GPIO v1.0

IP User Guide

April 2025

Licensing Notice

GPIO v1.0 is a fully tested, portable, configurable, and synthesisable soft IP core provided by **Chipmunk Logic™**. The IP source files are complied with IEEE VHDL/Verilog/System-Verilog standards. All the source codes are open-source licensed and hence may be used, modified, and shared without any restrictions or conflicts of interest with the original developer.

This IP core is provided 'as is,' without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and non-infringement. **Chipmunk Logic™** shall not be liable or held accountable for any loss or damage (direct or indirect) resulting from the use of any product, as the designs are not intended to be fail-safe or for use in any application requiring fail-safe performance. Hence, the user shall assume sole risk and liability for the use of any of our products in any of their applications.

Acronyms

APB	Advanced Peripheral Bus
ASIC	Application Specific Integrated Circuit
CDC	Clock Domain Crossing
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input Output
IP	Intellectual Property
ISR	Interrupt Service Routine
2-FF	Two-Flop

Table of Contents

1. GPIO

The GPIO IP core is designed to control general-purpose I/Os through APB interface. The core is highly configurable based on the number of I/Os required by the application. The core is also configurable to be interrupt-capable at inputs.

2. Features

- Configurable I/O channel width (1-32).
- Configurable as dedicated I/P or O/P channel or combination of both.
- Configurable as tristate I/O channel, which supports dynamic configuration of each I/O as input or output.
- Control and status registers to set/clear outputs and store the state of each I/O.
- Supports configuring input pins as interrupt sources (Level/Edge sensitive).
- Control and status registers to set/clear/mask the interrupt.
- APB interface to access the control and status registers.

3. Overview

The GPIO IP core provides an interface to control the direction of the general-purpose I/O pins, the data sent to the output pins, and the state of the input pins.

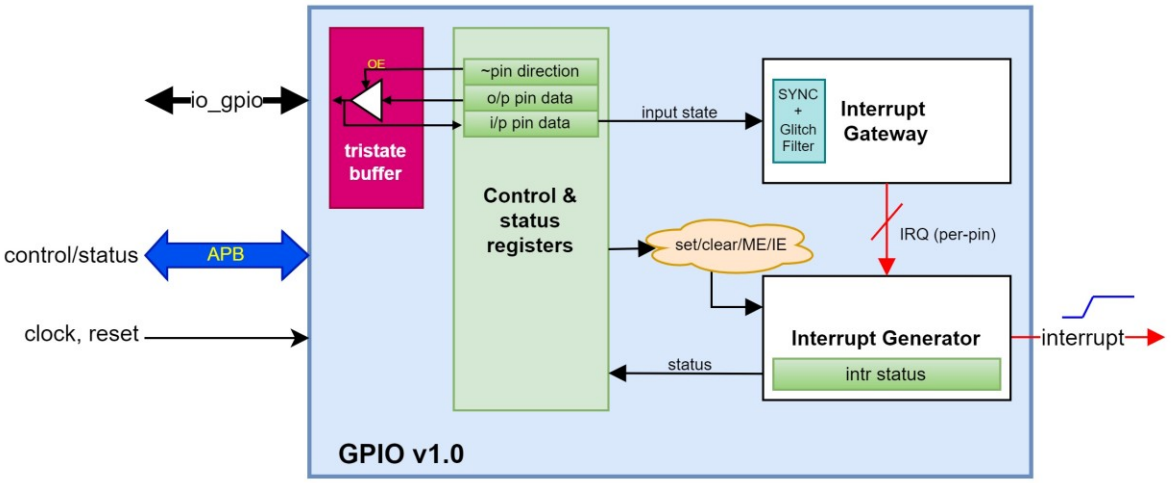


Figure 3.1: GPIO core

The different interfaces/sub-blocks in the core are described below.

- **APB interface**: This is the bus interface to access the control and status registers of the core.
- **GPIO interface**: The general-purpose I/O pins controlled by the core.
- **Interrupt**: The interrupt generated by the core.
- **Control and Status registers**: The set of registers to control the direction of the I/O pins, the data sent to the output pins, the state of the input pins, and the interrupt behavior.
- **Clock and Reset**: Core clock and reset.
- **Interrupt Gateway**: Conditions the interrupt event received at the input pin and generates the interrupt request (IRQ) to Interrupt Generator.
- **Interrupt Generator:** Accumulates all the interrupt requests from the input pins and generates the core interrupt.

