GPIO IP Core

User Guide

FPGA-IPUG-02076-2.0

April 2021

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
APB	Advanced Peripheral Bus
LMMI	Lattice Memory Mapped Interface
RTL	Register Transfer Level
GPIO	General Purpose Input/Output

1. Introduction

The General Purpose Input/Output (GPIO) peripheral soft IP is designed to control GPIOs through the Lattice Memory Mapped Interface (LMMI) or the Advanced Peripheral Bus Interface (APB). When configured as an input, it can detect the state of a GPIO by reading the state of the associated register. When configured as an output, it takes the value written into the associated register and controls the state of the controlled GPIO.

The IP can be attached to a CPU bus or used in bridges/peripherals needing memory organization of the I/O. The IP generator is configurable based on the number of GPIOs for a flexible use of the GPIO ports.

The design is implemented in Verilog HDL. The IP can be configured, generated and implemented based on Table 1.1.

Table 1.1. FPGA Software for IP Configuration, Generation and Implementation

Supported FPGA Family	IP Configuration and Generation	IP Implementation (Synthesis, Map, Place and Route)
iCE40 UltraPlus™	Lattice Radiant™ software	Lattice Radiant software
MachXO2™	Lattice Propel™ Builder software	Lattice Diamond® software
MachXO3™	Lattice Propel Builder software	Lattice Diamond software
MachXO3D™	Lattice Propel Builder software	Lattice Diamond software
Mach™-NX	Lattice Propel Builder software	Lattice Diamond software
CrossLink™-NX	Lattice Radiant software	Lattice Radiant software
Certus™-NX	Lattice Radiant software	Lattice Radiant software

1.1. Features

The key features of the GPIO peripheral IP Core are:

- Setting or clearing an output through separate registers to allow parallel control of the output
- Setting or clearing an output through a single register
- Separate input and output data and control registers
- Output register reflects the output driven status
- Input register reflects the input status
- All inputs are configurable as INT source with configurable edge or level detection
- Interrupts conform to the Lattice Interrupt Interface (LINTR)

1.2. Conventions

1.2.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.2.2. Signal Names

Signal Names that end with:

- *_n* are active low
- *_i* are input signals
- *_o* are output signals
- *_io* are bi-directional input/output signals

1.2.3. Host

The logic unit inside the FPGA interacts with the GPIO IP Core through either APB or LMMI.

1.2.4. Attribute

The names of attributes in this document are formatted in title case and italicized (*Attribute Name*).

2. Functional Description

2.1. Overview

The GPIO IP provides a dedicated interface to configure each GPIO as either an input or an output. When configured as an input, it can detect the state of a GPIO by reading the state of the associated register. When configured as an output, it takes the value written into the associated register and controls the state of the controlled GPIO.

Figure 2.1 shows the architecture of the GPIO IP.

When APB is selected as the interface from the user interface, the APB to LMMI bridge is instantiated and the LMMI is replaced by APB interface. This optional bridge is implemented to interface the GPIO IP in an APB system while preserving the internal LMMI for writing and reading to internal registers.

