OpenCores.Org

GPIO IP Core Specification

Authors:
Damjan Lampret
lampret@opencores.org
Goran Djakovic
goran.djakovic@flextronics.si

Rev. 1.2

December 17, 2003

This page has been intentionally left blank.

Revision History

Rev.	Date	Author	Description
0.1	4/2/01	Damjan Lampret	First Draft
0.2	20/2/01	Damjan Lampret	Compliance with WISHBONE Rev.B1
0.3	29/10/01	Damjan Lampret	Updated names of ports and RGPIO_CTRL. Added additional parameters to Appendix A.
1.0	25/12/01	Damjan Lampret	Added RGPIO_INTS and renamed RGPIO_CTRL[INT] into RGPIO_CTRL[INTS]. Updated Appendix Core Configuration.
1.1	17/12/03	Goran Djakovic	Added RGPIO_ECLK and RGPIO_NEC, modified RGPIO_CTRL
1.2	11/11/10	Mark Peryer	Replaced Wishbone host bus interface with APB

Table Of Contents

Table Of Figures

Table Of Tables

1 Introduction

The GPIO IP core is user-programmable general-purpose I/O controller. Its use is to implement functions that are not implemented with the dedicated controllers in a system and require simple input and/or output software controlled signals.

Features

The following lists the main features of GPIO IP core:

- Number of general-purpose I/O signals is user selectable and can be in range from 1 to 32. For more I/Os several GPIO cores can be used in parallel.
- All general-purpose I/O signals can be bi-directional (external bi-directional I/O cells are required in this case).
- All general-purpose I/O signals can be three-stated or open-drain enabled (external three-state or open-drain I/O cells are required in this case).
- General-purpose I/O signals programmed as inputs can cause an interrupt request to the CPU.
- General-purpose I/O signals programmed as inputs can be registered at raising edge of system clock or at user programmed edge of external clock.
- All general-purpose I/O signals are programmed as inputs at hardware reset.
- Auxiliary inputs to GPIO core to bypass outputs from RGPIO_OUT register.
- Alternative input reference clock signal from external interface.
- Extremely configurable (implementation of registers, external clock inverted versus negedge flip-flops etc.)
- APB 32 bit bus interface

2 Architecture

Figure 1 below shows general architecture of GPIO IP core. It consists of four main building blocks:

- APB host interface
- GPIO registers
- Auxiliary inputs
- Interface to external I/O cells and pads

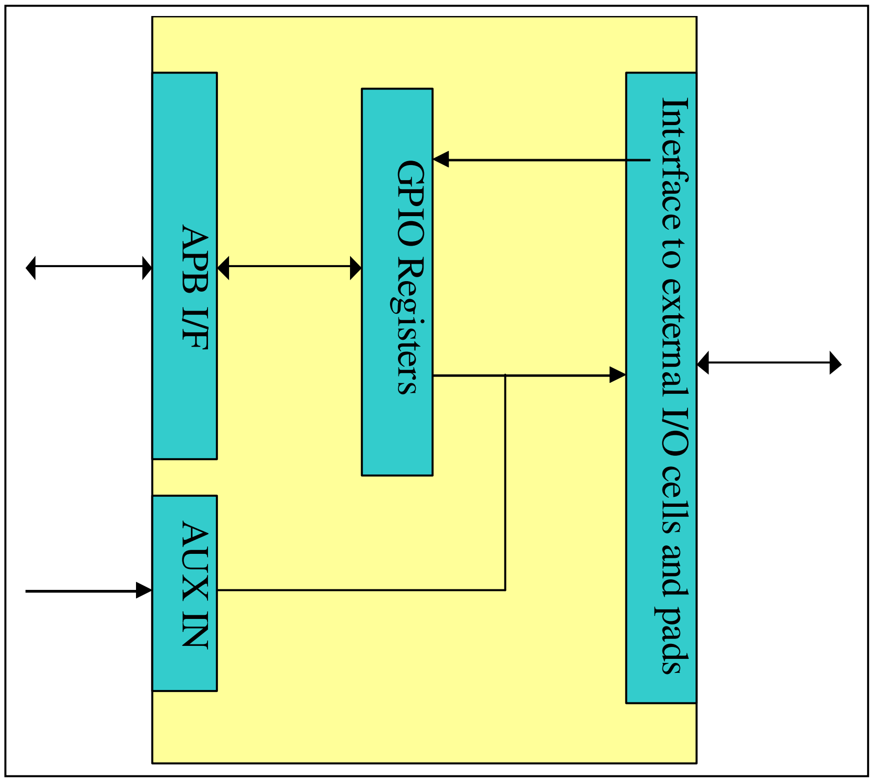


Figure 1. Core Architecture

Clocks

The GPIO core has two clock domains. All registers except RGPIO_IN are in system clock domain.
RGPIO_IN register can be clocked by system clock or by external clock reference.