4. Top-level Parameters/Macros

The GPIO IP core provides a set of configuration parameters and macros to generate customized cores. Table 4.1 lists all top-level parameters/macros used to configure the IP.

Name	Type	Width	Valid values	Description
EN_TRISTATE_BUFF	MACRO	-	-	Define this macro to configure the core in tristate channel mode. Read about different GPIO channel modes here.
EN_IPORTS	MACRO	-	-	Define this macro to generate dedicated input ports in the core. *Not available in tristate channel mode.*
EN_OPORTS	MACRO	-	-	Define this macro to generate dedicated output ports in the core. *Not available in tristate channel mode.*
EN_INTR_IF	MACRO	-	-	Define this macro to enable interrupt capability at all the inputs. *Not available if the core is configured in output channel mode.*
GPIO_WIDTH	PARAM	6	[1-32]	I/O width.
DEF_O_VAL	PARAM	GPIO_WIDTH	-	Default state of each output on reset. 1'b0 = Drives LOW/GND 1'b1 = Drives HIGH
DEF_IO_DIR	PARAM	GPIO_WIDTH	-	Default direction of each tristate I/O on reset. 1'b0 = Output 1'b1 = Input
EN_SYNC	PARAM	GPIO_WIDTH	-	Enable/Disable 2-FF CDC synchronizer at each input. 1'b1 = Enable synchronizer in the path 1'b0 = The input is passed through
EN_GLF	PARAM	GPIO_WIDTH	-	Enable/Disable glitch filter at each synchronized input. *Available only if synchronizer is enabled at the input.*

Table 4.1: Top-level Parameters/Macros

5. Top-level Ports/Interfaces

Table 5.1 lists all top-level I/O ports/interfaces of the IP.

Signal Name	Direction	Width	Description
Clock and Reset			
clk	INPUT	1	Core clock
resetn	INPUT	1	Core reset (Asynchronous active-low)
GPIO Interface			
io_gpio	INOUT	GPIO_WIDTH	Tristate I/O ports. *Available only in tristate channel mode.*
i_gpio	INPUT	GPIO_WIDTH	Dedicated input ports. *Available only in input channel and dual channel modes.*
o_gpio	OUTPUT	GPIO_WIDTH	Dedicated output ports. *Available only in output channel and dual channel modes.*
Interrupt			
o_intr	OUTPUT	1	Interrupt; level type, active-high. *Available only if interrupt capability is enabled.*
APB Slave Interface			
i_paddr	INPUT	8	Address
i_psel	INPUT	1	Select
i_penable	INPUT	1	Enable
i_pwrite	INPUT	1	Write enable
i_pwdata	INPUT	32	Write data
o_prdata	OUTPUT	32	Read data
o_pready	OUTPUT	1	Ready

Table 5.1: Top-level Ports/Interfaces

6. Register Map

The IP has a 32-bit register address space accessible through APB interface. Table 6.1 lists all the control and status registers.

Address offset	Register Name	Access type	Reset value	Description
0x00	gpio_i_data_reg	RO	-	Read Data register. Contains the data read from the input pins. Bits[GPIO_WIDTH-1:0] = Data Bits[31:GPIO_WIDTH] = 0 *In output channel mode, this register is unused and always read as 0.*
0x04	gpio_o_data_reg	RW	DEF_O_VAL	Write Data register. Contains the data driven to the output pins. Bits[GPIO_WIDTH-1:0] = Data Bits[31:GPIO_WIDTH] = 0 *In input channel mode, this register is unused and always read as DEF_O_VAL.*
0x08	gpio_io_dir	RW	DEF_IO_DIR	I/O Direction register. Configures the direction of each tristate I/O pin. Bits[GPIO_WIDTH-1:0] = I/O direction Bits[31:GPIO_WIDTH] = 0 For each bit: 1'b0 = Output 1'b1 = Input *Used only in tristate channel mode. In all other modes, this register is unused.*
0x0C	gpio_glb_intr_ctrl_reg	RW	0x0	Global Interrupt Control register. Controls the global behavior of interrupts in the core. Bit[0] = Global Interrupt Enable (GIE) Bit[1] = Global Interrupt Mask Enable (GME) Bits[31:2] = 0 *Available only if interrupt capability is enabled.*
0x10	gpio_intr_en_reg	RW	0x0	Interrupt Enable (per-pin) register. Enable/disable interrupt at each input pin. Bits[GPIO_WIDTH-1:0] = Interrupt Enable (IE) Bits[31:GPIO_WIDTH] = 0