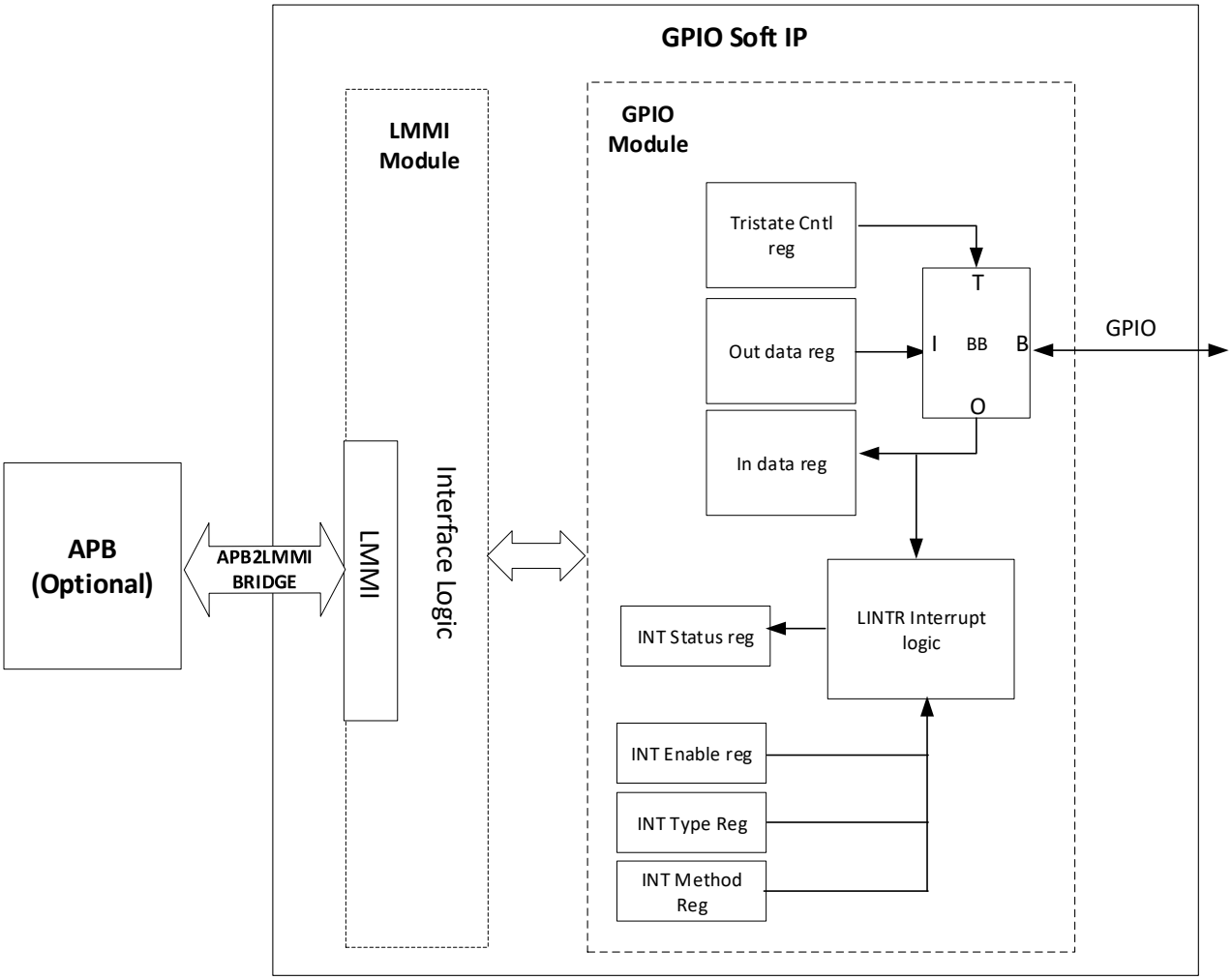

Figure 2.1. GPIO Top Level Block Diagram

Figure 2.2 shows the GPIO configuration interface.

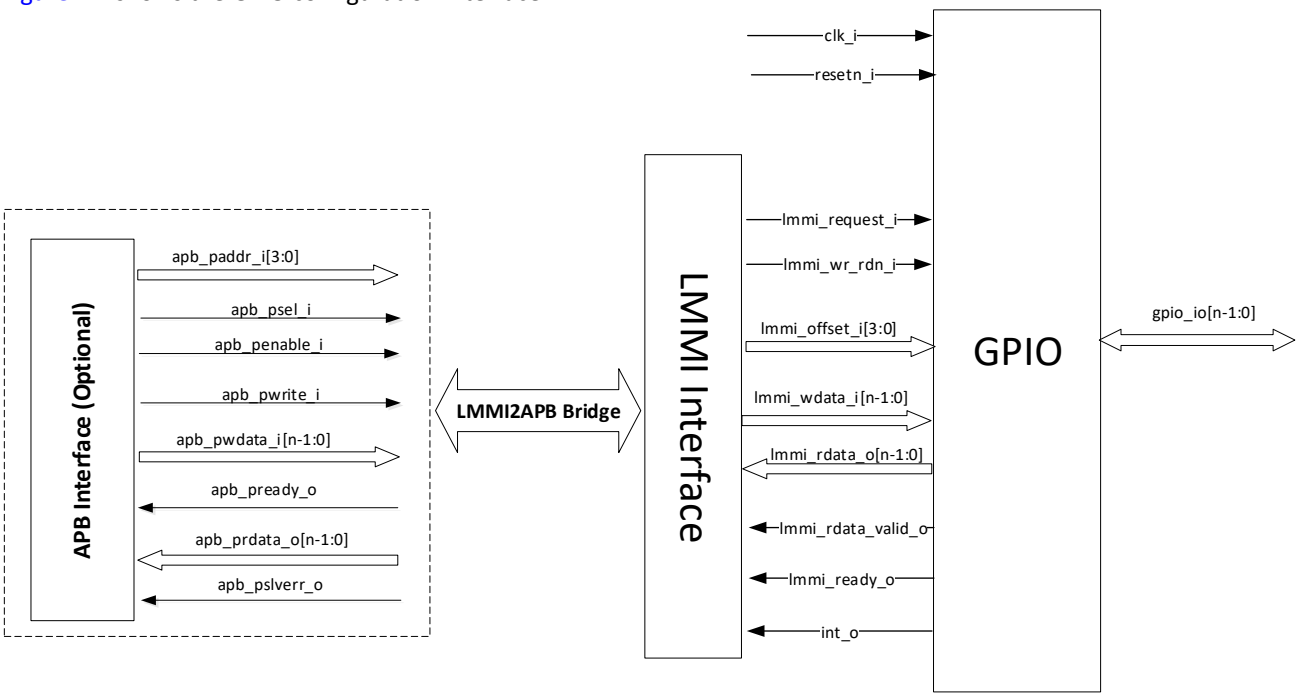

Figure 2.2. GPIO Configuration Interface

Each Input and Output Port is independently configurable. Each Input Port bit can be programmed to enable interrupt on the Input Edge (Rising or Falling) or on the Level (High or Low). Each Output Port's default value can be configured from the user interface to take either *0* or *1* at the time of GPIO generation. Value for n varies from 1 to 32 depending on the *Number of I/O Lines* entry from the user interface.

2.2. Signal Descriptions

Table 2.1 lists the top-level input and output signals for the GPIO IP.