APB Interface

The host interface is implemented using a 32 bit APB compliant slave interface.

GPIO Registers

The GPIO IP Core has several software accessible registers. Most registers have the same width as number of general-purpose I/O signals and they can be from 1 – 32 bits. The host through these registers programs type and operation of each general-purpose I/O signal.

Auxiliary Inputs

The auxiliary inputs can bypass RGPIO_OUT outputs based on programming of RPGIO_AUX register. Auxiliary inputs are used to multiplex other on-chip peripherals on GPIO pins.

Interface to External I/O Cells and Pads

External interface connects GPIO core to external I/O ring cells and pads. To support open-drain or three-state outputs, appropriate open-drain or three-state I/O cells must be used.
Part of external interface is also ECLK register. It can be used to register inputs based on external clock reference.

3 Operation

This section describes the operation of the GPIO core. The GPIO core provides toggling of general-purpose outputs and sampling of general-purpose inputs under software control.

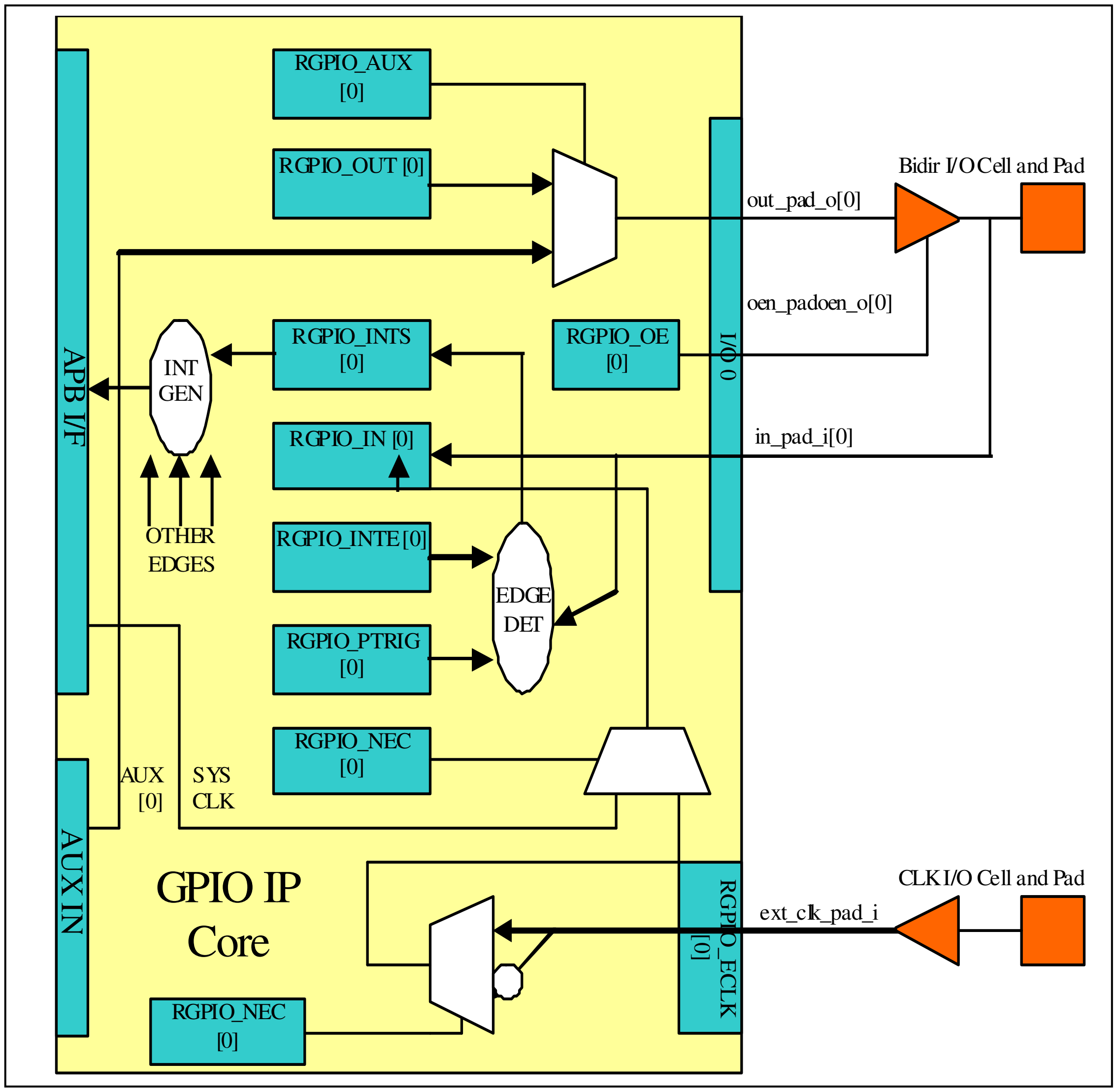


Figure 2. Block Diagram of GPIO Logic

General-purpose inputs can generate interrupts so that software does not have to be in poll mode all the time when sampling inputs.
Switching output drivers into open-drain or three-state mode will disable general-purpose outputs.
To lower number of pins of the chip, other on-chip peripherals can be multiplexed together with the GPIO pins. For this purpose, auxiliary inputs can be multiplexed on general-purpose outputs.

Hardware Reset

Following hardware reset all general-purpose I/O signals are set into input mode. Meaning, all output drivers are disabled. All interrupts are masked, so that inputs would not generate any spurious interrupts. Gpio_eclk signal is not used to latch inputs into RGPIO_IN register; instead system clock is used.

General-Purpose I/O as Polled Input

To use general-purpose I/O as input only, corresponding bit in RGPIO_OE register must be cleared to select input mode. Bit RGPIO_CTRL[INTE] and corresponding bit in RGPIO_INTE register must be cleared as well, to disabled generation of interrupts.
