



# GPIO Intel® FPGA IP User Guide

---

## Intel® Arria® 10 and Intel® Cyclone® 10 GX Devices

Updated for Intel® Quartus® Prime Design Suite: **21.1**

IP Version: **20.0.0**



[Subscribe](#)

[Send Feedback](#)

[ug-altera\\_gpio](#) | 2021.03.29

Latest document on the web: [PDF](#) | [HTML](#)

## Contents

---

<b>GPIO Intel® FPGA IP User Guide: Intel® Arria® 10 and Intel® Cyclone® 10 GX Devices.....</b>	<b>3</b>
Release Information for GPIO Intel FPGA IP.....	3
GPIO Intel FPGA IP Features.....	4
GPIO Intel FPGA IP Data Paths.....	4
Input Path.....	5
Output and Output Enable Paths.....	7
GPIO Intel FPGA IP Interface Signals.....	8
Shared Signals.....	11
Data Bit-Order for Data Interface.....	11
Input and Output Bus High and Low Bits.....	12
Data Interface Signals and Corresponding Clocks.....	12
Verifying Resource Utilization and Design Performance.....	13
GPIO Intel FPGA IP Parameter Settings.....	13
Register Packing.....	15
GPIO Intel FPGA IP Timing.....	16
Timing Components.....	16
Delay Elements.....	17
Timing Analysis.....	18
Timing Closure Guidelines.....	21
GPIO Intel FPGA IP Design Examples.....	21
GPIO IP Core Synthesizable Intel Quartus Prime Design Example.....	22
GPIO IP Core Simulation Design Example.....	22
IP Migration Flow for Arria V, Cyclone V, and Stratix V Devices.....	22
Migrating Your ALTDDIO_IN, ALTDDIO_OUT, ALTDDIO_BIDIR, and ALTIOBUF IP Cores.....	23
Guideline: Swap datain_h and datain_l Ports in Migrated IP.....	23
GPIO Intel FPGA IP User Guide Archives.....	24
Document Revision History for GPIO Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices.....	24

## GPIO Intel® FPGA IP User Guide: Intel® Arria® 10 and Intel® Cyclone® 10 GX Devices

---

The GPIO Intel® FPGA IP core supports the general purpose I/O (GPIO) features and components. You can use GPIOs in general applications that are not specific to transceivers, memory interfaces, or LVDS.

The GPIO IP core is available for Intel Arria® 10 and Intel Cyclone® 10 GX devices only. If you are migrating designs from Stratix® V, Arria V, or Cyclone V devices, you must migrate the ALTDDIO\_IN, ALTDDIO\_OUT, ALTDDIO\_BIDIR, or ALTIobuf IP cores.

### Related Information

- [IP Migration Flow for Arria V, Cyclone V, and Stratix V Devices](#) on page 22
- [Intel Stratix 10 I/O Implementation Guides](#)  
Provides the GPIOIP core user guide for Intel Stratix 10 devices.
- [Introduction to Intel FPGA IP Cores](#)  
Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- [Creating Version-Independent IP and Qsys Simulation Scripts](#)  
Create simulation scripts that do not require manual updates for software or IP version upgrades.
- [Project Management Best Practices](#)  
Guidelines for efficient management and portability of your project and IP files.
- [GPIO Intel FPGA IP User Guide Archives](#) on page 24  
Provides a list of user guides for previous versions of the GPIO IP core.
- [Double Data Rate I/O \(ALTDDIO\\_IN, ALTDDIO\\_OUT, and ALTDDIO\\_BIDIR\) IP Cores User Guide](#)
- [I/O Buffer \(ALTIobuf\) IP Core User Guide](#)

### Release Information for GPIO Intel FPGA IP

Intel FPGA IP versions match the Intel Quartus® Prime Design Suite software versions until v19.1. Starting in Intel Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

**Table 1. GPIO Intel FPGA IP Core Current Release Information**

Item	Description
IP Version	20.0
Intel Quartus Prime Version	21.1
Release Date	2021.03.29

## GPIO Intel FPGA IP Features

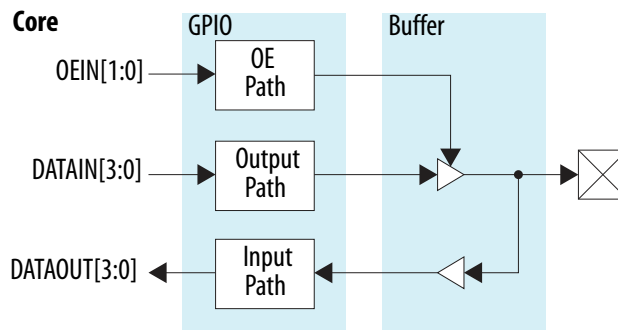
The GPIO IP core includes features to support the device I/O blocks. You can use the Intel Quartus Prime parameter editor to configure the GPIO IP core.

