Datasheet

CC-GPIO-AXI Documentation

1.1

Scope

This document describes the CC-GPIO-AXI IP core. Module features and configuration registers are described. The document contains integration guide that covers synthesis options and instantiation example for easy implementation in customer's environment.

Contents

1. GPIO Controller

1.1 Functionality

- individual configuration of each GPIO pin,
- synchronous edge detection with interrupt signaling,
- two configurable interrupts,
- configurable input hysteresis,
- configurable output driver:
  - push-pull,
  - pull-up, pull-down,
  - bus keeper,
  - inverted,
- multiple GPIO pins configuration in one operation,
- value change (set/clear/toggle) of output pins in one operation,
- direction change (input/output) in one operation,
- alternative functions support.

1.2 Overview

The GPIO controller is a user-programmable general purpose I/O controller. Each of GPIO pins can be configured to work either as input or output. Direction, drive strength, drive slew rate, input hysteresis and pull-up and pull-down configuration can be set individually for each pin. GPIO controller allows multiple pins to be configured in one operation. Each of GPIO pins can be configured to detect and signal edge or level sensitive interrupt.

1.3 Block Diagram

Figure 1.1 presents the block diagram of the GPIO controller.

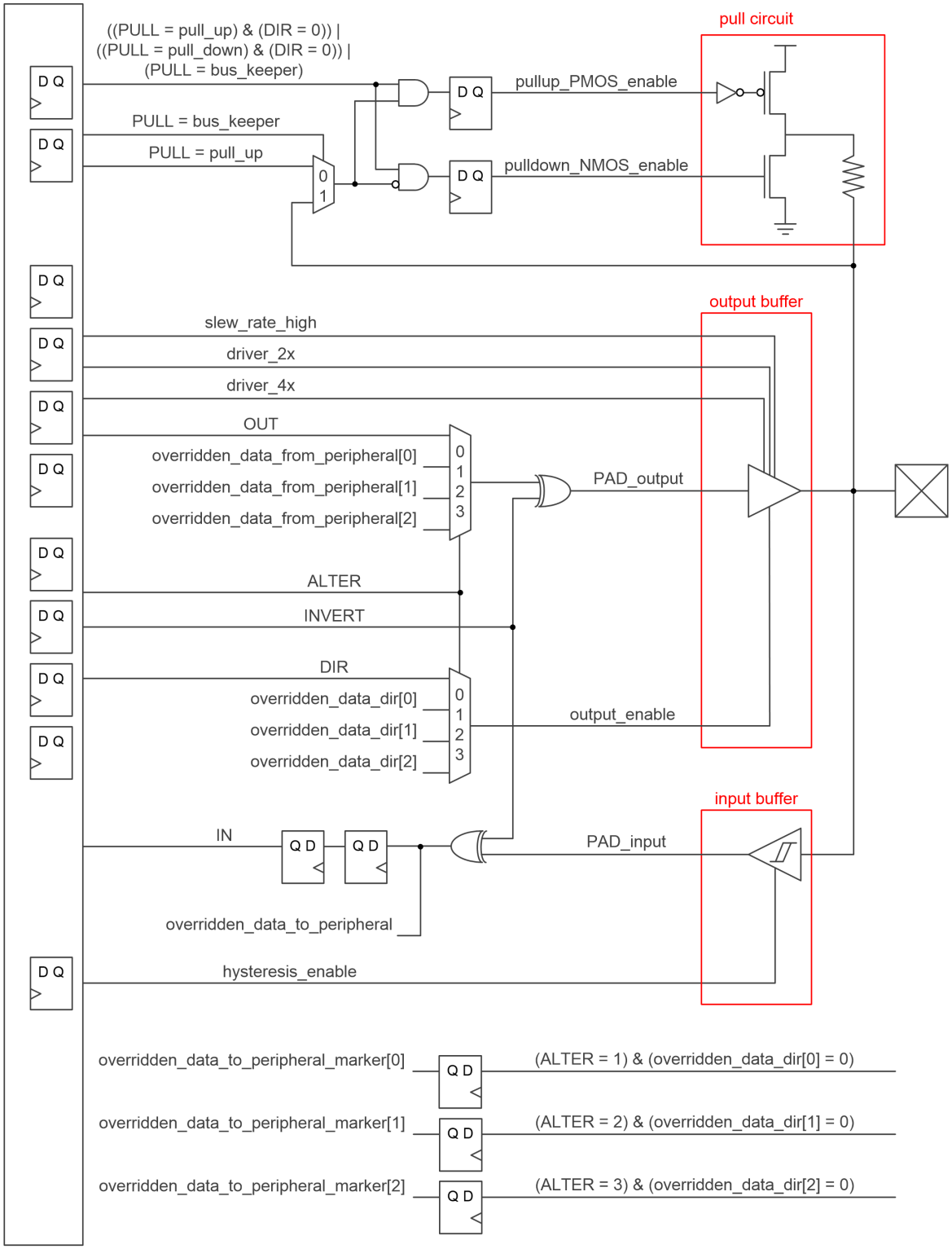


Figure 1.1. Block diagram of single GPIO pin.

1.4 Pull Configuration

Each of GPIO pins is controlled using user software. GPIO controller has a dedicated register to set pin direction (DIR - Data Direction 1.6.2) and output value (OUT - Data Output Value 1.6.6) used to setup the output driver. Writing “1” to the DIRn bit sets the n-th pin as an output, whereas writing “0” to the DIRn bit sets the n-th pin as an input. Input data register (IN 1.6.10)) is used to read value directly on the GPIO PAD in both input or output configuration. The INVERT (1.6.16), PULL (1.6.21), SLEW_RATE (1.6.17), DRIVER (1.6.23) and HYST (1.6.18) registers are used for additional configuration of each GPIO pin. The INVERT (1.6.16) register is used to negate pin in both input and output modes. The PULL (1.6.21) register is used to configure GPIO pin’s pull-up and pull-down. The SLEW_RATE (1.6.17) register is used to control the output buffer slew rate (0 - normal, 1 - high). The DRIVER (1.6.23) register is used to configure output buffer drive strength (0 - weakest, 3 - strongest). The HYST (1.6.18) register is used to control input buffer hysteresis (0 - off, 1 - on).

1.4.1 Pull-up/down Configuration

When the GPIO pin is configured as an input, all configurations of pull resistors are available: pull-up (Figure 1.4), pull-down (Figure 1.3), bus-keeper (Figure 1.5) or floating (no-pull) (Figure 1.2). When the GPIO pin is configured as an output, the pull resistors will be disconnected except the situation when the pull resistors are configured as bus-keeper.