Bit RGPIO_IN register reflects registered value of general-purpose input signal. RGPIO_IN is updated on positive edge of system clock or if RGPIO_ECLK appropriate bit is set, on gpio_eclk edge. Which clock edge is selected, is defined by value of RGPIO_NEC appropriate bit.

General-Purpose I/O as Input in Interrupt Mode

To use general-purpose I/O as input with generation of interrupts, corresponding bit in RGPIO_OE register must be cleared to select input mode. Corresponding bit in RGPIO_PTRIG register must be set to generate an interrupt on positive edge event on general-purpose input. To generate an interrupt on negative edge event, corresponding bit in RGPIO_PTRIG register must be cleared. If we are enabling interrupts for the first time, we also need to clear interrupt status register RGPIO_INTS. Last, RGPIO_CTRL[INTE] bit and corresponding bit in RGPIO_INTE register must be set to enable generation of interrupts.
Bit RGPIO_IN register reflects registered value of general-purpose input signal. RGPIO_IN is updated on positive edge of system clock or if RGPIO_ECLK appropriate bit is set, on gpio_eclk edge. Which clock edge is selected, is defined by value of RGPIO_NEC appropriate bit.
Which input caused an interrupt is recorded in interrupt status register RGPIO_INTS. Inputs that caused an interrupt since last clearing of RGPIO_INTS have bits set. Interrupt can be de-asserted by writing zero in RGPIO_INTS register and control register bit RGPIO_CTRL[INTS]. Another way to de-assert interrupts is to disable them by clearing control bit RGPIO_CTRL[INTE].

General-Purpose I/O as Output

To enable general-purpose I/O output driver, corresponding bit in RGPIO_OE must be set. Corresponding bit in RGPIO_OUT register must be set to the value that is required to be driven on output driver. Corresponding bit in RGPIO_INTE register must be cleared to disable generation of spurious interrupts.
Clearing bit in RGPIO_OE register will disable output driver and enable three-state or open-drain.