The GPIO IP core provides these components:

- Double data rate input/output (DDIO)—a digital component that doubles or halves the data rate of a communication channel.
- Delay chains—configure the delay chains to perform specific delay and assist in I/O timing closure.
- I/O buffers—connect the pads to the FPGA.

## GPIO Intel FPGA IP Data Paths

**Figure 1. High-Level View of Single-Ended GPIO**



**Table 2. GPIO IP Core Data Path Modes**

Data Path	Register Mode			
	Bypass	Simple Register	DDR I/O	
			Full-Rate	Half-Rate
Input	Data goes from the delay element to the core, bypassing all double data rate I/Os (DDIOs).	The full-rate DDIO operates as a simple register, bypassing half-rate DDIOs. The Fitter chooses whether to pack the register in the I/O or implement the register in the core, depending on the area and timing trade-offs.	The full-rate DDIO operates as a regular DDIO, bypassing the half-rate DDIOs.	The full-rate DDIO operates as a regular DDIO. The half-rate DDIOs convert full-rate data to half-rate data.
Output	Data goes from the core straight to the delay element, bypassing all DDIOs.	The full-rate DDIO operates as a simple register, bypassing half-rate DDIOs. The Fitter chooses whether to pack the register in the I/O or implement the register in the core, depending on the area and timing trade-offs.	The full-rate DDIO operates as a regular DDIO, bypassing the half-rate DDIOs.	The full-rate DDIO operates as a regular DDIO. The half-rate DDIOs convert full-rate data to half-rate data.
Bidirectional	The output buffer drives both an output pin and an input buffer.	The full-rate DDIO operates as a simple register. The output buffer drives both an output pin and an input buffer.	The full-rate DDIO operates as a regular DDIO. The output buffer drives both an output pin and an input buffer. The input buffer drives a set of three flip-flops.	The full-rate DDIO operates as a regular DDIO. The half-rate DDIOs convert full-rate data to half-rate. The output buffer drives both an output pin and an input buffer. The input buffer drives a set of three flip-flops.

If you use asynchronous clear and preset signals, all DDIOs share these same signals.

Half-rate and full-rate DDIOs connect to separate clocks. When you use half-rate and full-rate DDIOs, the full-rate clock must run at twice the half-rate frequency. You can use different phase relationships to meet timing requirements.

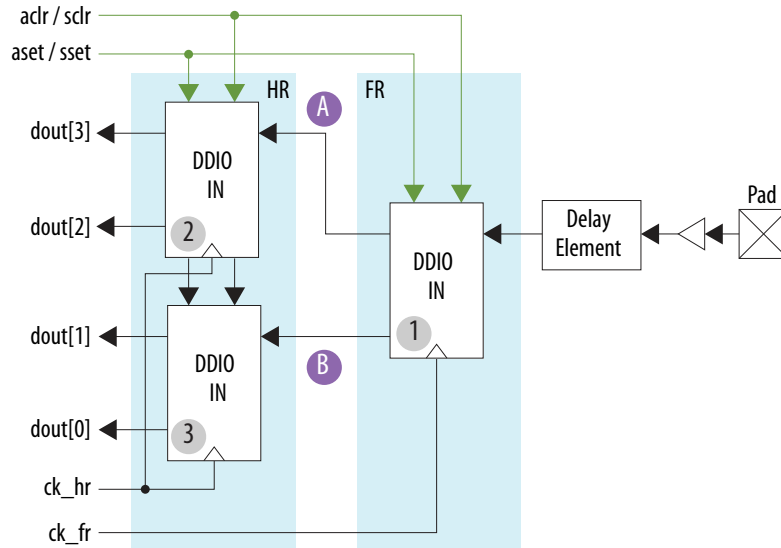
**Related Information**

[Input and Output Bus High and Low Bits](#) on page 12

**Input Path**

The pad sends data to the input buffer, and the input buffer feeds the delay element. After the data goes to the output of the delay element, the programmable bypass multiplexers select the features and paths to use. Each input path contains two stages of DDIOs, which are full-rate and half-rate.