Table 2.1. GPIO Signal Descriptions

Port Name	Direction	Active State	Width (Bits)	Description
Clock and Reset				
clk_i	Input	N/A	1	System Clock
resetn_i	Input	Low	1	Reset signal Resets LMM/APB Interface and sets registers to their default values.
GPIO Interface				
gpio_io[n-1:0]	Input/Output	—	1–n*	General purpose input/output pins The width of this port is configurable from the user interface based on the GPIO width. Available when *Remove Tri-State Buffer* is Unchecked
gpio_i[n-1:0]	Input		1–n	Dedicated input pin available when *Remove Tri-State Buffer* is Checked. The width of this port is configurable from the user interface based on the GPIO width.
gpio_o[n-1:0]	Output		1–n	Dedicated output pin available when *Remove Tri-State Buffer* is Checked. The width of this port is configurable from the user interface based on the GPIO width.
gpio_en_o[n-1:0]	Output		1–n	Direction output pin available when *Remove Tri-State Buffer* is Checked. The width of this port is configurable from the user interface based on the GPIO width.
LMMI				
lmmi_request_i	Input	High	1	Start transaction signal
lmmi_wr_rdn_i	Input	High/Low	1	Write = High Read = Low
lmmi_offset_i[3:0]	Input	N/A	4	Offset signal Offsets Slave registers, starting at offset 0
lmmi_wdata_i[n-1:0]	Input	N/A	1–n*	Write data
lmmi_rdata_o[n-1:0]	Output	N/A	1–n*	Read data
lmmi_rdata_valid_o	Output	High	1	Read transaction is complete and lmmi_rdata_o[] contains valid data.
lmmi_ready_o	Output	High	1	Slave is ready to start a new transaction. Slave can insert wait states by holding this signal low. For this particular soft IP, the signal is always high.
LINTR Interface				
int_o	Output	High	1	Interrupt Stays high as long as any enabled interrupt is pending.
APB Interface				
apb_paddr_i[5:0]	Input	N/A	4	APB Address signal
apb_psel_i	Input	High	1	APB Select signal
apb_penable_i	Input	High	1	APB Enable signal
apb_pwrite_i	Input	High	1	APB Direction signal
apb_pwdata_i[31:0]	Input	N/A	32	APB Write Data signal
apb_pready_o	Output	High	1	APB Ready signal
apb_prdata_o[31:0]	Output	N/A	32	APB Read Data signal
apb_pslverr_o	Output	High	1	APB Slave Error signal

***Note:** n – GPIO port width; value varies from 1 to 32.

2.3. Attribute Summary

Configuration summary for the GPIO IP is shown in Table 2.2. Attributes are specified using GPIO IP Catalog user interface in Lattice Radiant software.

Table 2.2. Attributes Table[1]

Attribute	Selectable Values	Default	Dependency on Other Attributes	Description
Number of I/O Lines [1-32]	1–32	1	None	Specifies the bit width of registers and corresponding I/O signal.
Remove Tri-State Buffer	Checked, Unchecked	Unchecked	None	This removes the tri-state buffer when checked.
Initial Output Value (hex) [0 - FFFFFFFF]	0-FFFFFFFF	0	Must not be greater than the *Number of I/O Lines*	This value is stored in the Output Register during the reset. Each bit corresponds to an I/O line. Specifies the reset value of WR_DATA_REG.
Initial Output Binary Value	Calculated	0	Display only. *Initial Output Value (hex)* is converted to binary value.	*Initial Output Value (hex)* is converted to binary value. Each bit corresponds to an I/O line.
I/O Direction (hex) [0 - FFFFFFFF]	0-FFFFFFFF	1	None	This value is stored in the Direction Register during the reset. Each bit corresponds to an I/O line. Specifies the reset value of DIRECTION_REG.
I/O Direction Encoded Value[2]	Calculated	o	Must not be greater than the *Number of I/O Lines.* Display only. *I/O Direction (hex)* is encoded value.	The value *o* corresponds to Output port and *i* corresponds to Input port.[2]
Interface	LMMI, APB	APB	None	User defines needed interface type from the user interface. Reconfigurable to APB only if the device used is LIFCL-40, LIFCL-17, and LFD2NX-40 and always LMMI when the device is iCE40 UltraPlus.

Notes:

1. All attributes can be configured from the General tab of the Lattice Radiant software user interface.
2. If I/O Direction is, for example, *0xC*, the I/O Direction Encoded Value is displayed as *ooii*, that is, the first two ports of GPIO are outputs, while the last two are inputs.

2.4. Register Description

The GPIO data register is used to read the general purpose input ports and write to the general purpose output ports. When no ports are configured as output, writing to the GPIO data register has no effect. Initial values for WR_DATA_REG and DIRECTION_REG registers are configurable and are set and reconfigured from the user interface.

Table 2.3 gives a summary of GPIO data registers.

Table 2.3. Register Address Map

Offset LMMI	Offset APB	Register Name	Access Type	Description
0x00	0x00	RD_DATA_REG	R	Read Data Register.
0x01	0x04	WR_DATA_REG	R/W	Write Data Register.
0x02	0x08	SET_DATA_REG	W	Set Data Register.
0x03	0x0C	CLEAR_DATA_REG	W	Clear Data Register.
0x04	0x10	DIRECTION_REG	R/W	Direction Control Register.
0x05	0x14	INT_TYPE_REG	R/W	Interrupt Type Configure Register.
0x06	0x18	INT_METHOD_REG	R/W	Interrupt Method Configure Register.
0x07	0x1C	INT_STATUS_REG	R/W	Interrupt Status Register.
0x08	0x20	IN_ENABLE_REG	R/W	Interrupt Enable Register.
0x09	0x24	INT_SET_REG	W	Interrupt Set Register.

The behavior of registers to write and read access is defined by its access type, which is defined in Table 2.4.

Table 2.4. Access Type Definition

Access Type	Behavior on Read Access	Behavior on Write Access
R	Returns register value	Ignores write access
W	Returns 0	Updates register value
R/W	Returns register value	Updates register value

The Receive Buffer Register is the interface to the Receiver Buffer/FIFO (RCVR FIFO). Reading from this register pops and returns the output data in the RCVR FIFO. If read is performed during RCVR FIFO empty, the last data in the FIFO is returned.

2.4.1. Read Data Register (RD_DATA_REG)

Reading the Read Data Register returns the data from the input pins. The bit width of this register is set by the *Number of I/O Lines* attribute as shown in Table 2.5. Reset value is not observable because value is updated immediately after reset.

Table 2.5. Read Data Register

Field	Name	Access	Width	Reset
[*Number of I/O Lines-1*:0]	rd_data	R	*Number of I/O Lines*	NA

2.4.2. Write Data Register (WR_DATA_REG)

Writing in the Write Data Register changes the data of the output pins. The bit width of this register is set by the *Number of I/O Lines* attribute as shown in Table 2.6.