General-Purpose I/O as Bi-Directional I/O

To use general-purpose I/O as bi-directional signal, corresponding bit in RGPIO_OE must be toggled to enable or disable three-state or open-drain mode of bi-directional driver. Corresponding bit in RGPIO_OUT register must be set to the value that is required to be driven on output driver. Corresponding bit in RGPIO_INTE register must be cleared to disable generation of spurious interrupts. If input should generate interrupts, corresponding bit in RGPIO_INTE register must be set and if required also corresponding bit in RGPIO_PTRIG should be set.
Corresponding bit RGPIO_IN register reflects registered value of general-purpose input signal. RGPIO_IN is updated on positive edge of system clock or if RGPIO_ECLK bit is set, on gpio_eclk edge. Which clock edge is selected, is defined by value of RGPIO_NEC bit.
If an interrupt is enabled and pending, it can be de-asserted by writing zero in RGPIO_INTS register and control register bit RGPIO_CTRL[INTS]. Another way to de-assert interrupts is to disable them by clearing control bit RGPIO_CTRL[INTE]

General-Purpose I/O driven by Auxiliary Input

To drive general-purpose output with auxiliary input, corresponding bit in RGPIO_OE must be set to enable output driver. Corresponding bit in RGPIO_AUX must be set to enable multiplexing of auxiliary input onto general-purpose output.

4 Registers

This section describes all control and status register inside the GPIO core. The *Address* field indicates address in hexadecimal. *Width* specifies the number of bits in the register, and *Access* specifies the valid access types for that register. R/W stands for read and write access and R stands for read only access.
Width of most registers is user selectable and is set by the user of the GPIO core at synthesis time.

Registers list

Name	Address	Width	Access	Description
RGPIO_IN	Base + 0x0	1 - 32	R	GPIO input data
RGPIO_OUT	Base + 0x4	1 - 32	R/W	GPIO output data
RGPIO_OE	Base + 0x8	1 - 32	R/W	GPIO output driver enable
RGPIO_INTE	Base + 0xC	1 - 32	R/W	Interrupt enable
RGPIO_PTRIG	Base + 0x10	1 - 32	R/W	Type of event that triggers an interrupt
RGPIO_AUX	Base + 0x14	1 - 32	R/W	Multiplex auxiliary inputs to GPIO outputs
RGPIO_CTRL	Base + 0x18	2	R/W	Control register
RGPIO_INTS	Base + 0x1C	1 - 32	R/W	Interrupt status
RGPIO_ECLK	Base + 0x20	1 - 32	R/W	Enable gpio_eclk to latch RGPIO_IN
RGPIO_NEC	Base + 0x24	1 - 32	R/W	Select active edge of gpio_eclk

Table 1. List of All Software Accessible Registers

Register RGPIO_IN description

RGPIO_IN register latches general-purpose inputs. Reference clock is either system clock or ECLK input. Selection between both clocks is performed with RGPIO_ECLK register.

Bit #	Access	Reset	Description
1 - 32	R	0x0	Latched value of general-purpose inputs

Table 2. Input Register

Register RGPIO_OUT description

RGPIO_OUT register drives general-purpose outputs. Additionally, external I/O cells can be operated open-drain or three-state with RGPIO_OE register.

Bit #	Access	Reset	Description
1 - 32	R/W	0x0	General-purpose driven outputs

Table 3. Output Register

Register RGPIO_OE description

RGPIO_OE enables output/bi-directional mode of operation for each general-purpose I/O signal. When bit is set, corresponding general-purpose output driver is enabled. When bit is cleared, output/bi-directional driver is operating in open-drain or three-state mode.

Bit #	Access	Reset	Description
1 – 32	R/W	0x0	Output/bi-directional external I/O drivers enables

Table 4. Output Enable Register

Register RGPIO_INTE description

RGPIO_INTE register defines which general-purpose inputs generate interrupt to the host. When bit is set, corresponding general-purpose input generates interrupt.
See also global interrupt enable bit RGPIO_CTRL[INTE].

Bit #	Access	Reset	Description
1 – 32	R/W	0x0	Enables for of interrupts generated by general-purpose input signals

Table 5. Interrupt Enable Register

Register RGPIO_PTRIG description

RGPIO_PTRIG register defines which edge of a general-purpose input generates an interrupt. Generation of an interrupt must be first enabled in RGPIO_INTE register and global interrupt enable bit RGPIO_CTRL[INTE]. When bit is set, corresponding input generates an interrupt when positive edge is encountered. When bit is cleared, corresponding input generates an interrupt when negative edge is encountered.