**Figure 2. Simplified View of Single-Ended GPIO Input Path**

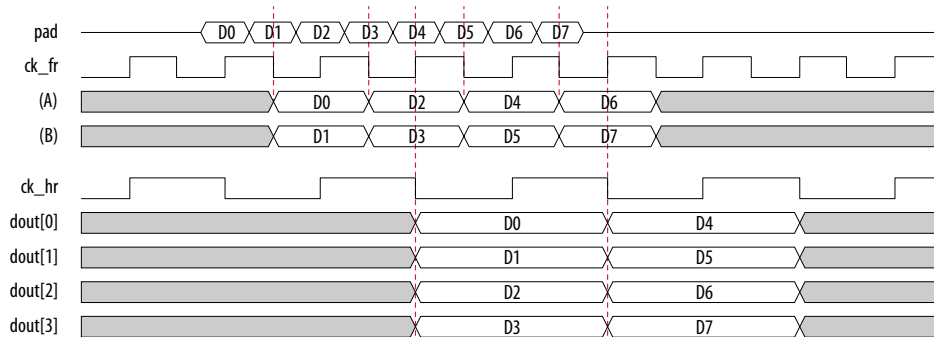


1. The pad receives data.
2. DDIO IN (1) captures data on the rising and falling edges of `ck_fr` and sends the data, signals (A) and (B) in the following waveform figure, at single data rate.
3. DDIO IN (2) and DDIO IN (3) halve the data rate.
4. `dout[3:0]` presents the data as a half-rate bus.

**Figure 3. Input Path Waveform in DDIO Mode with Half-Rate Conversion**

In this figure, the data goes from full-rate clock at double data rate to half-rate clock at single data rate. The data rate is divided by four and the bus size is increased by the same ratio. The overall throughput through the GPIO IP core remains unchanged.

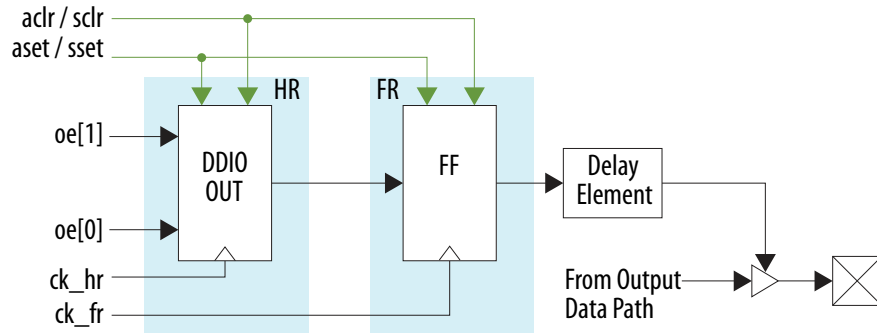
The actual timing relationship between different signals may vary depending on the specific design, delays, and phases that you choose for the full-rate and half-rate clocks.



**Note:** The GPIO IP core does not support dynamic calibration of bidirectional pins. For applications that require dynamic calibration of bidirectional pins, refer to the related information.



**Figure 6. Simplified View of Output Enable Path**



The difference between the output path and output enable (OE) path is that the OE path does not contain full-rate DDIO. To support packed-register implementations in the OE path, a simple register operates as full-rate DDIO. For the same reason, only one half-rate DDIO is present.

The OE path operates in the following three fundamental modes:

- Bypass—the core sends data directly to the delay element, bypassing all DDIOs.
- Packed Register—bypasses half-rate DDIO.
- SDR output at half-rate—half-rate DDIOs convert data from full-rate to half-rate.

*Note:*

The GPIO IP core does not support dynamic calibration of bidirectional pins. For applications that require dynamic calibration of bidirectional pins, refer to the related information.

