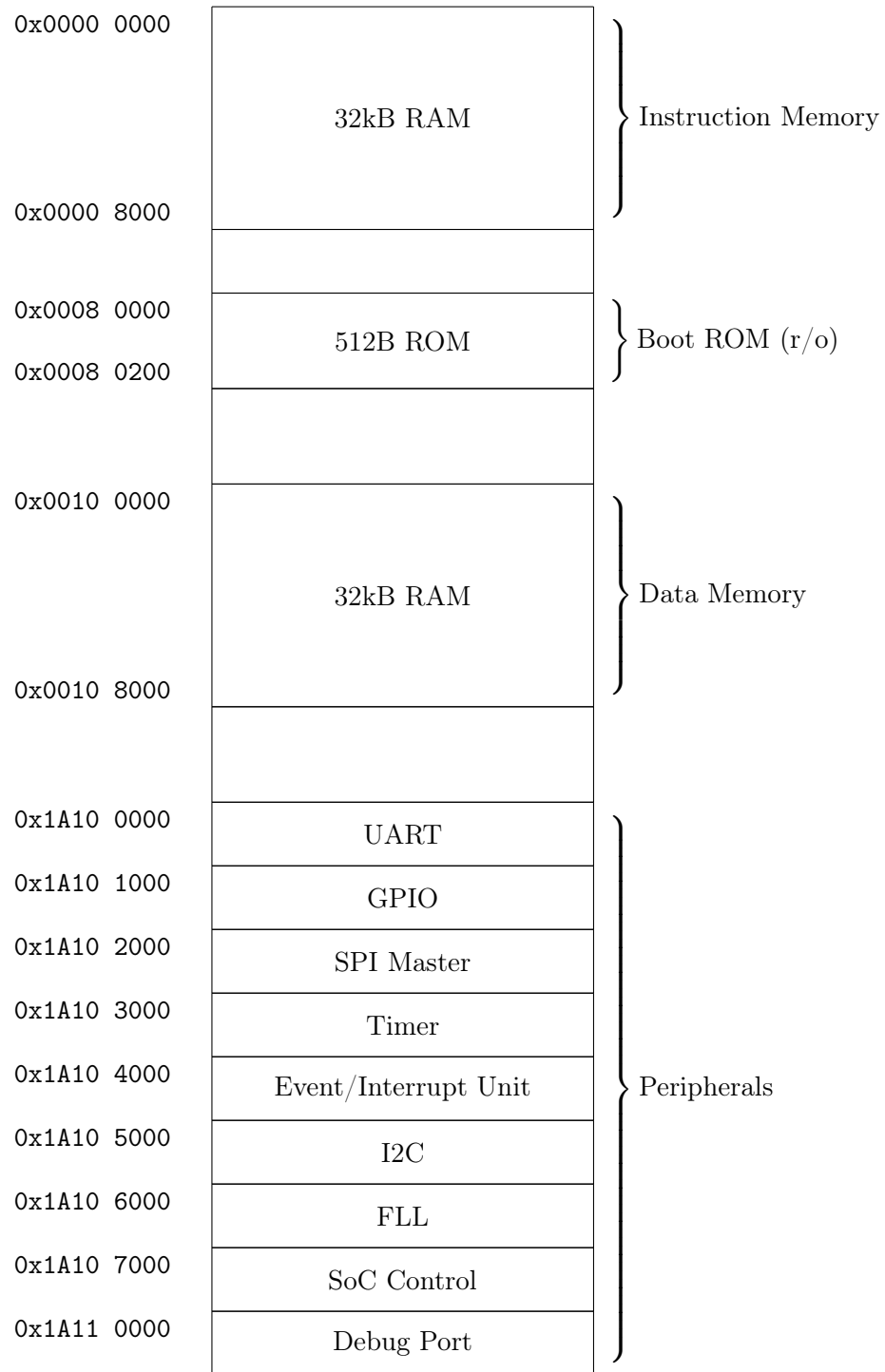


Figure 2.1: PULPINO memory map.