Bit #	Access	Reset	Description
1 - 32	R/W	0x0	Triggering of an interrupt (positive edge when set,

			negative edge when cleared)

Table 6. Trigger Register

Register RGPIO_AUX description

RGPIO_AUX multiplexes auxiliary inputs to general-purpose outputs. When bit is set, corresponding auxiliary input drives corresponding general-purpose output instead of a bit in RGPIO_OUT register.

Bit #	Access	Reset	Description
1 - 32	R/W	0x0	When cleared, gpio_out signal is driven by a bit in RGPIO_OUT register. When set, gpio_out signal is driven by corresponding gpio_aux input.

Table 7. Auxiliary Inputs Register

Register RGPIO_CTRL description

Control bits in RGPIO_CTRL register control operation of entire GPIO core as opposed to bits in all other registers that control only individual general-purpose I/O signals.

Bit #	Access	Reset	Description
0	R/W	0	INTE When set, interrupt generation is enabled. When cleared, interrupts are masked.
1	R/W	0	INTS When set, interrupt is pending. When cleared, no interrupt pending.

Table 8. Control Register

Register RGPIO_INTS description

RGPIO_INTS register is interrupt status register for GPIO inputs. Bits in RGPIO_INTS are set by GPIO core when corresponding inputs meet RGPIO_PTRIG criteria and cause an interrupt.
To de-assert an interrupt request, CPU must clear RGPIO_INTS register.

Bit #	Access	Reset	Description
1 - 32	R/W	0x0	When set, input caused an interrupt. When cleared, no interrupt is pending for corresponding input.

Table 9. Interrupt Status Register

Register RGPIO_ECLK description

Bit #	Access	Reset	Description
1 - 32	R/W	0x0	When set, gpio_eclk signal is used to latch appropriate general-purpose input into RGPIO_IN register. When cleared, system clock is used to latch input signal.

Table 10. ECLK Register

Register RGPIO_NEC description

Bit #	Access	Reset	Description
1 - 32	R/W	0x0	When set, gpio_eclk is active on negative edge. When cleared, gpio_eclk is active on positive edge. This bit has no function when appropriate bit in RGPIO_ECLK is cleared.

Table 11. NEC Register

5 IO ports

GPIO IP core has three interfaces. Figure 3 below shows all three interfaces:

- APB host interface
- Auxiliary inputs interface
- Interface to external I/O cells and pads