### Related Information

- [PHY Lite for Parallel Interfaces Intel FPGA IP Core User Guide: Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX Devices](#)  
Provides more information for applications that require dynamic OCT for bidirectional pins.
- [Input Path](#) on page 5

## GPIO Intel FPGA IP Interface Signals

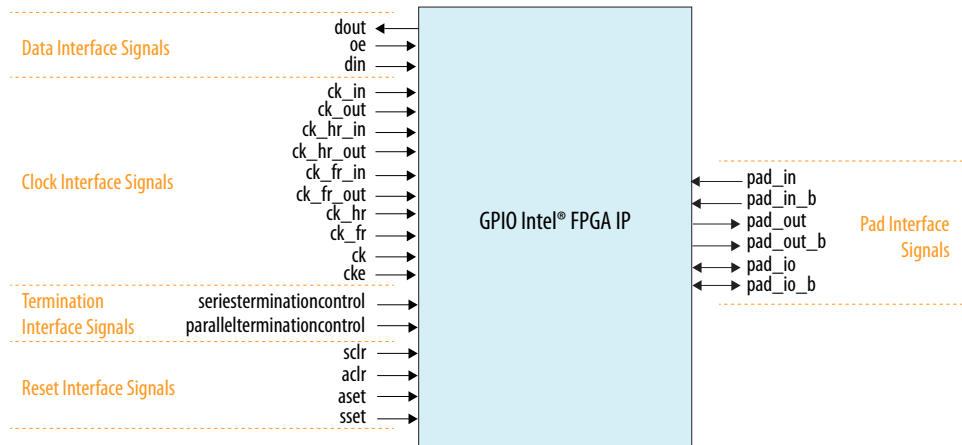
Depending on parameter settings you specify, different interface signals are available for the GPIO IP core.

**Figure 7. GPIO IP Core Interfaces**





**Figure 8. GPIO Interface Signals**



**Table 3. Pad Interface Signals**

The pad interface is the physical connection from the GPIO IP core to the pad. This interface can be an input, output or bidirectional interface, depending on the IP core configuration. In this table, *SIZE* is the data width specified in the IP core parameter editor.

Signal Name	Direction	Description
pad_in[SIZE-1:0]	Input	Input signal from the pad.
pad_in_b[SIZE-1:0]	Input	Negative node of the differential input signal from the pad. This port is available if you turn on the <b>Use differential buffer</b> option.
pad_out[SIZE-1:0]	Output	Output signal to the pad.
pad_out_b[SIZE-1:0]	Output	Negative node of the differential output signal to the pad. This port is available if you turn on the <b>Use differential buffer</b> option.
pad_io[SIZE-1:0]	Bidirectional	Bidirectional signal connection with the pad.
pad_io_b[SIZE-1:0]	Bidirectional	Negative node of the differential bidirectional signal connection with the pad. This port is available if you turn on the <b>Use differential buffer</b> option.

**Table 4. Data Interface Signals**

The data interface is an input or output interface from the GPIO IP core to the FPGA core. In this table, *SIZE* is the data width specified in the IP core parameter editor.

Signal Name	Direction	Description
din[ <i>DATA_SIZE</i> -1:0]	Input	Data input from the FPGA core in output or bidirectional mode. <i>DATA_SIZE</i> depends on the register mode: <ul style="list-style-type: none"> <li>Bypass or simple register—<i>DATA_SIZE</i> = <i>SIZE</i></li> <li>DDIO without half-rate logic—<i>DATA_SIZE</i> = 2 × <i>SIZE</i></li> <li>DDIO with half-rate logic—<i>DATA_SIZE</i> = 4 × <i>SIZE</i></li> </ul>
dout[ <i>DATA_SIZE</i> -1:0]	Output	Data output to the FPGA core in input or bidirectional mode, <i>DATA_SIZE</i> depends on the register mode: <ul style="list-style-type: none"> <li>Bypass or simple register—<i>DATA_SIZE</i> = <i>SIZE</i></li> <li>DDIO without half-rate logic—<i>DATA_SIZE</i> = 2 × <i>SIZE</i></li> <li>DDIO with half-rate logic—<i>DATA_SIZE</i> = 4 × <i>SIZE</i></li> </ul>
oe[ <i>OE_SIZE</i> -1:0]	Input	OE input from the FPGA core in output mode with <b>Enable output enable port</b> turned on, or bidirectional mode. OE is active high. When transmitting data, set this signal to 1. When receiving data, set this signal to 0. <i>OE_SIZE</i> depends on the register mode: <ul style="list-style-type: none"> <li>Bypass or simple register—<i>DATA_SIZE</i> = <i>SIZE</i></li> <li>DDIO without half-rate logic—<i>DATA_SIZE</i> = <i>SIZE</i></li> <li>DDIO with half-rate logic—<i>DATA_SIZE</i> = 2 × <i>SIZE</i></li> </ul>

**Table 5. Clock Interface Signals**

The clock interface is an input clock interface. It consists of different signals, depending on the configuration. The GPIO IP core can have zero, one, two, or four clock inputs. Clock ports appear differently in different configurations to reflect the actual function performed by the clock signal.