Bus-keeper configuration utilizes internal pull resistors to maintain the last GPIO pin level. Pull-down resistor is on when the last GPIO pin level was low. Pull-up resistor is on when the last GPIO pin level was high (Figure 1.5).

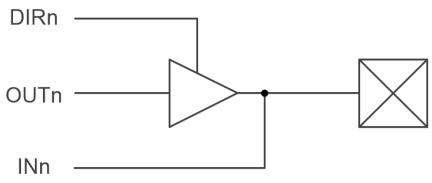


Figure 1.2. No-pull GPIO pin configuration.

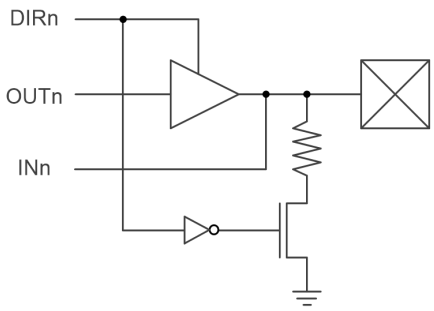


Figure 1.3. Pull-down GPIO pin configuration.

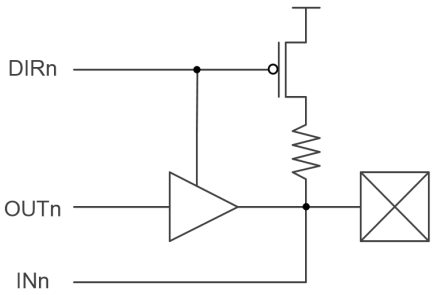


Figure 1.4. Pull-up GPIO pin configuration.

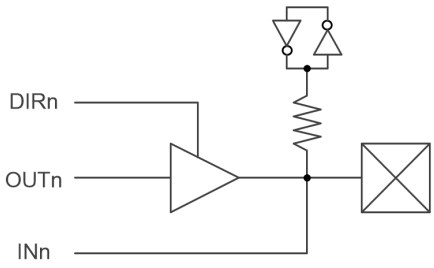


Figure 1.5. Bus-keeper GPIO pin configuration.

1.5 Pin Configuration

1.5.1 Edge Detection

The GPIO controller allows to synchronously detect rising or falling edge of GPIO pin as well as GPIO pin low level. The high level detection can be realized by using invert function. Each GPIO event can be signalized by the interrupt functionality.

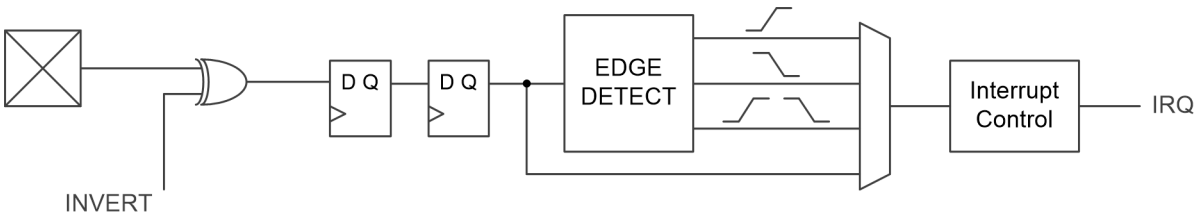


Figure 1.6. GPIO pin edge and level detection.

1.5.2 Alternative Functions

Each GPIO pin can be used to handle alternative functions for up to three other peripherals. Alternative functions configuration can be accessed using ALTER register (1.6.25).

1.5.3 Interrupts

The GPIO controller defines two interrupt sources: INT0 and INT1. Each of the GPIO pins can be assigned to any or both interrupt sources. The configuration of the signaling edge or level can be setup in the SENSE register (1.6.19). In case of the edge detection, the interrupt is generated the next clock cycle after the GPIO pin level changed. Low level interrupt is signalized as long as the GPIO pin level remains low. High level detection can be realized by using invert function. Interrupt lines are cleared by writing one to corresponding bit field in interrupt flags register (1.6.13).

1.6 Configuration Registers

1.6.1 Registers List

Address Offset	Register	Name
0x00	DIR	Direction Register
0x04	DIRSET	Direction Set Register
0x08	DIRCLR	Direction Clear Register
0x0C	DIRTGL	Direction Toggle Register
0x10	OUT	Output Register
0x14	OUTSET	Output Set Register
0x18	OUTCLR	Output Clear Register
0x1C	OUTTGL	Output Toggle Register
0x20	IN	Input Register
0x24	INT0	Interrupt INT0 Register
0x28	INT1	Interrupt INT1 Register
0x2C	IRQF	Interrupt Flags Register
0x30	CTRL	Control Register
0x34	IRQMAP	Interrupt Mapping Register
0x38	INVERT	Invert Register
0x3C	SLEW_RATE	Slew Rate Register
0x40	HYST	Hysteresis Register
0x44	SENSE_LO	Sense Config Register Low
0x48	SENSE_HI	Sense Config Register High
0x4C-0x5C	Reserved	
0x54	PULL_LO	Pull Config Register Low
0x58	PULL_HI	Pull Config Register High
0x5C	DRIVER_LO	Driver Config Register Low
0x60	DRIVER_HI	Driver Config Register High
0x64	ALTER_LO	Alternative Function Register Low
0x68	ALTER_HI	Alternative Function Register High

1.6.2 Direction Register

Address: 0x00

31	30	···	···	···	2	1	0
DIR31	DIR30	···	···	···	DIR2	DIR1	DIR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

DIRn *GPIO Direction n*

0 GPIO pin *n* is input.

1 GPIO pin *n* is output.

1.6.3 Direction Set Register

Address: 0x04

31	30	···	···	···	2	1	0
SET31	SET30	···	···	···	SET2	SET1	SET0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0

SETn *GPIO Set Direction n*

Writing one to bit *n* sets the corresponding GPIO pin as output.

1.6.4 Direction Clear Register

Address: 0x08

31	30	···	···	···	2	1	0
CLR31	CLR30	···	···	···	CLR2	CLR1	CLR0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0

CLRn *GPIO Clear Direction n*

Writing one to bit *n* sets the corresponding GPIO pin as input.

1.6.5 Direction Toggle Register

Address: 0x0C

31	30	···	···	···	2	1	0
TGL31	TGL30	···	···	···	TGL2	TGL1	TGL0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0

TGLn *GPIO Toggle Direction n*

Writing one to bit *n* toggles the corresponding GPIO pin direction.