2.1 Interrupt Vector Table

The IVT of PULPino follows the definition given for the RI5CY core.

Table 2.1: Interrupt/exception offset vector table

Description	Address
Reserved - Not used	0x00000000 - 0x00000058
Interrupt Line 23: I2C IRQ	0x0000005C
Interrupt Line 24: UART IRQ	0x00000060
Interrupt Line 25: GPIO IRQ	0x00000064
Interrupt Line 26: SPI Master 0	0x00000068
Interrupt Line 27: SPI Master 1	0x0000006C
Interrupt Line 28: Timer A Overflow	0x00000070
Interrupt Line 29: Timer A Output Cmp	0x00000074
Interrupt Line 30: Timer B Overflow	0x00000078
Interrupt Line 31: Timer B Output Cmp	0x0000007C
RESET	0x00000080
Illegal Instruction Exception	0x00000084
ECALL Instruction	0x00000088
Invalid Memory Access	0x0000008C

The entries in the IVT have to be either one 32-bit, or one or two 16-bit instructions that tell the core how to handle the interrupt. In most cases, this means the core will execute a jump to a real interrupt handler.

3 CPU Core

PULPINO supports both the OpenRISC OR10N core and the RISC-V RI5CY. The two cores have the same external interfaces and are thus plug-compatible.

The core use a very simple data and instruction interface to talk to data and instruction memories. To interface with AXI, a core2axi protocol converter is instantiated in PULPINO.

For debugging purposes, all core registers have been memory mapped which allows to them to be accessed over the AXI bus. 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.

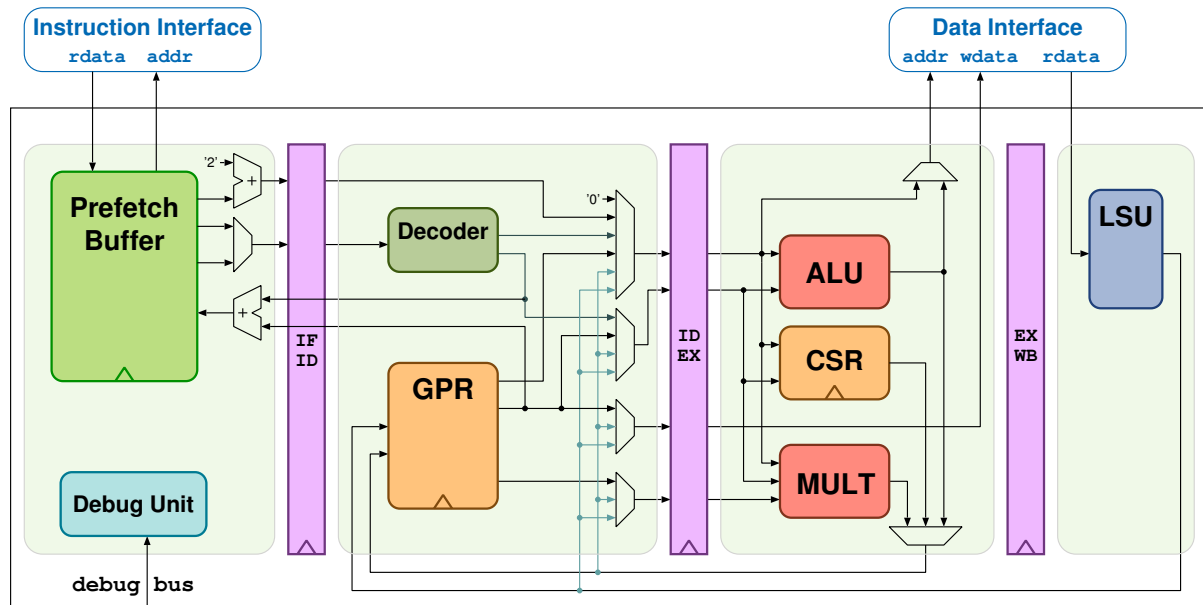


Figure 3.1: RISC-V core overview

Additional information is available in the core datasheet, see [ips/riscv/docs/datasheet](#).