Signal Name	Direction	Description
ck	Input	In input and output paths, this clock feeds a packed register or DDIO if you turn off the <b>Half Rate logic</b> parameter. In bidirectional mode, this clock is the unique clock for the input and output paths if you turn off the <b>Separate input/output Clocks</b> parameter.
ck_fr	Input	In input and output paths, these clocks feed the full-rate and half-rate DDIOs if you turn on the <b>Half Rate logic</b> parameter. In bidirectional mode, the input and output paths use these clocks if you turn off the <b>Separate input/output Clocks</b> parameter.
ck_hr		
ck_in	Input	In bidirectional mode, these clocks feed a packed register or DDIO in the input and output paths if you specify both these settings: <ul style="list-style-type: none"> <li>Turn off the <b>Half Rate logic</b> parameter.</li> <li>Turn on the <b>Separate input/output Clocks</b> parameter.</li> </ul>
ck_out		
ck_fr_in	Input	In bidirectional mode, these clocks feed a full-rate and half-rate DDIOs in the input and output paths if you specify both these settings <ul style="list-style-type: none"> <li>Turn on the <b>Half Rate logic</b> parameter.</li> <li>Turn on the <b>Separate input/output Clocks</b> parameter.</li> </ul> For example, <i>ck_fr_out</i> feeds the full-rate DDIO in the output path.
ck_fr_out		
ck_hr_in		
ck_hr_out		
cke	Input	Clock enable.

**Table 6. Termination Interface Signals**

The termination interface connects the GPIO IP core to the I/O buffers.

Signal Name	Direction	Description
seriesterminationcontrol	Input	Input from the termination control block (OCT) to the buffers. It sets the buffer series impedance value.
parallelerminationcontrol	Input	Input from the termination control block (OCT) to the buffers. It sets the buffer parallel impedance value.

**Table 7. Reset Interface Signals**

The reset interface connects the GPIO IP core to the DDIOs.

Signal Name	Direction	Description
sclr	Input	Synchronous clear input. Not available if you enable sset.
aclr	Input	Asynchronous clear input. Active high. Not available if you enable aset.
aset	Input	Asynchronous set input. Active high. Not available if you enable aclr.
sset	Input	Synchronous set input. Not available if you enable sclr.

**Related Information**

[Input and Output Bus High and Low Bits](#) on page 12

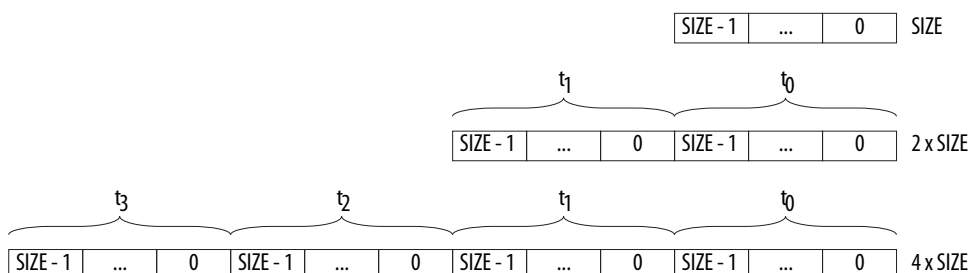
**Shared Signals**

- The input, output, and OE paths share the same clear and preset signals.
- The output and OE path shares the same clock signals.

**Data Bit-Order for Data Interface**

**Figure 9. Data Bit-Order Convention**

This figure shows the bit-order convention for the din, dout and oe data signals.



- If the data bus size value is  $SIZE$ , the LSB is at the right-most position.
- If the data bus size value is  $2 \times SIZE$ , the bus is made of two words of  $SIZE$ .
- If the data bus size value  $4 \times SIZE$ , the bus is made of four words of  $SIZE$ .
- The LSB is in the right-most position of each word.
- The right-most word specifies the first word going out for output buses and the first word coming in for input buses.