Table 2.6. Write Data Register

Field	Name	Access	Width	Reset
[*Number of I/O Lines* - 1:0]	wr_data	R/W	*Number of I/O Lines*	*Initial Output Value*

2.4.3. Set Data Register (SET_DATA_REG)

If any bit of the Set Data Register is set to *1*, the corresponding bit of wr_data gets set to *1*. Nothing changes otherwise. The bit width of this register is set by the *Number of I/O* Lines attribute as shown in Table 2.7.

Table 2.7. Set Data Register

Field	Name	Access	Width	Reset
[*Number of I/O Lines* - 1:0]	set_data	W	*Number of I/O Lines*	0

2.4.4. Clear Data Register (CLEAR_DATA_REG)

If any bit of the Clear Data Register is set to *1*, the corresponding bit of wr_data gets cleared (set to *0*. Nothing changes otherwise. The bit width of this register is set by the *Number of I/O Lines* attribute as shown in Table 2.8.

Table 2.8. Clear Data Register

Field	Name	Access	Width	Reset
[*Number of I/O Lines* - 1:0]	clear_data	W	*Number of I/O Lines*	0

2.4.5. Direction Register (DIRECTION_REG)

The Direction Register determines the direction of pins. If any bit of this register is set to *0*, the corresponding pin is configured as an input, otherwise as an output. The bit width of this register is set by the *Number of I/O Lines* attribute as shown in Table 2.9.

Table 2.9. Direction Register

Field	Name	Access	Width	Reset
[*Number of I/O Lines* - 1:0]	direction_reg	R/W	*Number of I/O Lines*	0

2.4.6. Interrupt Type Register (INT_TYPE_REG)

Interrupt Type Registers possible values are: edge (*0*) or level (*1*). The bit width of this register is set by the *Number of I/O Lines* attribute as shown in Table 2.10.

Table 2.10. Interrupt Type Register

Field	Name	Access	Width	Reset
[*Number of I/O Lines* - 1:0]	int_type	R/W	*Number of I/O Lines*	0

2.4.7. Interrupt Method Register (INT_METHOD_REG)

The Interrupt Method Registers values are: rising (*1*) or falling (*0*) in case of the edge type interrupt, high (*1*) or low (*0*) in case of the level type interrupt. The bit width of this register is set by the *Number of I/O Lines* attribute as shown in Table 2.11.

Table 2.11. Interrupt Method Register

Field	Name	Access	Width	Reset
[*Number of I/O Lines* - 1:0]	int_method	R/W	*Number of I/O Lines*	0

2.4.8. Interrupt Status Register (INT_STATUS_REG)

The Interrupt Status Register shows the interrupt status for each input, regardless of whether it is enabled or not. If any bit of this register is set to *1* and the corresponding bit of INT_ENABLE_REG is set as well, interrupt happens on the corresponding input. In order to clear interrupt, you must write *1* to the corresponding bit. The bit width of this register is set by the *Number of I/O Lines* attribute as shown in Table 2.12.

Table 2.12. Interrupt Status Register

Field	Name	Access	Width	Reset
[*Number of I/O Lines* - 1:0]	int_status	R/W	*Number of I/O Lines*	0

2.4.9. Interrupt Enable Register (INT_ENABLE_REG)

In the Interrupt Enable Register, each bit that is set to *1* enables interrupt for the corresponding port in case it is configured as an input. The bit width of this register is set by the *Number of I/O Lines* attribute as shown in Table 2.13.

Table 2.13. Interrupt Enable Register

Field	Name	Access	Width	Reset
[*Number of I/O Lines* - 1:0]	int_enable	R/W	*Number of I/O Lines*	0

2.4.10. Interrupt Set Register (INT_SET_REG)

In the Interrupt Set Register, you can generate interrupt by writing *1* to the corresponding bit of this register. This also sets the corresponding bit of the int_status register to *1*. The bit width of this register is set by the *Number of I/O Lines* attribute as shown in Table 2.14.

Table 2.14. Interrupt Set Register

Field	Name	Access	Width	Reset
[*Number of I/O Lines* - 1:0]	int_set	W	*Number of I/O Lines*	0

2.5. Modules Description

Functions supported by each of the blocks shown in Figure 2.1 are as follows:

- Direction Control – Configures the direction of GPIO (input or output).
- Out Data – Data to be driven into the GPIO pad when the direction is configured as an output.
- Interrupt logic – When GPIO is configured as an interrupt capable input, this logic detects the connection of GPIO as per three Interrupt Control Registers (Enable, Type, Method)
- In data – Reflects the data captured from a GPIO.

2.5.1. Lattice Memory Mapped Interface (LMMI)

The LMMI device module implements memory mapped registers. LMMI is a memory-mapped address/data interface, which supports both single and burst transactions with a maximum throughput of one transaction per clock cycle. The LMMI module is a fully synchronous module that runs off the LMMI clock. Registers must be programmed through LMMI to ensure that the GPIO IP functions as intended. LMMI defines a standard set of interface signals for register/memory access.

2.5.2. Lattice Interrupt (LINTR) Interface

The LINTR consists of an interrupt signal and a set of interrupt registers, which are accessed through LMMI. Each interrupt register has one or more bits, which represent the interrupt sources. The bit position of each interrupt source is the same in every interrupt register.

For detailed information on LMMI/LINTR, see Lattice Memory Mapped Interface and Lattice Interrupt Interface User Guide (FPGA-UG-02039).

2.5.3. General Purpose Input/Output (GPIO) Interface

The GPIO core consists of registers for reading and writing the GPIO channel. It also includes the necessary logic to identify an interrupt event, when the port input changes.

The GPIO IP features are:

- Setting or clearing an output through a single register to allow parallel control of the outputs.
- Setting or clearing an output by writing Set Data and Clear Data registers.
- Output register reflects the output driven status.
- Input register reflects the input status.
- All inputs may be configured as an interrupt source with configurable edge or level detection.