4 Advanced Debug Unit

The advanced debug unit has an AXI master interface to access peripherals and memories. In contrast to PulpinoV1 the adv. debug unit does no longer feature a specialized debug interface to readout all core registers.

All core registers are now memory mapped which means that they can be read over the AXI interface. Hence, debugging is not only possible over JTAG, but also SPI or any other interface.

The JTAG signals are connected to pins of the SoC.

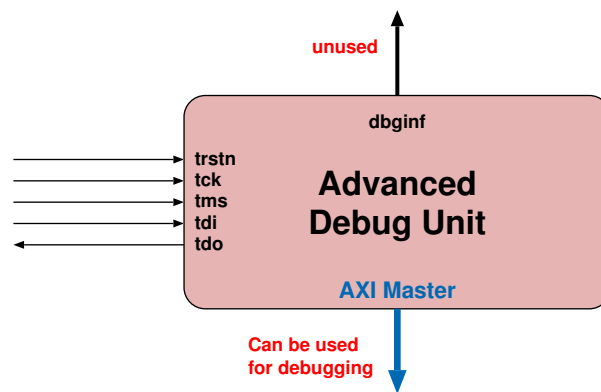


Figure 4.1: Advanced Debug Unit.

For more details please take a look at the documentation of the advanced debug unit.

5 Peripherals

All peripherals in PULPINO are connected to the APB bus, except the SPI slave which is a very special peripheral and not intended to be used from the core itself. See Chapter 6 for details on the SPI slave.

5.1 UART

The UART used in this system is compatible with a 16750. It features all the typical UART signals, see Table 5.1, plus some additional signals defined by the 16750.

Table 5.1: External UART Signals

Signal	Direction	Description
uart_tx	output	Transmit Data
uart_rx	input	Receive Data
uart_rts	output	Request to Send
uart_cts	input	Clear to send
uart_dtr	output	Data Terminal Ready
uart_dsr	input	Data Set Ready

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 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.

5.3 SPI Master

Table 5.3: SPI Signals

Signal	Direction	Description
spi_clk	output	Master Clock
spi_csn0	output	Chip Select 0
spi_csn1	output	Chip Select 1
spi_csn2	output	Chip Select 2
spi_csn3	output	Chip Select 3
spi_mode[1:0]	output	SPI Mode
spi_sdo0	output	Output Line 0
spi_sdo1	output	Output Line 1
spi_sdo2	output	Output Line 2
spi_sdo3	output	Output Line 3
spi_sdi0	input	Input Line 0
spi_sdi1	input	Input Line 1
spi_sdi2	input	Input Line 2
spi_sdi3	input	Input Line 3
events_o[1:0]	output	Event/Interrupt

5.3.1 STATUS (Status Register)

Address: 0x1A10_2000

Reset Value: 0x0000_0000



Bit 11:8 **CS**: Chip Select.

Specify the chip select signal that should be used for the next transfer.

Bit 4 **SRST**: Software Reset.

Clear FIFOs and abort active transfers.

Bit 3 **QWR**: Quad Write Command.

Perform a write using Quad SPI mode.

Bit 2 **QRD**: Quad Read Command.

Perform a read using Quad SPI mode.

Bit 1 **WR**: Write Command.

Perform a write using standard SPI mode.

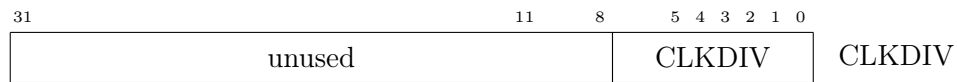
Bit 0 **RD**: Read Command.

Perform a read using standard SPI mode.

5.3.2 CLKDIV (Clock Divider)

Address: 0x1A10_2004

Reset Value: 0x0000_0000



Bit 7:0 **CLKDIV**: Clock Divider.

Clock divider value used to divide the SoC clock for the SPI transfers. This register should not be modified while a transfer is in progress.