**Related Information**

[Input Path](#) on page 5

**Input and Output Bus High and Low Bits**

The high and low bits in the input or output signals are included in the `din` and `dout` input and output buses.

**Input Bus**

For the `din` bus, if `datain_h` and `datain_l` are the high and low bits, with each width being `datain_width`:

- `datain_h = din[(2 × datain_width - 1):datain_width]`
- `datain_l = din[(datain_width - 1):0]`

For example, for `din[7:0] = 8'b11001010`:

- `datain_h = 4'b1100`
- `datain_l = 4'b1010`

**Output Bus**

For the `dout` bus, if `dataout_h` and `dataout_l` are the high and low bits, with each width being `dataout_width`:

- `dataout_h = dout[(2 × dataout_width - 1):dataout_width]`
- `dataout_l = dout[(dataout_width - 1):0]`

For example, for `dout[7:0] = 8'b11001010`:

- `dataout_h = 4'b1100`
- `dataout_l = 4'b1010`

**Data Interface Signals and Corresponding Clocks**

**Table 8. Data Interface Signals and Corresponding Clocks**

Signal Name	Parameter Configuration			Clock
	Register Mode	Half Rate	Separate Clocks	
din	<ul style="list-style-type: none"> <li>• Simple Register</li> <li>• DDIO</li> </ul>	Off	Off	ck
	DDIO	On	Off	ck_hr
	<ul style="list-style-type: none"> <li>• Simple Register</li> <li>• DDIO</li> </ul>	Off	On	ck_in
	DDIO	On	On	ck_hr_in
<ul style="list-style-type: none"> <li>• dout</li> <li>• oe</li> </ul>	<ul style="list-style-type: none"> <li>• Simple Register</li> <li>• DDIO</li> </ul>	Off	Off	ck

*continued...*

Signal Name	Parameter Configuration			Clock
	Register Mode	Half Rate	Separate Clocks	
	DDIO	On	Off	ck_hr
	<ul style="list-style-type: none"> <li>Simple Register</li> <li>DDIO</li> </ul>	Off	On	ck_out
	DDIO	On	On	ck_hr_out
<ul style="list-style-type: none"> <li>sclr</li> <li>sset</li> <li>All pad signals</li> </ul>	<ul style="list-style-type: none"> <li>Simple Register</li> <li>DDIO</li> </ul>	Off	Off	ck
	DDIO	On	Off	ck_fr
	<ul style="list-style-type: none"> <li>Simple Register</li> <li>DDIO</li> </ul>	Off	On	<ul style="list-style-type: none"> <li>Input path: ck_in</li> <li>Output path: ck_out</li> </ul>
	DDIO	On	On	<ul style="list-style-type: none"> <li>Input path: ck_fr_in</li> <li>Output path: ck_fr_out</li> </ul>

## Verifying Resource Utilization and Design Performance

You can refer to the Intel Quartus Prime compilation reports to get details about the resource usage and performance of your design.

1. On the menu, click **Processing** > **Start Compilation** to run a full compilation.
2. After compiling the design, click **Processing** > **Compilation Report**.
3. Using the **Table of Contents**, navigate to **Fitter** > **Resource Section**.
  - a. To view the resource usage information, select **Resource Usage Summary**.
  - b. To view the resource utilization information, select **Resource Utilization by Entity**.

## GPIO Intel FPGA IP Parameter Settings

You can set the parameter settings for the GPIO IP core in the Intel Quartus Prime software. There are three groups of options: **General**, **Buffer**, and **Registers**.

**Table 9. GPIO IP Core Parameters - General**

Parameter	Condition	Allowed Values	Description
Data Direction	—	<ul style="list-style-type: none"> <li>Input</li> <li>Output</li> <li>Bidir</li> </ul>	Specifies the data direction for the GPIO.
Data width	—	1 to 128	Specifies the data width.
Use legacy top-level port names	—	<ul style="list-style-type: none"> <li>On</li> <li>Off</li> </ul>	Use same port names as in Stratix V, Arria V, and Cyclone V devices. For example, dout becomes dataout_h and dataout_l, and din becomes datain_h and datain_l.
<i>continued...</i>			

Parameter	Condition	Allowed Values	Description
			Note: The behavior of these ports are different than in the Stratix V, Arria V, and Cyclone V devices. For the migration guideline, refer to the related information.