2.6. Programming Flow

2.6.1. Initialization

Initial values for all registers come from the user interface. To change default configuration, the following GPIO registers should be set properly before performing Read or Write operation:

- Direction Register
- Interrupt Type Register
- Interrupt Method Register
- Interrupt Enable Register

In case any of the interrupts are enabled, these must first be cleared by writing *1*s to the corresponding bits of the Interrupt Status Register.

2.6.2. Data Transfer (Transmit/Receive Operation)

Assuming that the module is not currently performing any operation, below are recommended steps for performing a GPIO transaction.

- To read from inputs, read the Read Data Register.
- To write to outputs, write to the Write Data Register.

If an interrupt occurs and you want to clear that interrupt, write *1*s to corresponding bits of the Interrupt Status Register.

2.7. I/O Port Usage

To configure some I/O as input only, some as output only, and the rest as bidirectional, enable Remove Tri-State Buffer attribute and set the I/O Direction attribute accordingly. I/O Direction is fixed for input and output ports. For bidirectional ports, however, this is initial direction only and you can modify the direction dynamically. For input only ports, connect the corresponding gpio_i bits of the IP to the GPIO input ports of the instantiating module. For output only ports, connect the corresponding gpio_o bits of the IP. For bidirectional ports, instantiate a tri-state buffer or implement the tri-state function in the instantiating module or in the top module of the design.

The example below shows how to use GPIO to use some ports as bidirectional while other ports as inputs and outputs only. The configuration setup for GPIO in this example is Number of I/O Lines = 12, Remove Tri-State Buffer = Checked, and I/O Direction = F0. This example shows when bit 0-3 are inputs only, bit 4-7 are outputs only, and bit 8-11 are bidirectional.

```
module  gpio_top (
//port declaration
clk_i,
resetn_i,
apb_penable_i,
apb_psel_i,
apb_pwrite_i,
apb_paddr_i,
apb_pwdata_i,
apb_prdata_o,
apb_pslverr_o,
apb_pready_o,
int_o,
gpio_0to3_i,
gpio_4to7_o,
gpio_8to11_io
);
    input clk_i ;
    input resetn_i ;
    input apb_penable_i ;
    input apb_psel_i ;
    input apb_pwrite_i ;
    input [5:0] apb_paddr_i ;
    input [31:0] apb_pwdata_i ;
    output [31:0] apb_prdata_o ;
    output apb_pslverr_o ;
    output apb_pready_o ;
    output int_o ;
//GPIO signals
    input [3:0] gpio_0to3_i;
    output [3:0] gpio_4to7_o;
    inout [3:0] gpio_8to11_io;

    wire [11:0] gpio_i_w;
    wire [11:0] gpio_o_w;
    wire [11:0] gpio_en_o_w;

assign gpio_i_w [3:0] = gpio_0to3_i [3:0];
assign gpio_4to7_o [3:0] = gpio_o_w[3:0];

//GPIO instance
gpio1 gpio_test (.gpio_i (gpio_i_w),
          .gpio_o (gpio_o_w),
          .gpio_en_o (gpio_en_o_w),
          .clk_i (clk_i),
          .resetn_i (resetn_i),
```

```
          .apb_penable_i (apb_penable_i),
          .apb_psel_i (apb_psel_i),
          .apb_pwrite_i (apb_pwrite_i),
          .apb_paddr_i (apb_paddr_i),
          .apb_pwdata_i (apb_pwdata_i),
          .apb_prdata_o (apb_prdata_o),
          .apb_pslverr_o (apb_pslverr_o),
          .apb_pready_o (apb_pready_o),
          .int_o (int_o)
);

//Tri-State Buffer for Bidirectional GPIO Signal
genvar i;
for(i = 0; i < 4; i=i+1) begin
BB u_BB_data(
        .B                    (gpio_8to11_io[i]),
        .I                     (gpio_o_w[i+8]),
        .T                    (gpio_en_o_w[i+8]),
        .O                   (gpio_i_w[i+8])
      );
end
endmodule
```

3. IP Generation and Evaluation

3.1. Licensing the IP

This IP is distributed without license.

3.2. IP Generation using Lattice Radiant Software

The Lattice Radiant software allows you to generate and customize modules and IPs and integrate them into the device architecture.

To generate the GPIO soft IP in Lattice Radiant Software:

1. In **IP Catalog** tab, select **GPIO** and double-click to open the **Module/IP Block Wizard**.
2. In the **Module/IP Block Wizard**, create a new instance of the IP, such as gpio_1.
3. Configure the GPIO module according to custom specifications using drop-down menus and check boxes. A sample configuration is shown in Figure 3.1. For configuration options, see Table 2.2.