5.3.3 SPICMD (SPI Command)

Address: 0x1A10_2008

Reset Value: 0x0000_0000



Bit 31:0 **SPICMD**: SPI Command.

When performing a read or write transfer the SPI command is sent first before any data is read or written. The length of the SPI command can be controlled with the **SPILEN** register.

5.3.4 SPIADR (SPI Address)

Address: 0x1A10_200C

Reset Value: 0x0000_0000



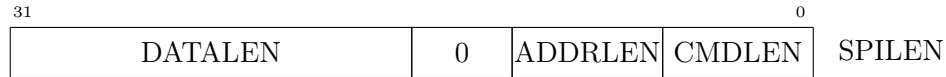
Bit 31:0 **SPIADR**: SPI Address.

When performing a read or write transfer the SPI command is sent first before any data is read or written, after this the SPI address is sent. The length of the SPI address can be controlled with the **SPILEN** register.

5.3.5 SPILEN (SPI Transfer Length)

Address: 0x1A10_2010

Reset Value: 0x0000_0000



Bit 31:16 **DATALEN**: SPI Data Length.

The number of bits read or written. Note that first the SPI command and address are written to an SPI slave device.

Bit 13:8 **ADDRLEN**: SPI Address Length.

The number of bits of the SPI address that should be sent.

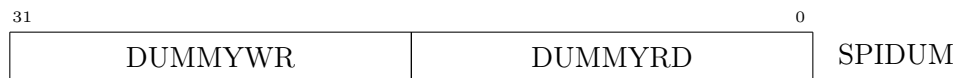
Bit 5:0 **CMDLEN**: SPI Command Length.

The number of bits of the SPI command that should be sent.

5.3.6 SPIDUM (SPI Dummy Cycles)

Address: 0x1A10_2014

Reset Value: 0x0000_0000



Bit 31:16 **DUMMYWR**: Write Dummy Cycles.

Dummy cycles (nothing being written or read) between sending the SPI command + SPI address and writing the data.

Bit 15:0 **DUMMYRD**: Read Dummy Cycles.

Dummy cycles (nothing being written or read) between sending the SPI command + SPI address and reading the data.

5.3.7 TXFIFO (SPI Transmit FIFO)

Address: 0x1A10_2018

Reset Value: 0x0000_0000



Bit 31:0 **TX**: Transmit Data.

Write data into the FIFO.

5.3.8 RXFIFO (SPI Receive FIFO)

Address: 0x1A10_2020

Reset Value: 0x0000_0000

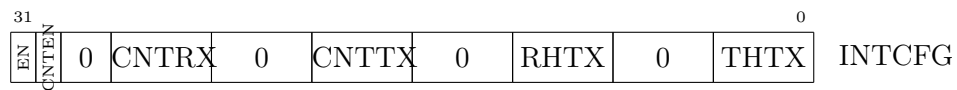


Bit 31:0 **RX**: Receive Data.
Read data from the FIFO.

5.3.9 INTCFG (Interrupt Configuration)

Address: 0x1A10_2024

Reset Value: 0x0000_0000



Bit 31 **EN**: Interrupt Enable.
Enable interrupts

5.4 I²C

I²C is an open-drain signaling protocol, meaning that the pad output driver will be switched on and off, while always driving a low value when enabled. Logic high values are achieved by using a pull-up resistor on the **SDA** and **SCL** lines.

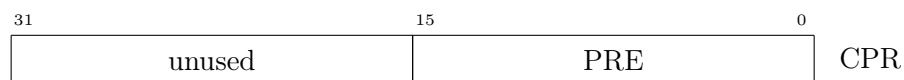
Table 5.4: I²C Signals

Signal	Direction	Description
scl_pad_i	input	SCL Input
scl_pad_o	output	SCL Output (always 0)
scl_padoen_o	output	SCL Pad Direction
sda_pad_i	input	SDA Input
sda_pad_o	output	SDA Output (always 0)
sda_padoen_o	output	SDA Pad Direction
interrupt_o	output	Event/Interrupt

5.4.1 CPR (Clock Prescale Register)

Address: 0x1A10_5000

Reset Value: 0x0000_0000



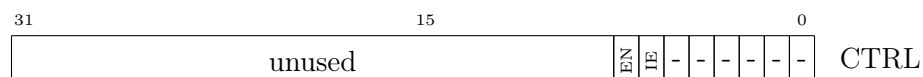
Bit 15:0 **PRE**: Prescaler.