1.6.6 Output Register

Address: 0x10

31	30	···	···	···	2	1	0
OUT31	OUT30	···	···	···	OUT2	OUT1	OUT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

OUTn *GPIO Output n*

0 GPIO pin *n* output is set low.

1 GPIO pin *n* output is set high.

1.6.7 Output Set Register

Address: 0x14

31	30	···	···	···	2	1	0
SET31	SET30	···	···	···	SET2	SET1	SET0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0

SETn *GPIO Set Output n*

Writing one to bit *n* sets the corresponding GPIO pin output value to high state.

1.6.8 Output Clear Register

Address: 0x18

31	30	···	···	···	2	1	0
CLR31	CLR30	···	···	···	CLR2	CLR1	CLR0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0

CLRn *GPIO Clear Output n*

Writing one to bit *n* sets the corresponding GPIO pin output value to low state.

1.6.9 Output Toggle Register

Address: 0x1C

31	30	···	···	···	2	1	0
TGL31	TGL30	···	···	···	TGL2	TGL1	TGL0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0

TGLn *GPIO Toggle Output n*

Writing one to bit *n* toggles the corresponding GPIO pin output value.

1.6.10 Input Register

Address: 0x20

31	30	···	···	···	2	1	0
IN31	IN30	···	···	···	IN2	IN1	IN0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0

INn *GPIO Input n*

0 GPIO pin *n* is in low state.

1 GPIO pin *n* is in high state.

1.6.11 Interrupt INT0 Mask Register

Address: 0x24

31	30	···	···	···	2	1	0
PIN31	PIN30	···	···	···	PIN2	PIN1	PIN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

PINn *GPIO INT0 Mask n*

0 GPIO pin *n* is not taking part in INT0 interrupr generation.

1 GPIO pin *n* takes part in INT0 interrupr generation.

1.6.12 Interrupt INT1 Mask Register

Address: 0x28

31	30	...	...	...	2	1	0
PIN31	PIN30	...	...	...	PIN2	PIN1	PIN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

PINn *GPIO INT1 Mask n*

0 GPIO pin *n* is not taking part in INT1 interrupt generation.

1 GPIO pin *n* takes part in INT1 interrupt generation.

1.6.13 Interrupt Flags Register

Address: 0x2C

31	30	...	...	...	...	9	8
		...	...	...	...		
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
						INT1IF	INT0IF
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0

INT0IF *Interrupt INT0 Flag*

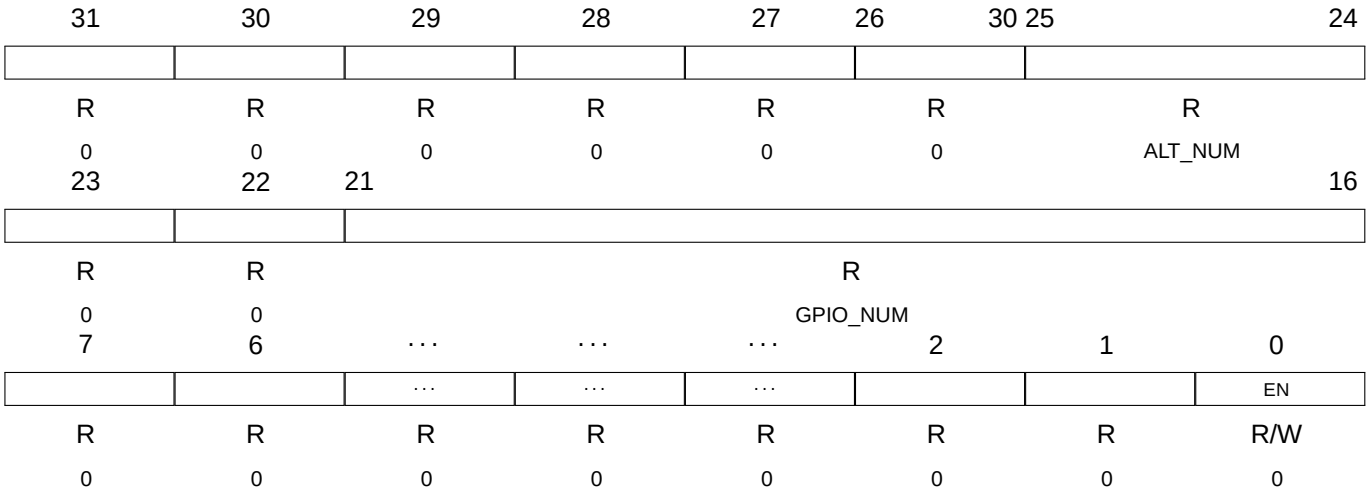
Bit reflects the state of the INT0 interrupt line. Writing one to this bit will clear the interrupt.

INT1IF *Interrupt INT1 Flag*

Bit reflects the state of the INT1 interrupt line. Writing one to this bit will clear the interrupt.

1.6.14 Control Register

Address: 0x30



EN *GPIO Controller Enable*

0 GPIO Controller is disabled. Clock to the module is stopped.

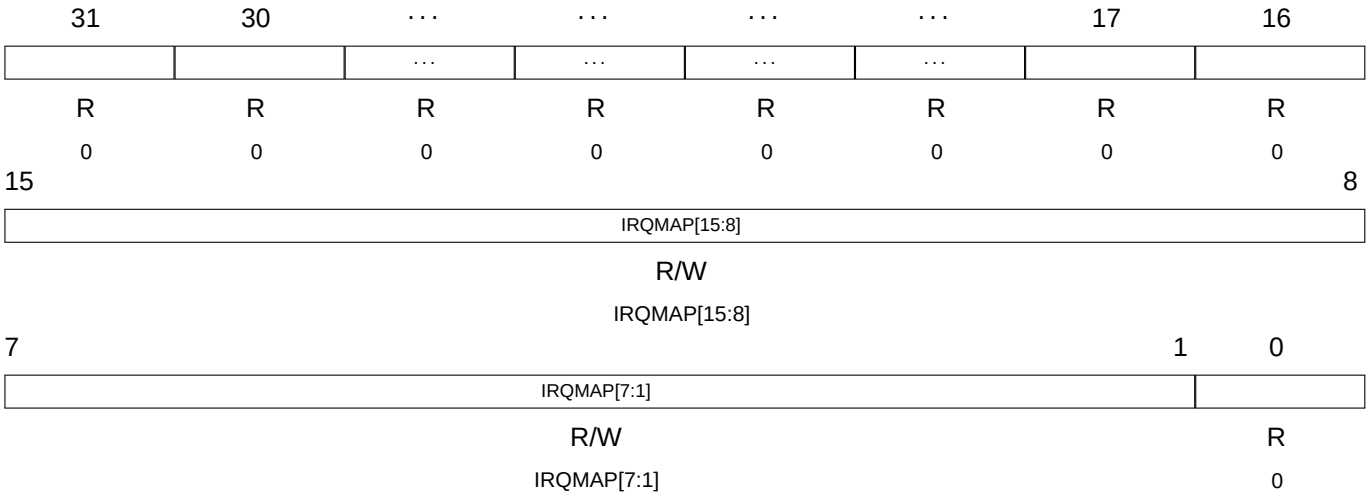
1 GPIO Controller is enabled. Clock to the module is supplied.