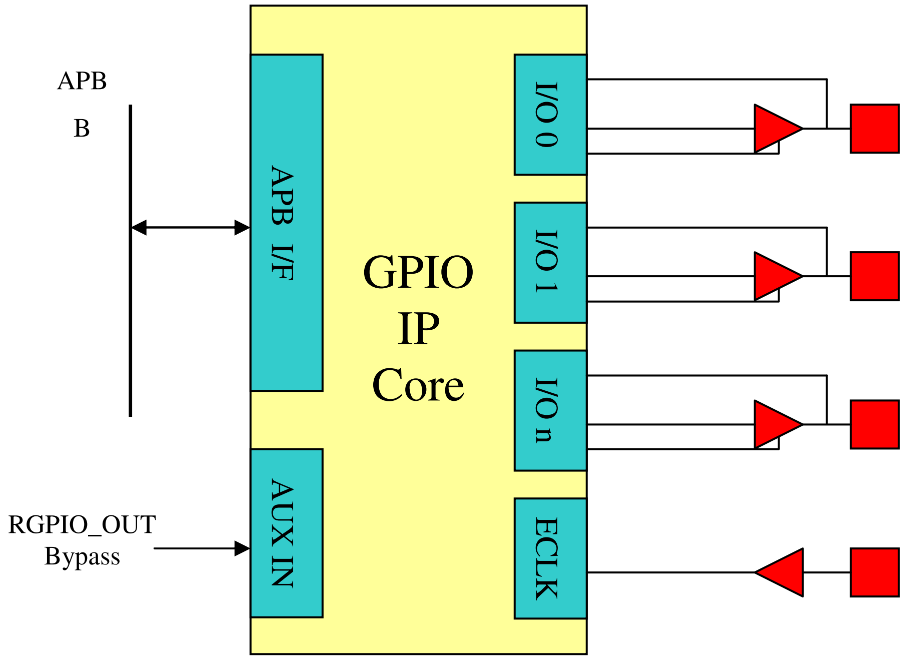


Figure 3. Core Interfaces

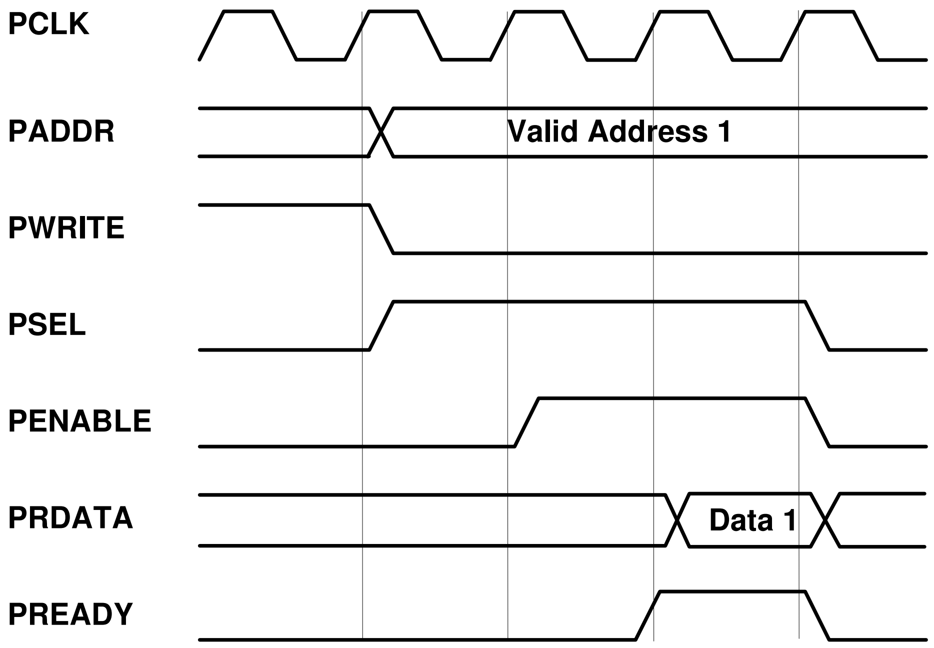
APB host interface

The host interface is a 32 bit APB compliant slave interface. When it needs the intervention of the local microcontroller, it will assert IRQ.

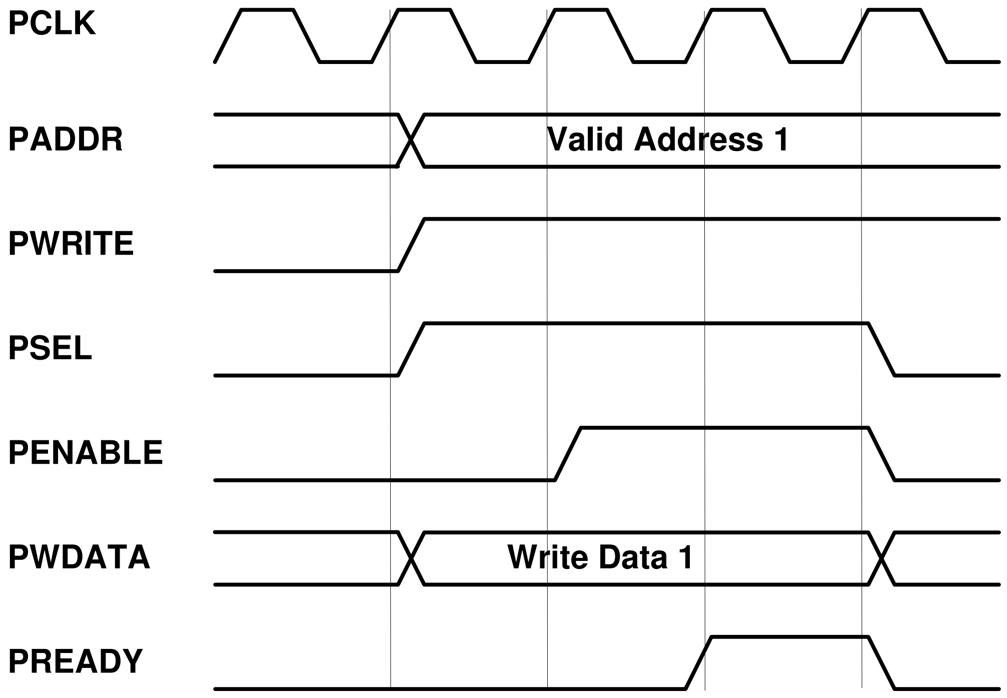
Port	Width	Direction	Description
PCLK	1	Input	APB Bus clock
PRESETn	1	Input	Asynchronous Reset
PADDR	32	Input	Address Bus
PSEL	1	Input	APB Peripheral Select signal
PWDATA	32	Input	Data input bus
PRDATA	32	Output	Data output bus
PWRITE	1	Input	Write enable
PENABLE	1	Input	Enables transfer cycle
PREADY	1	Output	Peripheral ready to complete transfer
PCLK	1	Input	APB Bus clock
PRESETn	1	Input	Asynchronous Reset
PADDR	32	Input	Address Bus
IRQ	1	Output	Interrupt output