Sets the clock prescaler by the value in PRE to achieve the desired I²C clock by dividing the current system clock by the given factor.

5.4.2 CTRL (Control Register)

Address: 0x1A10_5004

Reset Value: 0x0000_0000



Bit 7 **EN**: Enable.

Enable the I²C peripheral.

Bit 6 **IE**: Interrupt enable.

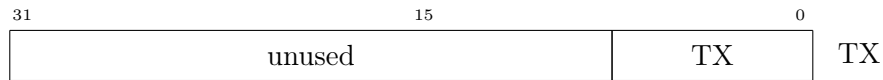
Enable interrupts.

Bit 5:0 **Reserved:** Set to 0.

5.4.3 TX (Transmit Register)

Address: 0x1A10_5008

Reset Value: 0x0000_0000

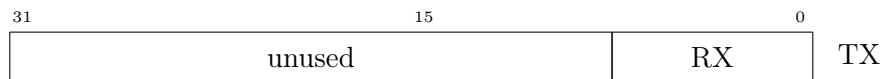


Bit 7:0 **TX**: Transmit Register

5.4.4 RX (Receive Register)

Address: 0x1A10_500C

Reset Value: 0x0000_0000

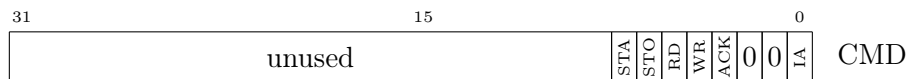


Bit 7:0 **RX**: Receive Register

5.4.5 CMD (Command Register)

Address: 0x1A10_5010

Reset Value: 0x0000_0000



Bit 7 **STA**: Send start bit.

Bit 6 **STO**: Send stop bit.

Bit 5 **RD**: Read from bus.

Bit 4 **WR**: Write to bus.

Bit 3 **ACK**: Acknowledge received data.

Bit 2:1 **Reserved**: Set to 0.

Bit 0 **IA**: Interrupt Acknowledge.

Set to one to acknowledge interrupt. Cleared when transmission is done or arbitration is lost.

5.4.6 STATUS (Status Register)

Address: 0x1A10_5014

Reset Value: 0x0000_0000



Bit 7 **RXA**: Acknowledge from sent data.

Bit 6 **BUS**: Bus is busy.

Bit 5 **AL**: Arbitration lost.

Bit 4:2 **Reserved:** Set to 0.

Bit 1 **TIP**: Transfer in progress.

Bit 0 **IRQ**: Interrupt received.

This flag is always set when transmission has finished or bus arbitration was lost, regardless of whether interrupts are enabled or not. This flag can possibly be polled and is cleared by writing 1 to the IA command register.

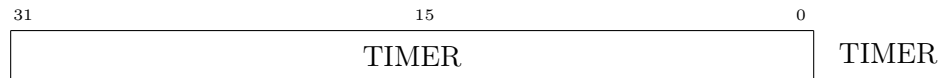
5.5 Timer

The timer unit has 2 timers per default. This can be overwritten by a parameter when instantiating the timer.

5.5.1 TIMER (Current Timer Value)

Address: 0x1A10_3070

Reset Value: 0x0000_0000



Bit 31:0 **TIMER**: Current Timer Value.

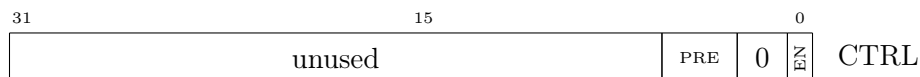
Current value of the timer. There is an internal prescaler per timer that allows to specify the interval in which the timer will be increased. The prescaler is controlled by CTRL.