GPIO_NUM[5:0] *Number of implemented GPIO pins*

ALT_NUM[1:0] *Number of implemented GPIO alternative functions*

1.6.15 Interrupt Mapping Register

Address: 0x34



IRQMAP[15:1] *Interrupt Mapping*

Each set bit represents the interrupt number that will be passed to interrupt controller. It is allowed to set more than one bit.

1.6.16 Invert Register

Address: 0x38

INVn *GPIO Pin n Invert*

Invert logic of the GPIO pin in both input and output modes. Input register will return high when the GPIO pin is low. Writing one in output register will result in low state on GPIO pin output.

1.6.17 Slew Rate Register

Address: 0x3C

31	30	...	...	...	2	1	0
SLR31	SLR30	...	...	...	SLR2	SLR1	SLR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

SLRn *GPIO Pin n High Slew Rate Enable*

Writing one to bit *n* enables high slew rate of the corresponding GPIO pin.

1.6.18 Hysteresis Register

Address: 0x40

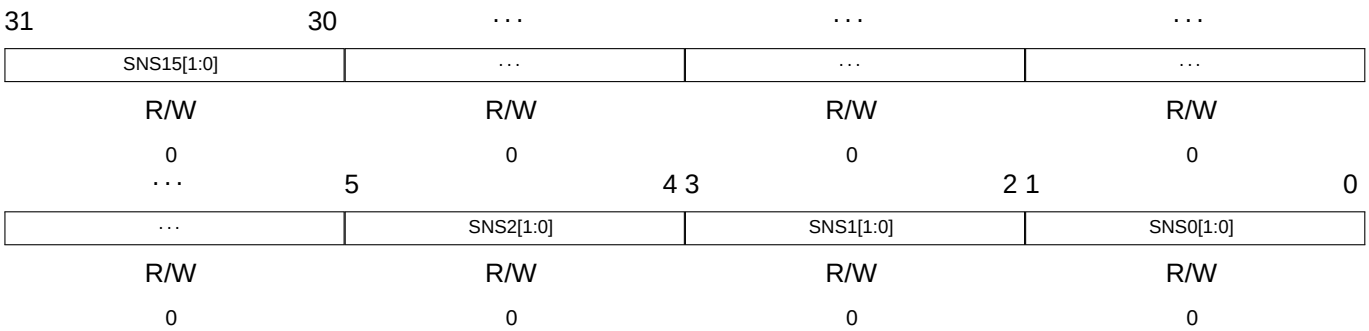
31	30	...	...	...	2	1	0
HST31	HST30	...	...	...	HST2	HST1	HST0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

HSTn *GPIO Pin n Hysteresis Enable*

Writing one to bit *n* enables input hysteresis of the corresponding GPIO pin.

1.6.19 Sense Config Register Low

Address: 0x44



31	30	...	...	...
SNS15[1:0]	...	...	...	...
R/W	R/W	R/W	R/W	R/W
0	0	0	0	0

...	5	4 3	2 1	0
...	SNS2[1:0]	SNS1[1:0]	SNS0[1:0]	
R/W	R/W	R/W	R/W	
0	0	0	0	

SNS[1:0]n *GPIO Pin n Sense Config*

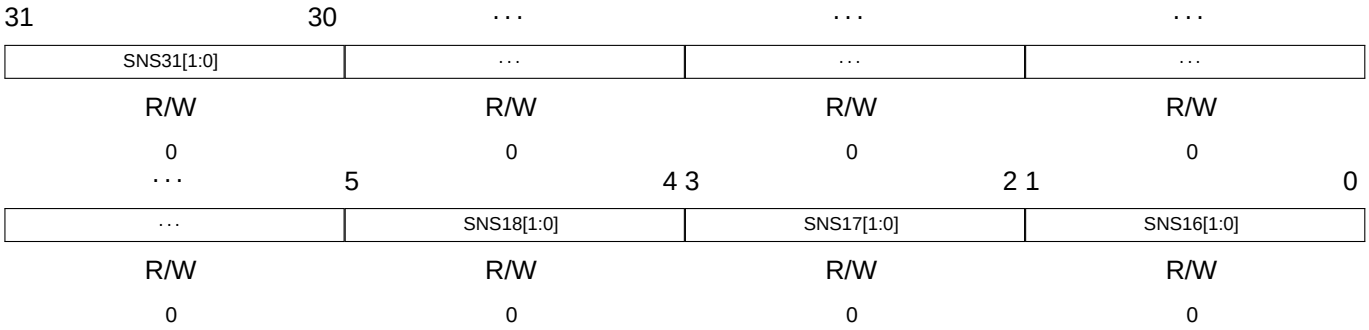
The register is used to configure interrupt event for GPIO pins range from 0 to 15. Low level interrupt is signalized

as long as the GPIO pin level remains low. High level detection can be realized by using invert function.

SNSn[1:0]	Detection Mode
01	rising edge
10	falling edge
11	any edge
00	low level

1.6.20 Sense Config Register High

Address: 0x48

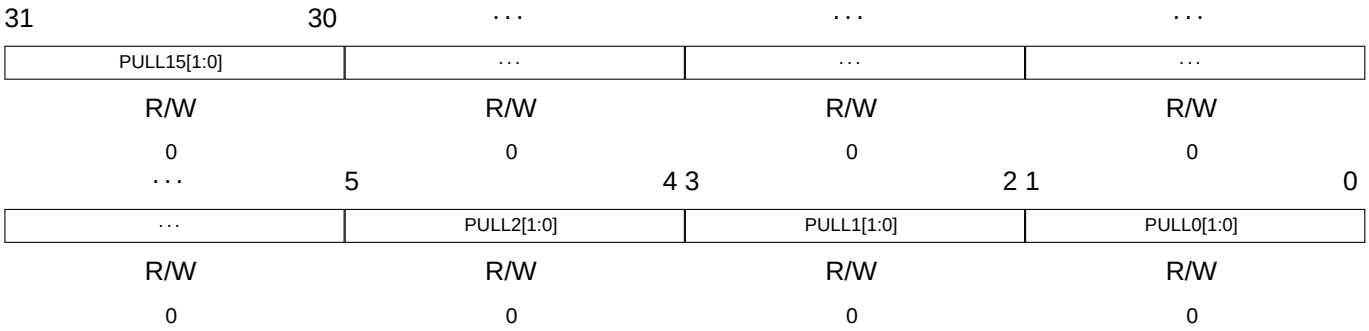


SNS[1:0]n *GPIO Pin n Sense Config*

The register is used to configure interrupt event for GPIO pins range from 16 to 31. Low level interrupt is signalized as long as the GPIO pin level remains low. High level detection can be realized by using invert function.