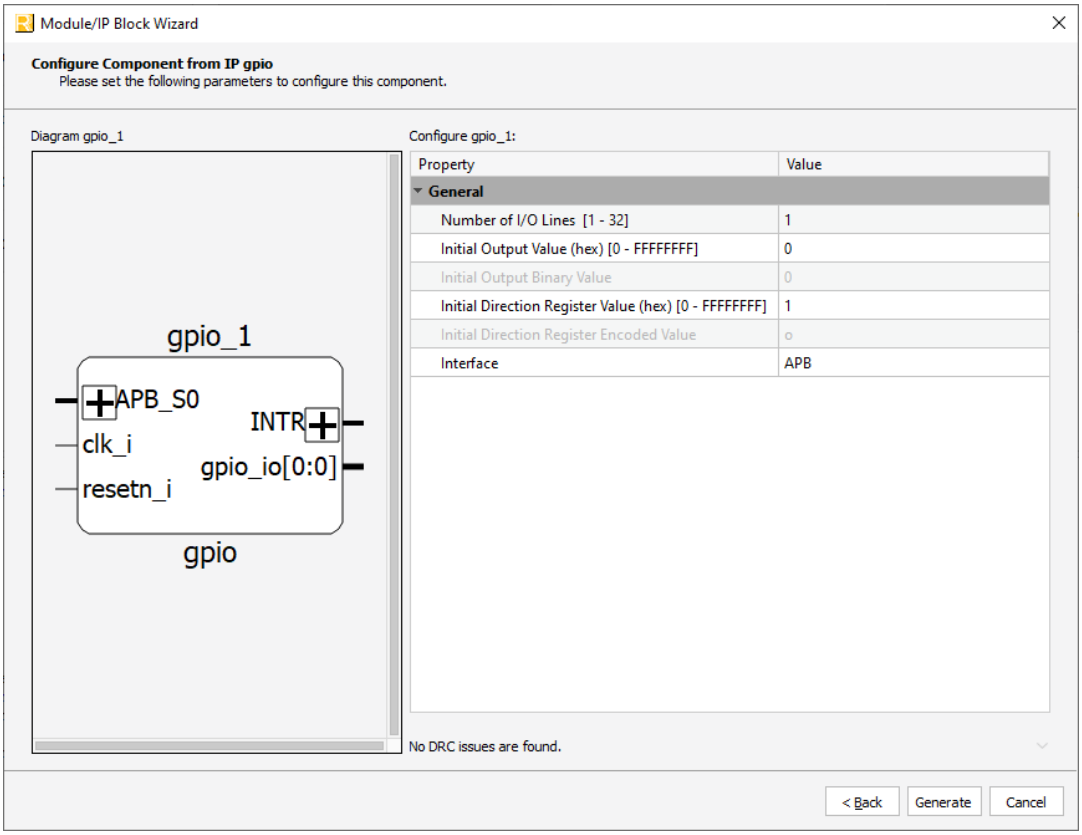

Figure 3.1. Configure Block of GPIO Module

4. Click **Generate**. The **Check Generating Result** dialog box opens, showing design block messages and results as shown in Figure 3.2.

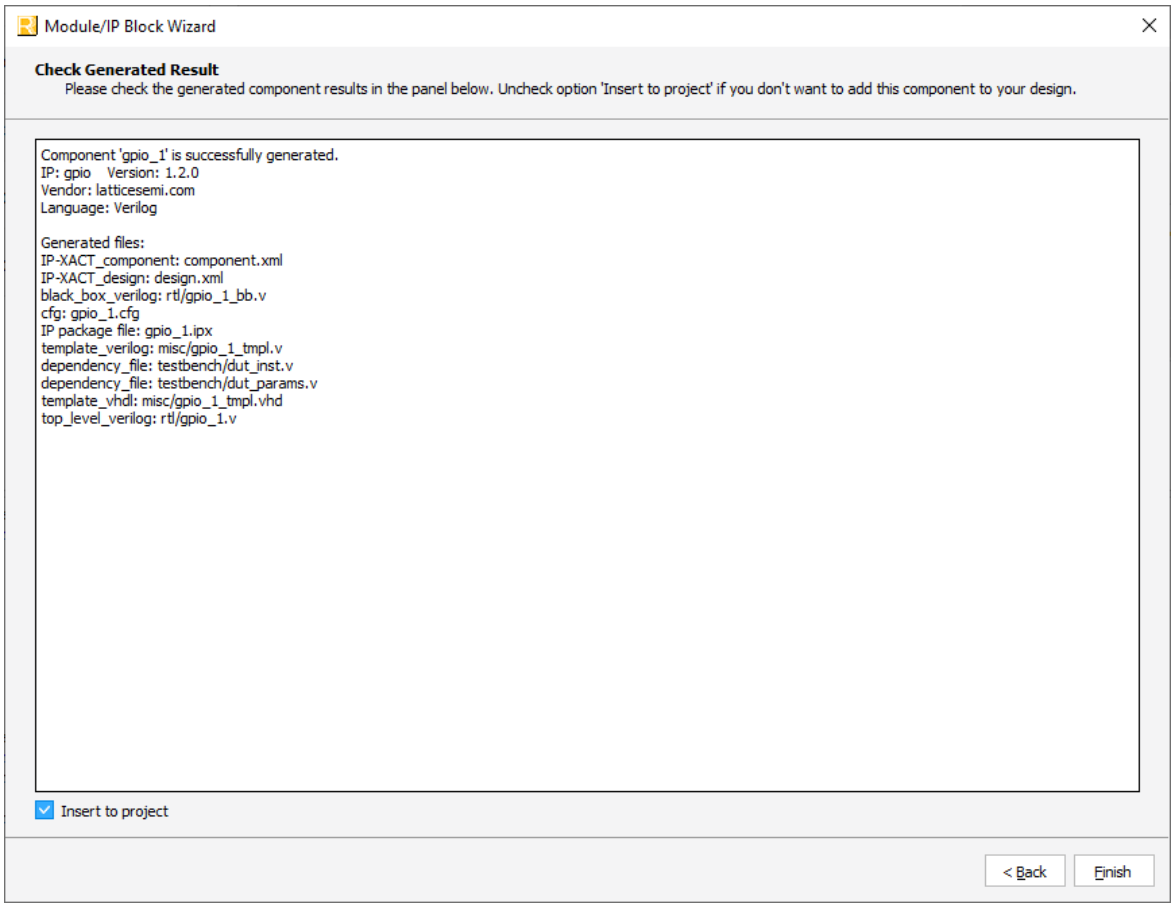

Figure 3.2. Check Generating Result

5. Click the **Finish** button to generate the Verilog file.

3.3. IP Generation using Lattice Propel Builder

Lattice Propel Builder allows you to generate and customize modules and IPs and integrate them into the device architecture.

To generate the GPIO soft IP in Lattice Propel Builder:

1. Create a new design by clicking **File > New Design**.
2. In **IP Catalog** tab, select **GPIO** and double-click to open the **Module/IP Block Wizard**.
3. In the **Module/IP Block Wizard**, create a new instance of the IP, such as gpio_1.
4. Configure the GPIO module according to custom specifications, using drop-down menus and check boxes. A sample configuration is shown in Figure 3.1. For configuration options, see Table 2.2.
5. Click **Generate**. The **Check Generating Result** dialog box opens, showing design block messages and results as shown in Figure 3.2.
6. Click the **Finish** button to generate the Verilog file.