**Table 10. GPIO IP Core Parameters - Buffer**

Parameter	Condition	Allowed Values	Description
Use differential buffer	—	<ul style="list-style-type: none"> <li>On</li> <li>Off</li> </ul>	If turned on, enables differential I/O buffers.
Use pseudo differential buffer	<ul style="list-style-type: none"> <li>Data Direction = Output</li> <li>Use differential buffer = On</li> </ul>	<ul style="list-style-type: none"> <li>On</li> <li>Off</li> </ul>	If turned on in output mode, enables pseudo differential output buffers. This option is automatically turned on for bidirectional mode if you turn on <b>Use differential buffer</b> .
Use bus-hold circuitry	<ul style="list-style-type: none"> <li>Data Direction = Input or Bidir</li> <li>Use differential buffer = Off</li> </ul>	<ul style="list-style-type: none"> <li>On</li> <li>Off</li> </ul>	If turned on, the bus hold circuitry can weakly hold the signal on an I/O pin at its last-driven state where the output buffer state will be 1 or 0 but not high-impedance.
Use open drain output	<ul style="list-style-type: none"> <li>Data Direction = Output or Bidir</li> <li>Use differential buffer = Off</li> </ul>	<ul style="list-style-type: none"> <li>On</li> <li>Off</li> </ul>	If turned on, the open drain output enables the device to provide system-level control signals such as interrupt and write enable signals that can be asserted by multiple devices in your system.
Enable output enable port	Data Direction = Output	<ul style="list-style-type: none"> <li>On</li> <li>Off</li> </ul>	If turned on, enables user input to the OE port. This option is automatically turned on for bidirectional mode.
Enable seriestermination / paralleltermination ports	—	<ul style="list-style-type: none"> <li>On</li> <li>Off</li> </ul>	If turned on, enables the <code>seriesterminationcontrol</code> and <code>parallelterminationcontrol</code> ports of the output buffer.

**Table 11. GPIO IP Core Parameters - Registers**

Parameter	Condition	Allowed Values	Description
Register mode	—	<ul style="list-style-type: none"> <li>None</li> <li>Simple register</li> <li>DDIO</li> </ul>	Specifies the register mode for the GPIO IP core: <ul style="list-style-type: none"> <li><b>None</b>—specifies a simple wire connection from/to the buffer.</li> <li><b>Simple register</b>—specifies that the DDIO is used as a simple register in single data-rate mode (SDR). The Fitter may pack this register in the I/O.</li> <li><b>DDIO</b>— specifies that the IP core uses the DDIO.</li> </ul>
Enable synchronous clear / preset port	<ul style="list-style-type: none"> <li>Register mode = DDIO</li> </ul>	<ul style="list-style-type: none"> <li>None</li> <li>Clear</li> <li>Preset</li> </ul>	Specifies how to implement synchronous reset port. <ul style="list-style-type: none"> <li><b>None</b>—Disables synchronous reset port.</li> <li><b>Clear</b>—Enables the <code>SCLR</code> port for synchronous clears.</li> <li><b>Preset</b>—Enables the <code>SSET</code> port for synchronous preset.</li> </ul>
<i>continued...</i>			

Parameter	Condition	Allowed Values	Description
Enable asynchronous clear / preset port	<ul style="list-style-type: none"> <li>Register mode = DDIO</li> </ul>	<ul style="list-style-type: none"> <li>None</li> <li>Clear</li> <li>Preset</li> </ul>	Specifies how to implement asynchronous reset port. <ul style="list-style-type: none"> <li><b>None</b>—Disables asynchronous reset port.</li> <li><b>Clear</b>—Enables the ACLR port for asynchronous clears.</li> <li><b>Preset</b>—Enables the ASET port for asynchronous preset.</li> </ul> ACLR and ASET signals are active high.
Enable clock enable ports	Register mode = DDIO	<ul style="list-style-type: none"> <li>On</li> <li>Off</li> </ul>	<ul style="list-style-type: none"> <li><b>On</b>—exposes the clock enable (CKE) port to allow you to control when data is clocked in or out. This signal prevents data from being passed through without your control.</li> <li><b>Off</b>—clock enable port is not exposed and data always pass through the register automatically.</li> </ul>
Half Rate logic	Register mode = DDIO	<ul style="list-style-type: none"> <li>On</li> <li>Off</li> </ul>	If turned on, enables half-rate DDIO.
Separate input / output Clocks	<ul style="list-style-type: none"> <li>Data Direction = Bidir</li> <li>Register mode = Simple register or DDIO</li> </ul>	<ul style="list-style-type: none"> <li>On</li> <li>Off</li> </ul>	If turned on, enables separate clocks (CK_IN and CK_OUT) for the input and output paths in bidirectional mode.