When TIMER reaches FFFF_FFFF an interrupt will be raised.

5.5.2 CTRL (Timer Control)

Address: 0x1A10_3074

Reset Value: 0x0000_0000



Bit 5:3 **PRE**: Prescaler value.

Bit 0 **EN**: Enable the timer.

5.5.3 CMP (Timer Compare)

Address: 0x1A10_3078

Reset Value: 0x0000_0000



Bit 31:0 **CMP**: Timer Compare.

An interrupt will be raised when TIMER reaches CMP

5.6 Event Unit

PULPINO features a lightweight event and interrupt unit which supports vectorized interrupts of up to 32 lines and event triggering of up to 32 input lines. The interrupt and event lines are separately masked and buffered, see Figure 5.1.

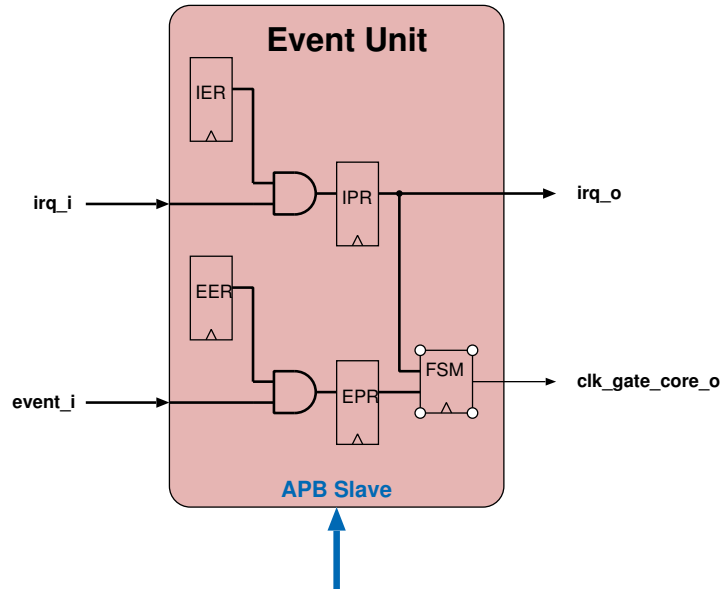


Figure 5.1: Event Unit.

The current assignment of event and interrupt lines is given in Figure 5.2. Note that `irq_i` and `event_i` are bound together.

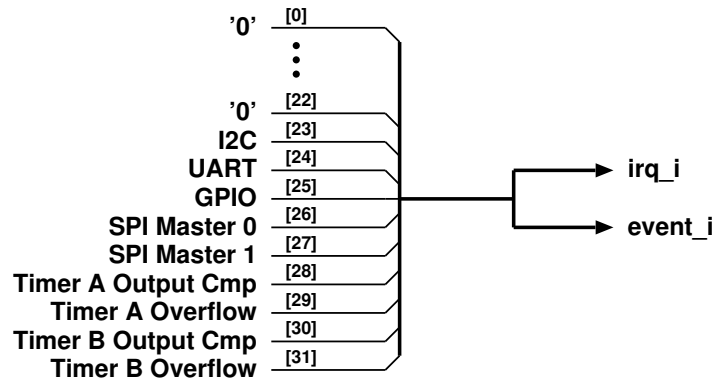


Figure 5.2: Event Lines.

5.6.1 IER (Interrupt Enable)

Address: 0x1A10_4000

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		
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	IER

Bit 31:0 **IER**: Interrupt Enable.
Enable interrupts per line.

5.6.2 IPR (Interrupt Pending)

Address: 0x1A10_4004

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	IPR

Bit 31:0 **IPR**: Interrupt Pending.
Write/read pending interrupts per line.

5.6.3 ISP (Interrupt Set Pending)

Address: 0x1A10_4008

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	ISP

Bit 31:0 **ISP**: Interrupt Set Pending.
Set interrupt pending register per line. By setting a bit here, an interrupt will be triggered on the selected line(s).