3.4. Functional Simulation

This section provides information on how run simulation when the IP is generated using the Lattice Radiant software. For IP that is generated using Lattice Propel Builder, refer to the System Simulation Flow section of the Lattice Propel 1.0 User Guide.

To run Verilog simulation:

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** as shown in Figure 3.3.

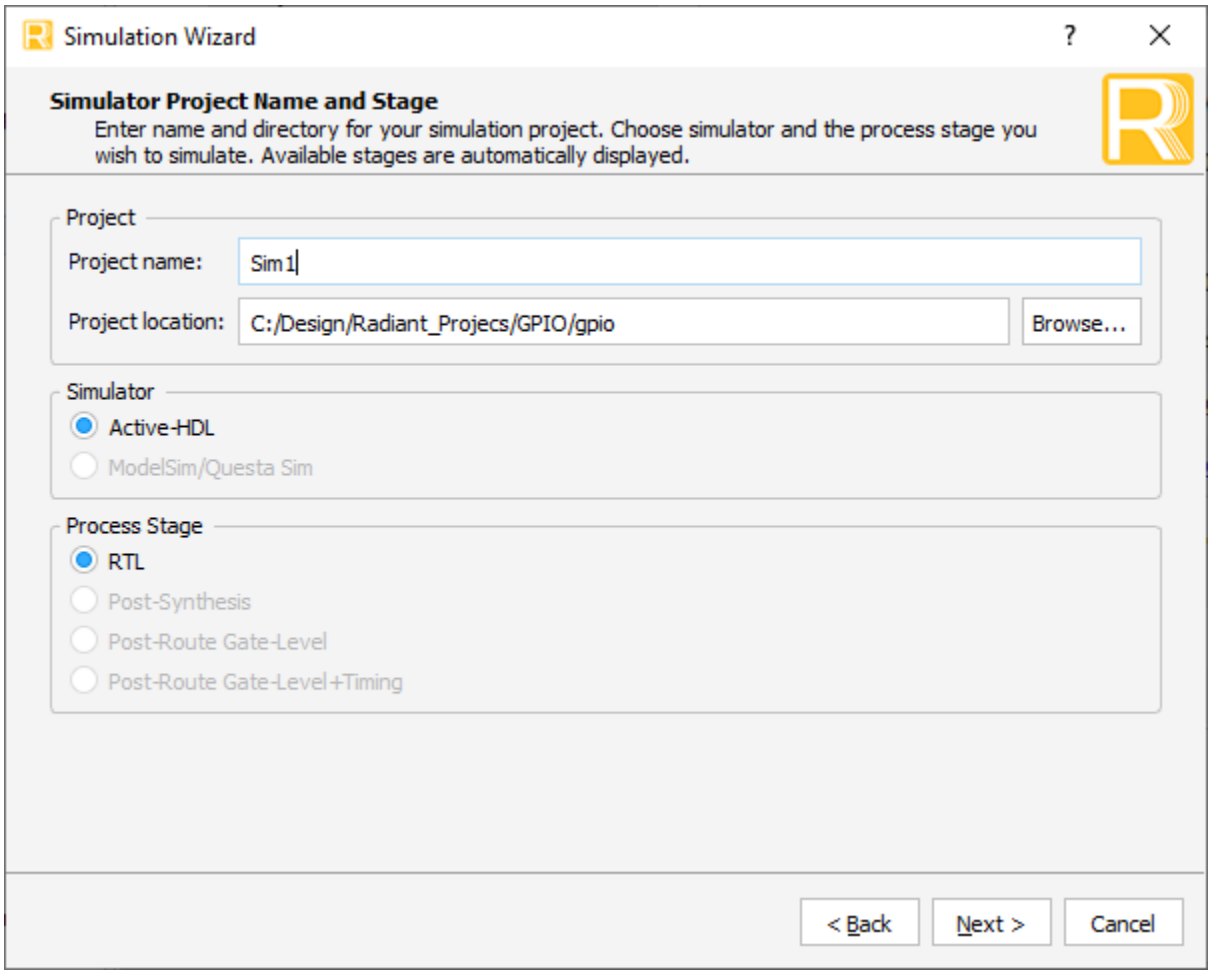

Figure 3.3. Simulation Wizard

2. Double-click **Next** to open the **Add and Reorder Source** window as shown in Figure 3.4.

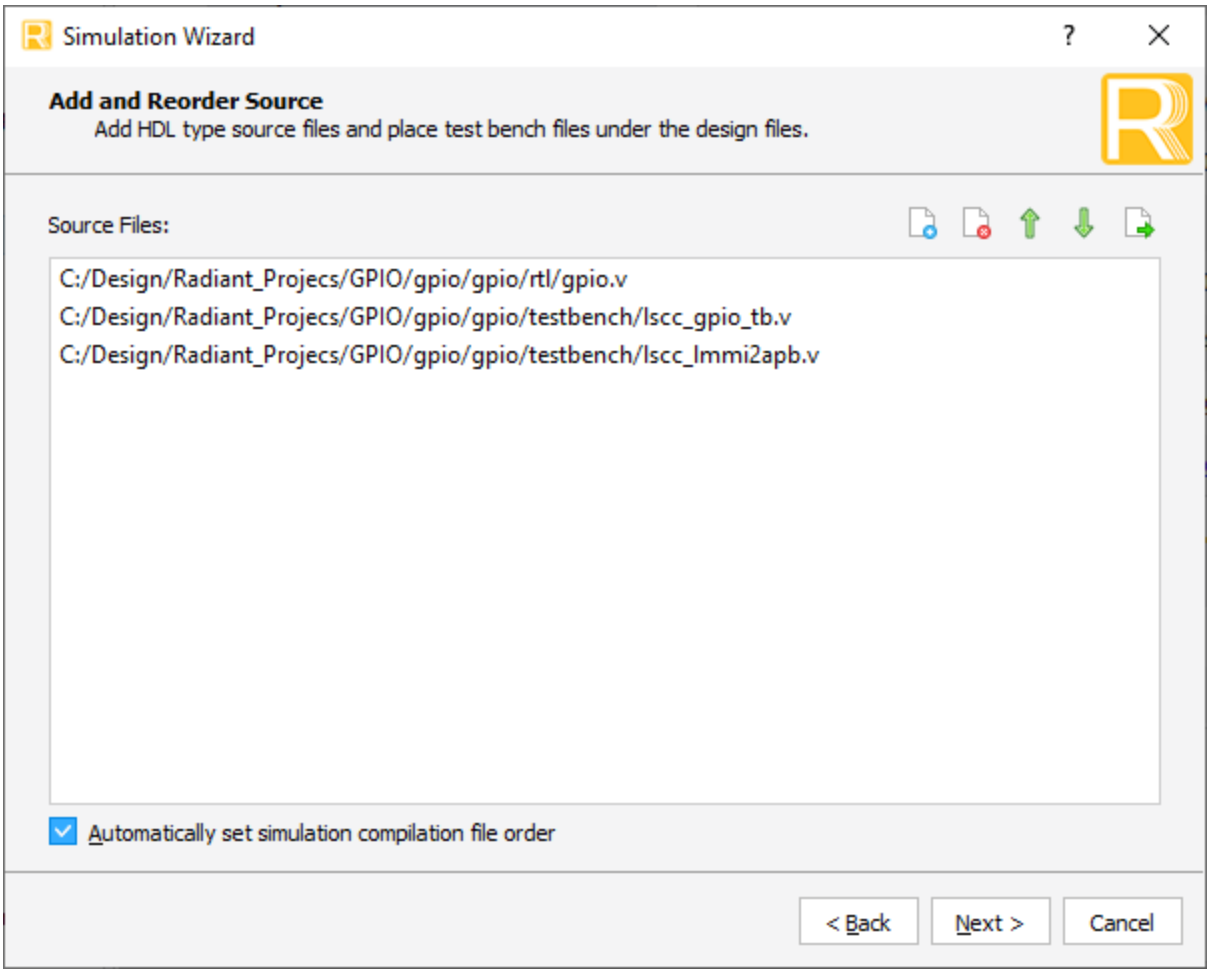

Figure 3.4. Adding and Reordering Source

3. Click **Next** and **Finish** to run the simulation.

References

- iCE40 UltraPlus Web Page at www.latticesemi.com
- MachXO2 Web Page at www.latticesemi.com
- MachXO3 Web Page at www.latticesemi.com
- MachXO3D Web Page at www.latticesemi.com
- Mach-NX Web Page at www.latticesemi.com
- CrossLink-NX Web Page at www.latticesemi.com
- Certus-NX Web Page at www.latticesemi.com

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Revision 2.0, April 2021

Section	Change Summary
Introduction	Updated Table 1.1 to add support for MachXO2 and MachXO3.
References	Added references to MachXO2 and MachXO3.

Revision 1.9, December 2020

Section	Change Summary
Introduction	• Added Mach-NX to Table 1.1.
Functional Description	• Added GPIO Interface ports to Table 2.1. • Added and updated attributes in Table 2.2. • Added I/O Port Usage section.

Revision 1.8, June 2020

Section	Change Summary
Introduction	Updated Table 1.1 to add support for Certus-NX.
Functional Description	Updated Table 2.2 to add LIFCL-17 and LFD2NX-40.
References	Updated link to Lattice Radiant Software 2.1 User Guide.

Revision 1.7, May 2020

Section	Change Summary
All	Changed document title to GPIO IP Core; removed Lattice Radiant Software.