				For each bit: 1'b0 = Interrupt disabled 1'b1 = Interrupt enabled *Available only if interrupt capability is enabled.*
0x14	gpio_intr_mask_reg	RW	32'h0	Interrupt Mask (per-pin) register. The interrupt mask of each input pin. Bits[GPIO_WIDTH-1:0] = Interrupt Mask Enable (ME) Bits[31:GPIO_WIDTH] = 0 For each bit: 1'b0 = Interrupt unmasked 1'b1 = Interrupt masked *Available only if interrupt capability is enabled.*
0x18	gpio_intr_type_reg	RW	32'h0	Interrupt Type register. The type of interrupt source at each input pin. Bits[GPIO_WIDTH-1:0] = Interrupt type Bits[31:GPIO_WIDTH] = 0 For each bit: 1'b0 = Level-triggered 1'b1 = Edge-triggered *Available only if interrupt capability is enabled.*
0x1C	gpio_intr_evnt_reg	RW	32'h0	Interrupt Event register. The active event of interrupt source at each input pin. Bits[GPIO_WIDTH-1:0] = Interrupt event Bits[31:GPIO_WIDTH] = 0 For each bit: 1'b0 = Active-low for Level-triggered interrupts, Falling edge for edge-triggered interrupts. 1'b1 = Active-high for Level-triggered interrupts, Rising edge for edge-triggered interrupts. *Available only if interrupt capability is enabled.*
0x20	gpio_intr_sts_reg	RO	32'h0	Interrupt Status register. The status of the interrupt from each input pin after masking. If any bit in this register is set, then the interrupt is set. To clear the interrupt status of a pin, write 1 to the corresponding bit in the register.

				Bits[GPIO_WIDTH-1:0] = Interrupt status Bits[31:GPIO_WIDTH] = 0 For each bit: 1'b0 = Interrupt status is cleared 1'b1 = Interrupt status is set *Available only if interrupt capability is enabled.*
0x24	gpio_intr_sts_raw_reg	RO	32'h0	Interrupt Status (Raw) register. The raw status of the interrupt from each input pin before masking. To clear the raw interrupt status of a pin, write 1 to the corresponding bit in the Interrupt Status register, gpio_intr_sts_reg. Bits[GPIO_WIDTH-1:0] = Interrupt status Bits[31:GPIO_WIDTH] = 0 For each bit: 1'b0 = Interrupt status (Raw) is cleared 1'b1 = Interrupt status (Raw) is set *Available only if interrupt capability is enabled.*
0x28	gpio_intr_set_rg	WO	32'h0	Interrupt Set register. Provides software control to set the interrupt status of a pin by writing 1 to the corresponding bit in the register. Writing 0 to this register has no effect. Bits[GPIO_WIDTH-1:0] = Interrupt set control Bits[31:GPIO_WIDTH] = 0 *Available only if interrupt capability is enabled.*

7. Designing with the IP

This chapter discusses the guidelines, including clocking and reset, configuration, and other considerations while designing with the IP.

7.1 GPIO channel modes

The IP can be configured in one of four channel modes during setup: tristate channel mode, input channel mode, output channel mode, and dual channel mode.

- **Tristate channel mode**: All the I/Os of the core are bi-directional and tristate. The I/Os are dynamically configurable as input pins or output pins through register configuration.
- **Input channel mode**: The core has dedicated input pins.
- **Output channel mode**: The core has dedicated output pins.
- **Dual channel mode**: The core has dedicated input and output pins.

All the modes except output channel mode support interrupts.

7.2 Clock and Reset

The core clock, `clk`, synchronizes the operation of the core. The core reset, `resetn`, is asynchronous and active-low. The reset assertion is asynchronous, but the de-assertion should be externally synchronized to `clk`.

Clock and Reset Sequencing

1. Assert the core reset.
2. Bring up the core clock.
3. Assert the core reset for at least 8 clock cycles.
4. Release the reset.
5. The core is now ready for use.

7.3 Controlling the I/Os

If the core is configured in the tristate channel mode, the I/O direction register, `gpio_io_dir`, should be configured first to set the direction of each pin. In all other modes, the direction register does not need to be configured. The GPIO data registers store/control the data driven at the I/O pins.