Table 10. APB Interface Signals

The timing diagrams for the APB bus interface are as follows:



APB Read Cycle Timing – With one wait state



APB Write cycle timing – with one wait state

Auxiliary inputs

The auxiliary inputs can bypass RGPIO_OUT outputs based on programming of RPGIO_AUX register. Auxiliary inputs are used to multiplex other on-chip peripherals on GPIO pins.

Port	Width	Direction	Description
aux_i	1 - 32	Inputs	GPIO auxiliary inputs

Table 11. Auxiliary input signals

Interface to external I/O cells and pads

External interface connects GPIO core to external I/O ring cells and pads. To support open-drain or three-state outputs, I/O cells with open-drain or three-state support must be used.
Part of external interface is also ECLK signal. It can be used to register inputs based on external clock reference.

Port	Width	Direction	Description
in_pad_i	1 – 32	Inputs	GPIO inputs
out_pad_o	1 – 32	Outputs	GPIO outputs
oen_padoen_o	1 – 32	Outputs	GPIO output drivers enables (for three-state or open-drain drivers)
ext_clk_pad_i	1	Input	Alternative GPIO inputs' latch clock

Table13. External interface

A

Core HW Configuration

This section describes parameters that are set by the user of the core and define configuration of the core. Parameters must be set by the user before actual use of the core in simulation or synthesis.

```
//
// Number of GPIO I/O signals
//
// This is the most important parameter of the GPIO IP core. It defines how many
// I/O signals core has. Range is from 1 to 32. If more than 32 I/O signals are
// required, use several instances of GPIO IP core.
//
// Default is 16.
//
`define GPIO_IOS 31

//depending on number of GPIO_IOS, define this...
// for example: if there is 26 GPIO_IOS, define GPIO_LINES26
//

`define GPIO_LINES31

//
// Undefine this one if you don't want to remove GPIO block from your design
// but you also don't need it. When it is undefined, all GPIO ports still
// remain valid and the core can be synthesized however internally there is
// no GPIO funationality.
//
// Defined by default (duhh !).
//
`define GPIO_IMPLEMENTED

//
// Define to register all WISHBONE outputs.
//
// Register outputs if you are using GPIO core as a block and synthesizing
// and place&routing it separately from the rest of the system.
//
// If you do not need registered outputs, you can save some area by not defining
// this macro. By default it is defined.
//
`define GPIO_REGISTERED_WB_OUTPUTS

//
// Define to register all GPIO pad outputs.
//
// Register outputs if you are using GPIO core as a block and synthesizing
// and place&routing it separately from the rest of the system.
//
// If you do not need registered outputs, you can save some area by not defining
// this macro. By default it is defined.
//
`define GPIO_REGISTERED_IO_OUTPUTS

//
// Define to avoid using negative edge clock flip-flops for external clock
// (caused by NEC register. Instead an inverted external clock with
// positive edge clock flip-flops will be used.
//
```

```
// By default it is not defined.
//
//`define GPIO_NO_NEGEDGE_FLOPS

//
// If GPIO_NO_NEGEDGE_FLOPS is defined, a mux needs to be placed on external clock
// clk_pad_i to implement RGPIO_CTRL[NEC] functionality. If no mux is allowed on
// clock signal, enable the following define.
//
// By default it is not defined.
//
//`define GPIO_NO_CLKPAD_LOGIC

//
// Undefine if you don't need to read GPIO registers except for RGPIO_IN register.