Introduction	Replaced the Quick Facts table with Table 1.1. FPGA Software for IP Configuration, Generation and Implementation.

Functional Description	• Updated Signal Descriptions section.
	  • Modified port names in Table 2.1.
	• Updated Attribute Summary section.
	  • Changed the default value of Interface from LMMI to APB.
	  • Modified attributes in Table 2.2.

IP Generation and Evaluation	• Updated licensing information • Added IP Generation using Lattice Propel Builder
References	Added reference to the Lattice Propel Builder user guide.
—	Minor style/formatting changes.

Revision 1.6, February 2020

Section	Change Summary
Introduction	Updated Table 1.1 to add LIFCL-17 as targeted device.

Revision 1.5, December 2019

Section	Change Summary
Introduction	Updated Table 1.1. Quick Facts.
Functional Description	• Updated Interface description in Table 2.2. Attributes Table. • Added Programming Flow section.
IP Generation and Evaluation	Updated licensing information.

Revision 1.4, October 2019

Section	Change Summary
—	Appended Lattice Radiant Software to document title.
Disclaimers	Added this section.
Acronyms in This Document	Added this section.
Attribute Summary	Added information in Table 2.2. Attributes Table.

Revision 1.3, September 2019

Section	Change Summary
All	Updated signal names.

Revision 1.2, September 2019

Section	Change Summary
All	Added APB Block description to text. Updated figures and tables to reflect addition of APB.

Revision 1.1, April 2019

Section	Change Summary
All	• Redesigned document structure • Renamed TRISTATE_REG to DIRECTION_REG in the Registers Map Table and in the corresponding descriptions everywhere. Swapped input and output values (0 to 1, 1 to 0) • Changed Address Space Offset heading in Table 2 to Address Offset • Added Soft IP Revision History

Revision 1.0, November 2018

Section	Change Summary
All	Initial release

www.latticesemi.com