Reading the state of an I/O pin

1. Read the corresponding bit in `gpio_i_data_reg`.

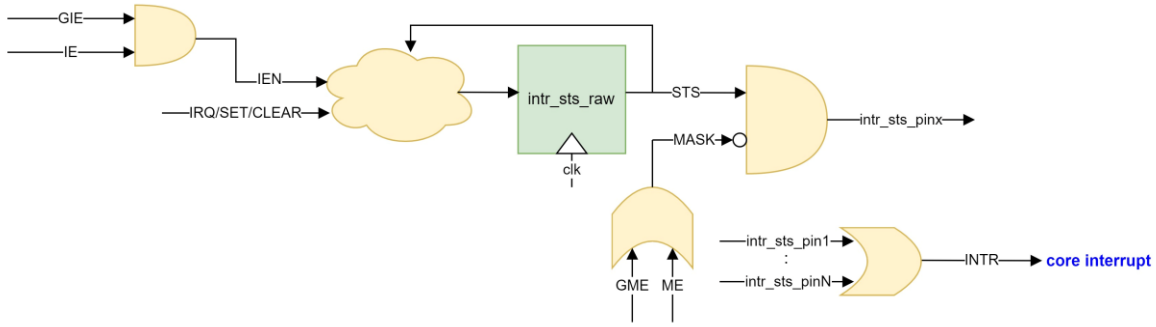
Driving an I/O pin

1. Write the corresponding bit in `gpio_o_data_reg`.

Note: In output channel mode, the state of the output pins should always be read from `gpio_o_data_reg`. The register, `gpio_i_data_reg`, doesn't store the state of dedicated output pins.

7.4 Interrupts

The core treats the input pins as interrupt sources when the interrupt capability is enabled. If the interrupt is enabled for a pin, an interrupt event (IRQ) at the pin (Level/Edge) will set the interrupt status of that pin. If any bit is set in the interrupt status and the interrupt is unmasked, the core will set the interrupt at `o_intr`. The interrupt status can be cleared by writing 1 to the corresponding bit in the interrupt status register, `gpio_intr_sts_reg`.



Configuring the core for interrupts

1. Configure the interrupt type in `gpio_intr_type_reg`.
2. Configure the interrupt event in `gpio_intr_evnt_reg`.
3. Configure the GIE and GME in `gpio_glb_intr_ctrl_reg`.
4. Configure the IE and the mask in `gpio_intr_en_reg` and `gpio_intr_mask_reg`.
5. The core is ready to receive interrupts at input pins.

Interrupt Enable and Interrupt Mask

The following summarizes the behavior of the IE and the mask of the input pins.

1. The interrupt status of a pin can be set only if the interrupt is enabled (IE = 1) for the pin and an interrupt event occurs at the pin.
2. The interrupt status sets the core interrupt only if the interrupt is unmasked (ME = 0).
3. If the interrupt is disabled for a pin (IE = 0), any existing status is cleared, and future interrupt events at the pin are ignored.
4. Masking (ME = 1) doesn't clear the interrupt status. In the masked state, an interrupt event at the pin can still set the interrupt status. It can be unmasked at any time to set the core interrupt.
5. The system/host should set the mask (rather than setting IE = 0), if the intention is to temporarily disable the interrupt without losing the interrupt status.

Software-set Interrupts

The core also supports generating software-set (via APB interface) interrupts. These interrupts are generated by setting the interrupt status of a pin by writing 1 to the corresponding bit in `gpio_intr_set_rg`. The interrupt status is set only if the interrupt is enabled for the corresponding pin.

8. Integrating the IP

8.1 APB bus integration

The core can be easily plugged into a bus that complies with a standard 32-bit APB interface. The write accesses are of 2 cycles and read accesses are of 3 cycles.

8.2 Interrupt handling

The core generates an active-high level interrupt, which the bus master or host should acknowledge. The interrupt handler in the host can identify the source of the interrupt by reading the interrupt status register, and execute the corresponding ISR. Before exiting the ISR, the source should be acknowledged by writing 1 to the corresponding bit in the status register. This will clear the interrupt status of the source.

9. Testing the IP

The GPIO IP core can be tested with the test bench provided with the IP package. It configures the core in 4-bit tristate channel mode, with interrupts enabled.

The test bench dynamically configures the I/Os to two input and two output ports. It stimulates and drives the input port to trigger an interrupt. The test bench's ISR toggles the corresponding output port in response to each interrupt.