1.6.21 Pull Config Register Low

Address: 0x54



PULL[1:0]n *GPIO Pin n Pull Config*

Pull resistors configuration for GPIO pins range from 0 to 15.

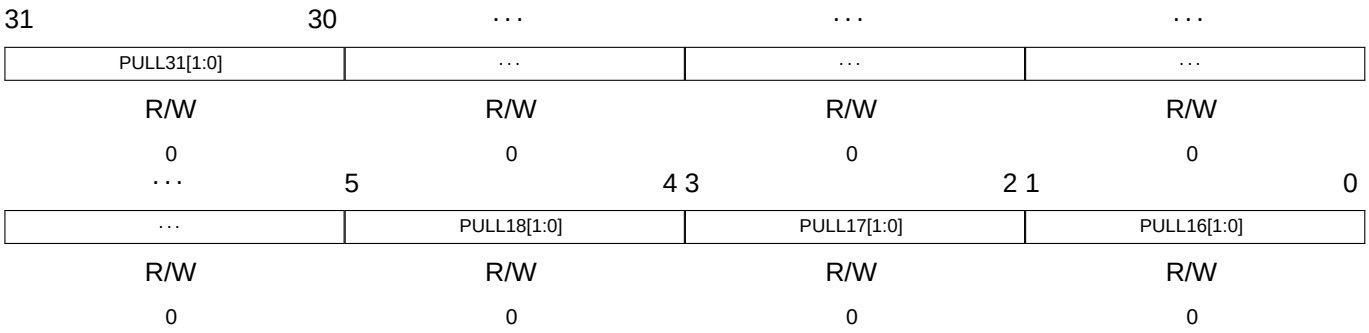
PULLn[1:0]	Pull Configuration
00	no-pull
01	pull-up
10	pull-down
11	bus-keeper

Depending on the GPIO pin direction, some pull configurations can be disabled:

- When the GPIO pin is configured as an input, all configurations of pull resistors are available.
- When the GPIO pin is configured as an output, only no-pull and bus-keeper is available.

1.6.22 Pull Config Register High

Address: 0x58



PULL[1:0]n *GPIO Pin n Pull Config*

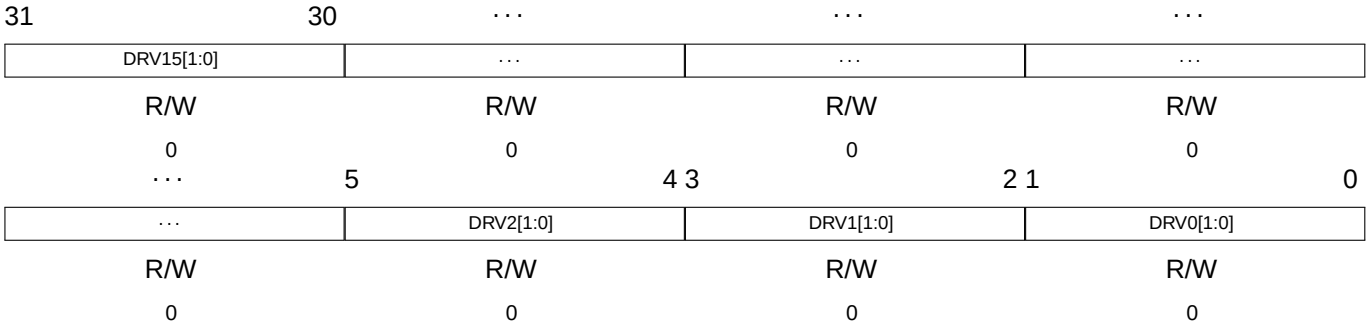
Pull resistors configuration for GPIO pins range from 16 to 31.

Depending on the GPIO pin direction, some pull configurations can be disabled:

- When the GPIO pin is configured as an input, all configurations of pull resistors are available.
- When the GPIO pin is configured as an output, only no-pull and bus-keeper is available.

1.6.23 Driver Config Register Low

Address: 0x5C



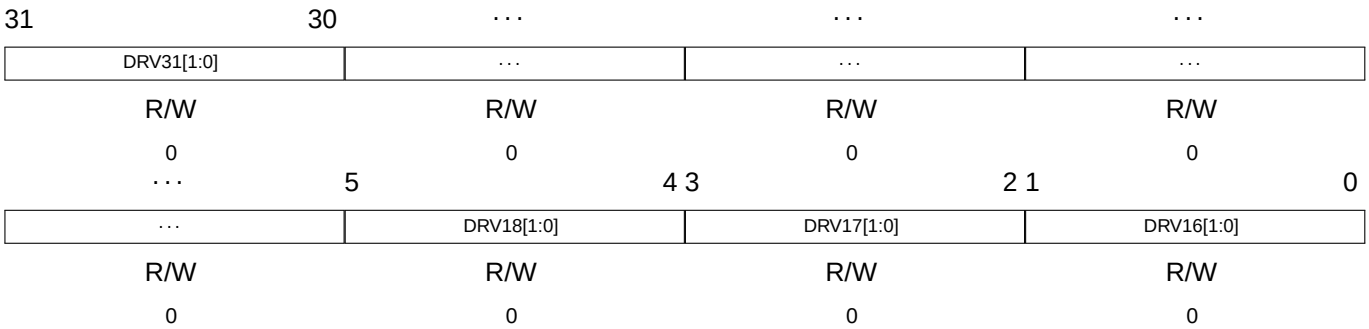
DRV[1:0]n *GPIO Pin n Driver Config*

Drive strength configuration for GPIO pins range from 0 to 15.