5.6.4 ICP (Interrupt Clear Pending)

Address: 0x1A10_400C

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	
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	ICP

Bit 31:0 **ICP**: Interrupt Clear Pending.
Clear pending interrupt. By setting a bit here, a pending interrupt will be cleared.

5.6.5 EER (Event Enable)

Address: 0x1A10_4010

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	
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	EER

Bit 31:0 **EER**: Event Enable.
Enable events per line.

5.6.6 EPR (Event Pending)

Address: 0x1A10_4014

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	EPR

Bit 31:0 **EPR**: Event Pending.
Write/read pending events per line.

5.6.7 ESP (Event Set Pending)

Address: 0x1A10_4018

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	ESP

Bit 31:0 **ESP**: Event Set Pending.
Set event pending register per line. By setting a bit here, an event will be set on the selected line(s).

5.6.8 ECP (Event Clear Pending)

Address: 0x1A10_401C

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	
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	ECP

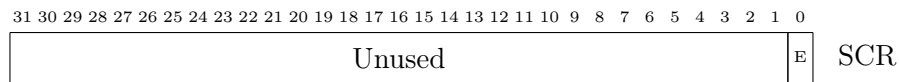
Bit 31:0 **ECP**: Event Clear Pending.

Clear pending event. By setting a bit here, a pending event will be cleared.

5.6.9 SCR (Sleep Control)

Address: 0x1A10_4020

Reset Value: 0x0000_0000



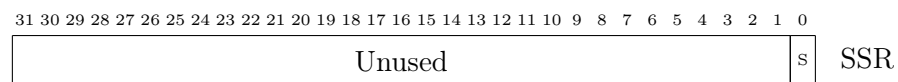
Bit 0 **E**: Sleep Enabled.

Put the core to sleep. The core will be woken up again when there is an interrupt or event.

5.6.10 SSR (Sleep Status)

Address: 0x1A10_4024

Reset Value: 0x0000_0000



Bit 0 **S**: Sleep Status.

Set if the core is currently asleep and has its clock gated.

5.7 SoC Control

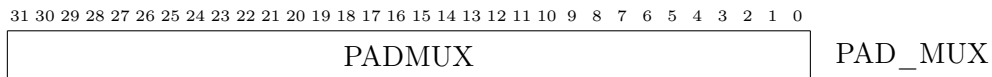
PULPINO features a small and simple APB peripheral which provides information about the platform and provides the means for pad muxing on the ASIC.

The following registers can be accessed.

5.7.1 PAD Mux

Address: 0x1A10_7000

Reset Value: 0x0000_0000

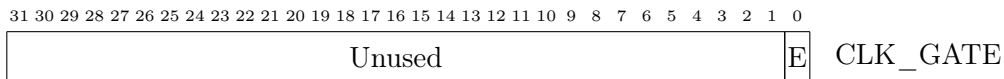


Bit 31:0 **PADMUX**: The content of this register can be used to multiplex pads when targeting an ASIC.

5.7.2 CLK Gate

Address: 0x1A10_7004

Reset Value: 0x0000_0001

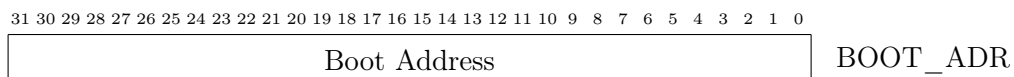


Bit 31:0 **CLK GATE**: This register contains the value of the clock gate enable signal (E) used to clock gate the core. It is active high.

5.7.3 Boot Address

Address: 0x1A10_7008

Reset Value: 0x0000_8000

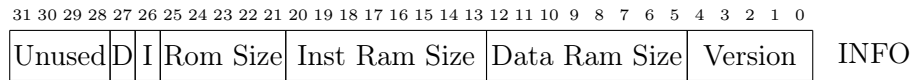


Bit 31:0 **Boot Address**: This register holds the boot address. It is possible to boot from a ROM, or directly from the instruction memory.