// When it is undefined all reads of GPIO registers return RGPIO_IN register. This
// is usually useful if you want really small area (for example when implemented in
// FPGA).
//
// To follow GPIO IP core specification document this one must be defined. Also to
// successfully run the test bench it must be defined. By default it is defined.
//
`define GPIO_READREGS

//
// Full WISHBONE address decoding
//
// It is is undefined, partial WISHBONE address decoding is performed.
// Undefine it if you need to save some area.
//
// By default it is defined.
//
`define GPIO_FULL_DECODE

//
// Strict 32-bit WISHBONE access
//
// If this one is defined, all WISHBONE accesses must be 32-bit. If it is
// not defined, err_o is asserted whenever 8- or 16-bit access is made.
// Undefine it if you need to save some area.
//
// By default it is defined.
//
//`define GPIO_STRICT_32BIT_ACCESS
//
`ifdef GPIO_STRICT_32BIT_ACCESS
`else
// added by gorand :
// if GPIO_STRICT_32BIT_ACCESS is not defined,
// depending on number of gpio I/O lines, the following are defined :
// if the number of I/O lines is in range 1-8,   GPIO_WB_BYTES1 is defined,
// if the number of I/O lines is in range 9-16,  GPIO_WB_BYTES2 is defined,
// if the number of I/O lines is in range 17-24, GPIO_WB_BYTES3 is defined,
// if the number of I/O lines is in range 25-32, GPIO_WB_BYTES4 is defined,

`define GPIO_WB_BYTES4
`endif

//
// WISHBONE address bits used for full decoding of GPIO registers.
//
`define GPIO_ADDRHH 7
`define GPIO_ADDRHL 6
`define GPIO_ADDRLH 1
`define GPIO_ADDRLL 0

//
// Bits of WISHBONE address used for partial decoding of GPIO registers.
//
// Default 5:2.
//
`define GPIO_OFS_BITS  `GPIO_ADDRHL-1:`GPIO_ADDRLH+1

//
// Addresses of GPIO registers
```

```
//
// To comply with GPIO IP core specification document they must go from
// address 0 to address 0x18 in the following order: RGPIO_IN, RGPIO_OUT,
// RGPIO_OE, RGPIO_INTE, RGPIO_PTRIG, RGPIO_AUX and RGPIO_CTRL
//
// If particular register is not needed, it's address definition can be omitted
// and the register will not be implemented. Instead a fixed default value will
// be used.
//
`define GPIO_RGPIO_IN           4'h0  // Address 0x00
`define GPIO_RGPIO_OUT        4'h1    // Address 0x04
`define GPIO_RGPIO_OE           4'h2  // Address 0x08
`define GPIO_RGPIO_INTE               4'h3    // Address 0x0c
`define GPIO_RGPIO_PTRIG      4'h4    // Address 0x10
`define GPIO_RGPIO_AUX        4'h5    // Address 0x14
`define GPIO_RGPIO_CTRL               4'h6    // Address 0x18
`define GPIO_RGPIO_INTS               4'h7    // Address 0x1c
`define GPIO_RGPIO_ECLK   4'h8  // Address 0x20
`define GPIO_RGPIO_NEC    4'h9  // Address 0x24

//
// Default values for unimplemented GPIO registers
//
`define GPIO_DEF_RGPIO_IN     `GPIO_IOS'h0
`define GPIO_DEF_RGPIO_OUT    `GPIO_IOS'h0
`define GPIO_DEF_RGPIO_OE     `GPIO_IOS'h0
`define GPIO_DEF_RGPIO_INTE   `GPIO_IOS'h0
`define GPIO_DEF_RGPIO_PTRIG  `GPIO_IOS'h0
`define GPIO_DEF_RGPIO_AUX    `GPIO_IOS'h0
`define GPIO_DEF_RGPIO_CTRL   `GPIO_IOS'h0
`define GPIO_DEF_RGPIO_ECLK `GPIO_IOS'h0
`define GPIO_DEF_RGPIO_NEC `GPIO_IOS'h0


//
// RGPIO_CTRL bits
//
// To comply with the GPIO IP core specification document they must go from
// bit 0 to bit 1 in the following order: INTE, INT
//
`define GPIO_RGPIO_CTRL_INTE          0
`define GPIO_RGPIO_CTRL_INTS          1
```