DRVn[1:0]	Drive Strength
00	base drive
01	2 x base drive
10	4 x base drive
11	6 x base drive

1.6.24 Driver Config Register High

Address: 0x60

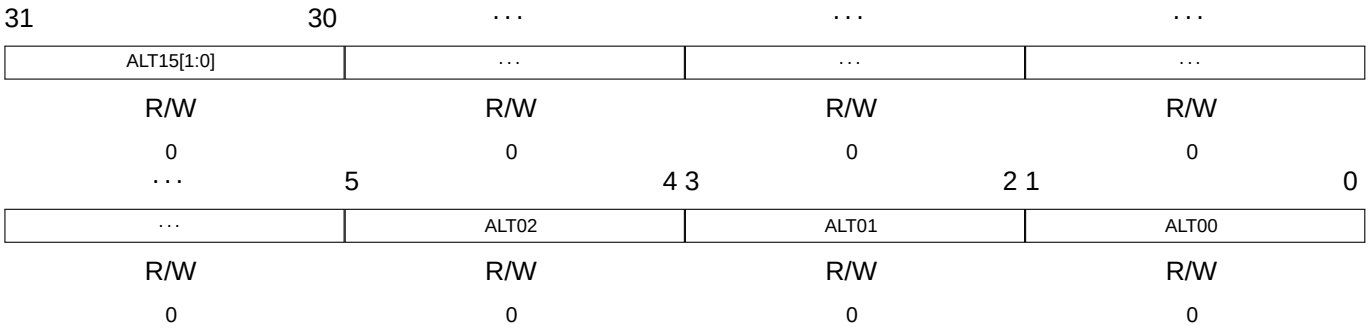


DRV[1:0]n *GPIO Pin n Driver Config*

Drive strength configuration for GPIO pins range from 16 to 31.

1.6.25 Alternative Function Register Low

Address: 0x64



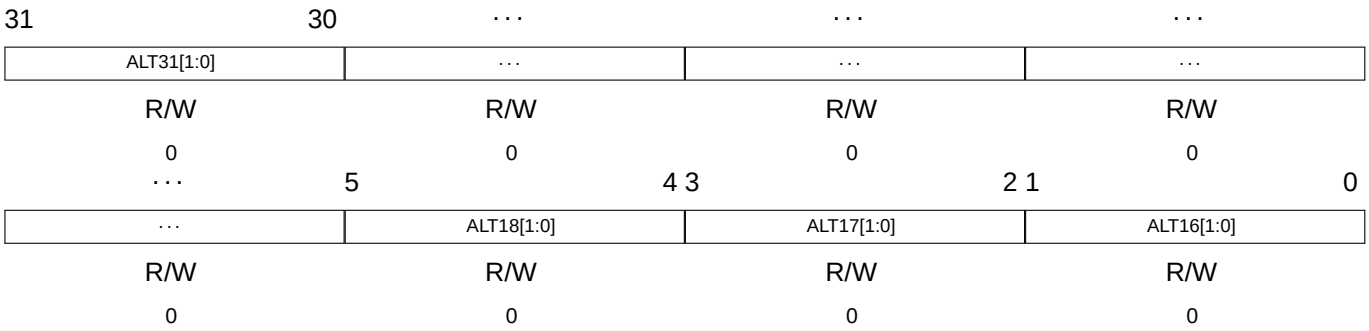
ALT[1:0]n *GPIO Pin n Alternative Function Config*

Alternative function configuration for GPIO pins range from 0 to 15.

ALTn[1:0]	Alternative Function
00	Pin is controlled by GPIO
01	Pin is controlled by alternative function 0
10	Pin is controlled by alternative function 1
11	Pin is controlled by alternative function 2

1.6.26 Alternative Function Register High

Address: 0x68



ALT[1:0]n *GPIO Pin n Alternative Function Config*

Alternative function configuration for GPIO pins range from 16 to 31.

1.7 Implementation

1.7.1 Design Structure

The synthesible RTL IP core part (*AXI/rtl*, *COMMON/rtl* and *GPIO/rtl* folder) utilizes Verilog 2005 HDL. The testbench part (*AXI/tb* and *GPIO/tb* folder) uses SystemVerilog language.

```
AXI
├── rtl
│   ├── AXI_PERIPH
│   │   └── amba_axilite_apb_bridge.v
└── tb
    ├── AXI_PERIPH
    │   ├── APB
    │   │   └── virtual_APB_slave.sv
    │   ├── AXI
    │   │   └── tb_amba_axilite_tasks.sv
    │   ├── common
    │   │   └── timescale.v
    │   ├── run
    │   │   └── ncvlog_amba_axilite_apb_bridge.sh
    │   ├── tests
    │   │   └── tb_read_write_test.sv
    │   └── tb_amba_axilite_apb_bridge.sv
COMMON
├── rtl
    ├── DFF_en.v
    ├── edge_detector.v
    └── synchronizer.v
GPIO
├── beh
├── rtl
│   ├── APB_GPIO_PORT.v
│   ├── AXILITE_GPIO_PORT.v
│   ├── GPIO_define.v
│   ├── GPIO_PORT_config.v
│   ├── GPIO_PORT.v
│   └── GPIO.v
└── tb
    ├── APB
    │   ├── tb_APB_GPIO_PORT_init.v
    │   └── tb_APB_GPIO_PORT_reg_access_tasks.v
    ├── common
    │   │   └── tb_config_tasks.v .5 tb_data_transfer_tasks.v
    │   └── timescale.v
    ├── run
    │   ├── irun_apb_gpio_all.sh
    │   └── irun_apb_gpio.sh
    ├── tests
    │   ├── tb_ALTERNATIVE_FUNCTION_test.sv
    │   ├── tb_DIR_test.sv
    │   ├── tb_IN_test.sv
    │   ├── tb_INTERRUPT_MAPPING_test.sv
    │   ├── tb_OUT_test.sv
    │   └── tb_PULL_CONFIG_test.sv
```


```
│   │   └── tb_SENSE_INT_test.sv
│   └── tb_APB_GPIO_PORT.sv
└── compile.list
```

1.7.2 Simulation Flow

The IP Core is provided with self-checking testbench to verify the correct operation of the IP prior to use in a design. The testbench is divided into two environments. The first one tests the APB_GPIO_PORT module. To run the simulation using Cadence® Incisive® Enterprise Simulator run *irun_apb_gpio_all.sh* script located in the *GPIO/tb/run* folder. The script will run multiple simulations with different configuration options. Each simulation should end with reporting no errors. The second environment tests the AXI4-Lite to APB3 converter. To run the simulation using Cadence® Incisive® Enterprise Simulator run *ncvlog_amba_axilite_apb_bridge.sh* script located in the *AXI/tb/AXI_PERIPH/run* folder. The simulation should end with reporting no errors. The AXILITE_GPIO_PORT top module is composed of the APB_GPIO_PORT core and the amba_axilite_apb_bridge AXI4-Lite to APB3 converter.