5.7.4 Info

Address: 0x1A10_7010

Reset Value: 0x0000_8082

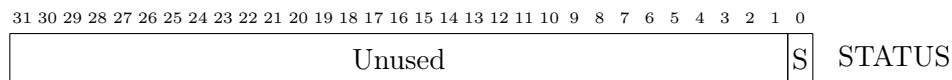


Bit 31:0 Info Register: This register holds information about the PULPino architecture. Version contains the pulpino version. The flags D and I report if there is a data/instruction cache present. Rom Size defines the size of the boot ROM. Finally, Inst. Ram Size and Data Ram Size define the size of the RAMs in multiples of 8 kB.

5.7.5 Status

Address: 0x1A10_7014

Reset Value: 0x0000_0001

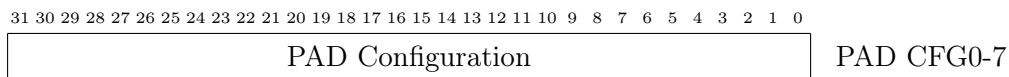


Bit 31:0 Status Register: The status register bit S can be used to hold the final result of a test for verification purposes.

5.7.6 PAD Configuration

Address: 0x1A10_7020 - 0x1A10_703C

Reset Value: 0x0000_0000



Bit 31:0 PAD CFG0-7: These 8 registers can be used for ASIC targets to configure pads, e.g. pull up, pull down values.

5.8 Debug Port

This block contains a apb2per bridge which allows to convert from APB commands to the debug bus. The debug bus directly connects the debug unit of the core to the APB bus and allows for memory mapped debugging. Since all core registers are memory mapped, it is possible to debug the core with a memory interface, rather than a highly specialized interface.

The debug bus signals are described as follows:

Table 5.5: Debug Bus Signals

Signal	Direction	Description
dbg_req	Output	Request signal
dbg_addr	Output	Address
dbg_we	Output	Write enable
dbg_wdata[31:0]	Output	Data to write
dbg_gnt	Input	Grant
dbg_rvalid	Input	Response valid
dbg_rdata[31:0]	Input	Read data

The protocol is very similar to the core data, and instruction interfaces. To read or write, the Master has to raise the request with the address, the write enable, and the data to write. As soon as the Slave sets the grant to 1, the transfer is granted. In case of a write operation the rvalid signal rises to one to inform the Master that the transaction was successful. In case of a read operation, the Slave returns the requested data on the rdata line, exactly one cycle after the grant.

6 SPI Slave

The SPI slave is an active peripheral in the sense that it receives/sends data without the assistance of the core. Its intended purpose is to function as an external interface through which a user of the system can access the internal memories from outside. This mechanism can be used to pre-load programs into the memories, start the system, wait for an acknowledgment that the program has terminated and then examine the results.

The SPI slave has an AXI master through which it can access all peripherals and memories. From the outside it can be accessed by sending SPI/QSPI commands to it. QSPI is an extension to SPI that uses four data lanes instead of one and is thus four times as fast as the standard SPI.

If pins on an ASIC are scarce, it is possible to just connect the standard SPI interface to pins and accept the performance loss in transfers from/to the SoC compared to the QSPI mode. The signals `spi_sdi1-3` and `spi_sdo1-3` are exclusively used for QSPI mode.

Table 6.1: SPI Slave Signals

Signal	Direction	Description
<code>spi_sclk</code>	input	Slave Clock
<code>spi_cs</code>	input	Chip Select
<code>spi_mode[1:0]</code>	output	SPI Mode
<code>spi_sdi0</code>	input	Input Line 0
<code>spi_sdi1</code>	input	Input Line 1
<code>spi_sdi2</code>	input	Input Line 2
<code>spi_sdi3</code>	input	Input Line 3
<code>spi_sdo0</code>	output	Output Line 0
<code>spi_sdo1</code>	output	Output Line 1
<code>spi_sdo2</code>	output	Output Line 2
<code>spi_sdo3</code>	output	Output Line 3