- io_gpio[3] → input, interrupt-capable, edge-triggered
- io_gpio[2] → input, interrupt-capable, level-triggered
- io_gpio[1] → output, toggles when io_gpio[3] receives interrupt
- io_gpio[0] → output, toggles when io_gpio[2] receives interrupt

The test bench supports two modes: synthesis and simulation modes. In simulation mode, the clock and reset, and stimulus are automatically driven. In synthesis mode, the test bench is synthesisable and can be tested on an FPGA board with an external clock, reset, and I/O connections.

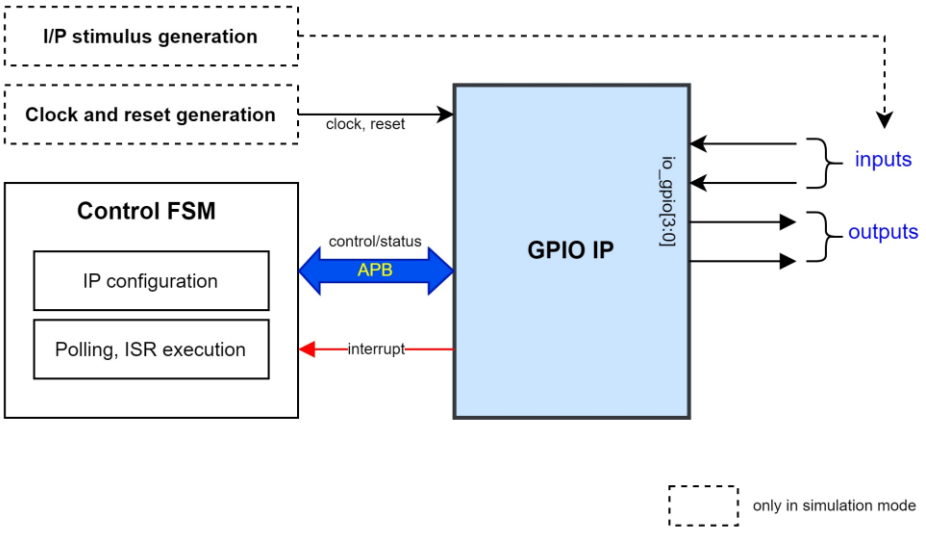


Figure 9.1: Test setup

10. Application Notes

1. If the core is configured in tristate channel mode and targeted on FPGAs, the I/O pins should automatically get mapped to IOBs, which support tristate buffered I/Os. If ASIC, you may need to instantiate appropriate library cells inside the tristate buffer wrapper.
2. If interrupt capability is enabled and the input signals at the pins are asynchronous to the core clock domain, it's recommended to enable synchronizers during the core generation.

11. Known Limitations/Issues

[NOT APPLICABLE]

References

[NOT APPLICABLE]

Appendix

a) FPGA Resource Utilization

IP configuration	Tristate channel mode, 8-bit wide I/O, interrupts enabled. Synchronizers + Glitch filter is also enabled at all I/Os. EN_TRISTATE_BUFF, EN_INTR_IF = defined GPIO_WIDTH = 8 DEF_O_VAL = 8'h00 DEF_IO_DIR = 8'hFF EN_SYNC = 8'hFF EN_GLF = 8'hFF
FPGA Targeted	Xilinx Basys-3 (XC7A-35T-CPG236-1), Artix-7 FPGA based board
Synthesiser	Vivado 2019.2
Targeted clock frequency	100 MHz
LUTs	89
Registers	132

b) Test Summary

FPGA Targeted	Xilinx Basys-3 (XC7A-35T-CPG236-1), Artix-7 FPGA based board
Synthesiser	Vivado 2019.2
Core clock	100 MHz
GPIO Channel modes tested	1. Tristate channel mode with/without interrupt. 2. Input channel mode with/without interrupt. 3. Output channel mode. 4. Dual channel mode with/without interrupt
Interrupt types tested	Level, Edge; both polarities
Test result	Successfully passed

Revision History

The following tables shows the revision history of this document.

Date	Version	Revision
Apr-2025	1.0	• Initial version

GPIO v1.0

An open-source licensed soft IP core

Developer : **Mitu Raj**
Vendor : **Chipmunk Logic™**, chip@chipmunklogic.com
Website : **chipmunklogic.com**