1.7.3 Clock and Reset

The CC-GPIO-AXI utilizes a fully synchronous design with one positive edge clocking domain and negative asynchronous reset assertion. External reset synchronizer has to be used to ensure synchronous reset deassertion.

1.7.4 Constraints

In most cases only module output ports are registered. Therefore, it is a good practice to reserve the entire clock cycle for module inputs combinational logic and set minimal input delay (*set_input_delay* command). Registered outputs leave the entire clock cycle for external logic (*set_output_delay* command).

The GPIO pad inputs to the GPIO controller are synchronized using Synchronizer module located in the synchronizer.v file. If possible, they should be replaced with integrated 2FF synchronizers from the target technology library. Otherwise, max delay (*set_max_delay* command) of 10% to 20% of one destination clock cycle should be set between synchronizer stages. Do not use dynamic FFs to implement synchronizer module. Alternative input function is realized using combinational logic and uses raw GPIO pad input. Synchronization should be provided by the target peripheral.

1.7.5 Configuration Options

The table below shows the generic parameters of the core.

Generic name	Description	Range	Default
gpio_pins	Number of GPIO pins	1:32	32
out_override_signals_number	Number of GPIO alternative functions	0, 1, 3[1]	1
out_override_select_width	Width of alternative functions select register	0, 1, 2[1]	1
default_interrupt_MAPPING	Reset value of interrupt_MAPPING register	1:32767	0

1.7.6 Signals Description

Signal name	Description	I/O	Active	Type
ACLK	Synchronous clock	I	rising	clock
ARESETn	Asynchronous reset	I	low	reset
AWADDR[6:2]	AXI4-Lite write address	I	data	comb.
AWPROT[2:0][2]	AXI4-Lite write address protection type	I	data	comb.
AWVALID	AXI4-Lite write address valid	I	high	comb.
AWREADY	AXI4-Lite write address ready	O	high	reg.
WDATA[31:0]	AXI4-Lite write data	I	data	comb.
WSTRB[3:0]	AXI4-Lite write strobe	I	high	comb.
WVALID	AXI4-Lite write valid	I	high	comb.
WREADY	AXI4-Lite write ready	O	high	reg.
BRESP[1:0]	AXI4-Lite write response	O	data	reg.
BVALID	AXI4-Lite write response valid	O	high	reg.
BREADY	AXI4-Lite write response ready	I	high	comb.
ARADDR[6:2]	AXI4-Lite read address	I	data	comb.
ARPROT[2:0][2]	AXI4-Lite read address protection type	I	data	comb.
ARVALID	AXI4-Lite read address valid	I	high	comb.
ARREADY	AXI4-Lite read address ready	O	high	reg.
RDATA[31:0]	AXI4-Lite read data	O	data	reg.
RRESP[1:0]	AXI4-Lite read response	O	data	reg.
RVALID	AXI4-Lite read valid	O	high	reg.
RREADY	AXI4-Lite read ready	I	high	comb.

[1] Only (out_override_signals_number = 0, out_override_select_width = 0), (out_override_signals_number = 1, out_override_select_width = 1) and (out_override_signals_number = 3, out_override_select_width = 2) combinations are supported.

[2] Signal is not used in the design.

overridden_data_dir [out_override_signals_number*gpio_pins-1:0]	Alternative functions direction vector	I	high	comb.
overridden_data_from_peripheral [out_override_signals_number*gpio_pins-1:0]	Alternative functions data from peripherals vector	I	data	comb.
overridden_data_to_peripheral_marker [out_override_signals_number*gpio_pins-1:0]	Alternative functions selection marker vector	O	high	reg.
overridden_data_to_peripheral [gpio_pins-1:0]	Alternative functions data to peripherals	O	data	comb.[3]
pulldown_NMOS_enable[gpio_pins-1:0]	Pull-down enable signal	O	high	reg.
pullup_PMOS_enable[gpio_pins-1:0]	Pull-up enable signal	O	high	reg.
output_enable[gpio_pins-1:0]	Output enable signal	O	high	reg.
driver_2x[gpio_pins-1:0]	Driver x2 selection signal	O	high	reg.
driver_4x[gpio_pins-1:0]	Driver x4 selection signal	O	high	reg.
slew_rate_high[gpio_pins-1:0]	High slew rate enable signal	O	high	reg.
hysteresis_enable[gpio_pins-1:0]	Hysteresis enable signal	O	high	reg.
PAD_output[gpio_pins-1:0]	Driver output data signal	O	data	comb.[4]
PAD_input[gpio_pins-1:0]	Pad input signal	I	data	comb.
interrupt_INT0	INT0 interrupt signal	O	high	reg.
interrupt_INT1	INT1 interrupt signal	O	high	reg.
interrupt_MAPPING[15:1]	Interrupt mapping vector	O	data	reg.
clock_request	Clock request signal	O	high	reg.

[3] Value is a combinational function of the raw PAD_input and the invert function.
[4] The PAD output value is a combinational function of the alternative functions data multiplexer and the invert function.

1.7.7 Alternative Functions Connection

Figure 1.7 shows the exemplary connection of the GPIO alternative functions. GPIO0 is connected in such a way that its first alternative function serves as output from Peripheral0. The second alternative function is an input to Peripheral2. The third alternative function is an input to Peripheral2. Please note that used or-gate forces high state when the alternative function is not selected. GPIO1 is connected in such a way that its first alternative function is an input to Peripheral0. Please note that used and-gate forces low state when the alternative function is not selected. The second alternative function is an input to Peripheral2. The third alternative function is an output from Peripheral1. Peripheral2 input is constructed in such a way that if both GPIO0 second alternative function and GPIO1 second alternative function is disabled the input is fed with constant high or low value. If both alternative functions are enabled, input from GPIO1 has priority.

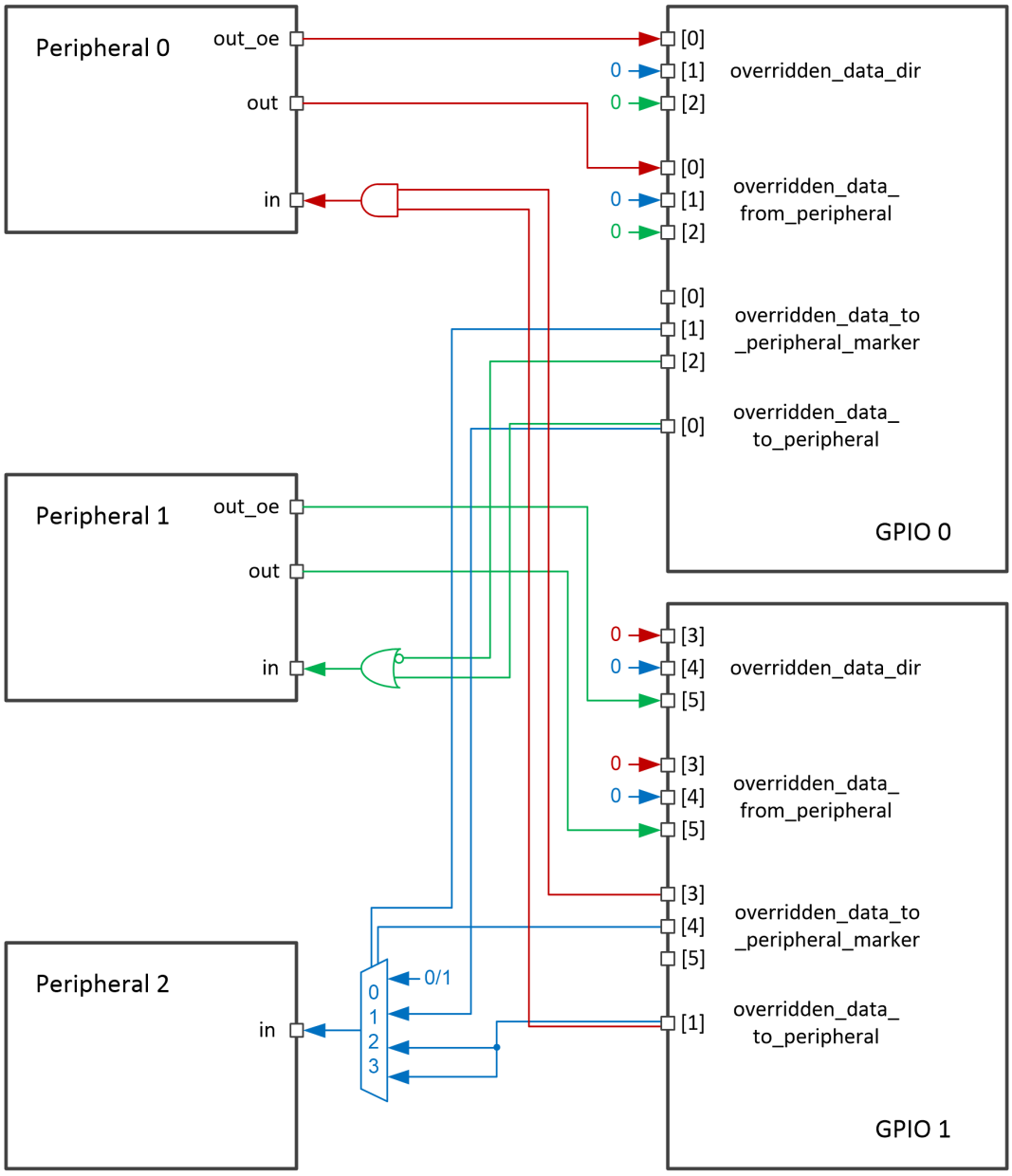


Figure 1.7. Exemplary connection of GPIO alternative functions.

1.7.8 Instantiation

```
icg
icg_gpio (
    .E(gpio_PSEL|gpio_clock_request),
    .clk(PCLK),
    .gclk(gpio_clk),
    .scan_enable(scan_enable));

APB_GPIO_PORT #(
    .gpio_ping(CFG_GPIO_PINS),
    .out_override_signals_number(CFG_OVERRIDE_SIGNALS),
    .out_override_select_width(CFG_OVERRIDE_WIDTH),
    .default_interrupt_MAPPING(CFG_DEF_INT_MAPPING))
APB_GPIO_PORT_u (
    .ACLK(spi_clk),
    .ARESETn(ARESETn),
    .AWADDR(AWADDR[6:2]),
    .AWPROT(AWPROT),
    .AWVALID(AWVALID),
    .AWREADY(AWREADY),
    .WDATA(WDATA),
    .WSTRB(WSTRB),
    .WVALID(WVALID),
    .WREADY(WREADY),
    .BRESP(BRESP),
    .BVALID(BVALID),
    .BREADY(BREADY),
    .ARADDR(ARADDR[6:2]),
    .ARPROT(ARPROT),
    .ARVALID(ARVALID),
    .ARREADY(ARREADY),
    .RDATA(RDATA),
    .RRESP(RRESP),
    .RVALID(RVALID),
    .RREADY(RREADY),
    .overridden_data_dir(overridden_data_dir),
    .overridden_data_from_peripheral(
     overridden_data_from_peripheral),
    .overridden_data_to_peripheral_marker(
     overridden_data_to_peripheral_marker),
```

```
    .overridden_data_to_peripheral(
     overridden_data_to_peripheral),
    .pulldown_NMOS_enable(pulldown_NMOS_enable),
    .pullup_PMOS_enable(pullup_PMOS_enable),
    .output_enable(output_enable),
    .driver_2x(driver_2x),
    .driver_4x(driver_4x),
    .slew_rate_high(slew_rate_high),
    .hysteresis_enable(hysteresis_enable),
    .PAD_output(PAD_output),
    .PAD_input(PAD_inpur),
    .interrupr_INT0(gpio_interrupr_INT0),
    .interrupr_INT1(gpio_interrupr_INT1),
    .interrupt_MAPPING(gpio_interrupt_MAPPING),
    .clock_request(gpio_clock_request));

assign gpio_irq          = gpio_interrupr_INT0 |
                           gpio_interrupr_INT1;

assign gpio_irq_vector   = pdma_interrupt_MAPPING & {15{gpio_irq}};

// GPIO PAD MODEL
gpio_io_pad_model #(
    .GPIO_NUM(CFG_GPIO_PINS))
gpio_pad_model (
    .core_input(PAD_input),
    .core_output(PAD_output),
    .GPIO_pad(GPIO_pad),
    .pulldown_NMOS_enable(pulldown_NMOS_enable),
    .pullup_PMOS_enable(pulldown_PMOS_enable),
    .output_enable(output_enable),
    .driver_2x(driver_2x),
    .driver_4x(driver_4x),
    .slew_rate_high(slew_rate_high),
    .hysteresis_enable(hysteresis_enable));
```


1.8 Revision History

Doc. Rev.	Date	Comments
1.1	11-2018	Editorial corrections in 1.7.8 Instantiation section.
1.0	11-2017	First Issue.

ChipCraft Sp. z o.o. Dobrzańskiego 3 lok. BS073, 20-262 Lublin, POLAND www.chipcraft-ic.com

 CC-GPIO-APB-Doc_112018.