### Related Information

- [Input and Output Bus High and Low Bits](#) on page 12
- [Guideline: Swap datain\\_h and datain\\_l Ports in Migrated IP](#) on page 23

## Register Packing

The GPIO IP core allows you to pack register into the periphery to save area and resource utilization.

You can configure the full-rate DDIO on the input and output path as a flip flop. To do so, add the .qsf assignments listed in this table.

**Table 12. Register Packing QSF Assignments**

Path	QSF Assignment
Input register packing	<code>set_instance_assignment -name FAST_INPUT_REGISTER ON -to &lt;path to register&gt;</code>
Output register packing	<code>set_instance_assignment -name FAST_OUTPUT_REGISTER ON -to &lt;path to register&gt;</code>
Output enable register packing	<code>set_instance_assignment -name FAST_OUTPUT_ENABLE_REGISTER ON -to &lt;path to register&gt;</code>

**Note:** These assignments do not guarantee register packing. However, these assignments enable the Fitter to find a legal placement. Otherwise, the Fitter keeps the flip flop in the core.

## GPIO Intel FPGA IP Timing

The performance of the GPIO IP core depends on the I/O constraints and clock phases. To validate the timing for your GPIO configuration, Intel recommends that you use the Timing Analyzer.

### Related Information

[The Intel Quartus Prime Timing Analyzer](#)

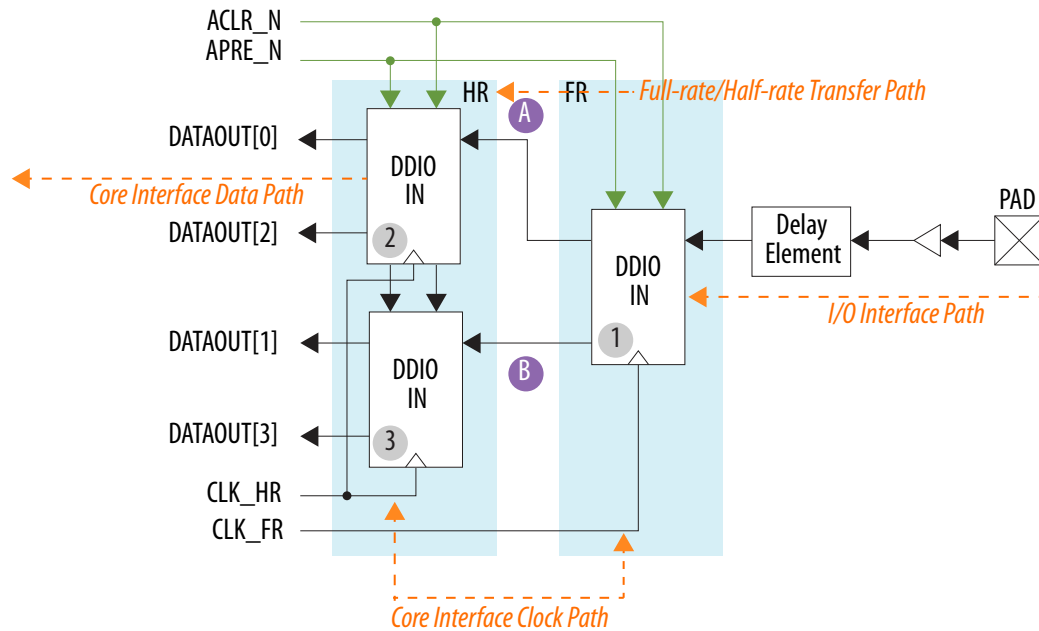
## Timing Components

The GPIO IP core timing components consist of three paths.

- I/O interface paths—from the FPGA to external receiving devices and from external transmitting devices to the FPGA.
- Core interface paths of data and clock—from the I/O to the core and from the core to I/O.
- Transfer paths—from half-rate to full-rate DDIO, and from full-rate to half-rate DDIO.

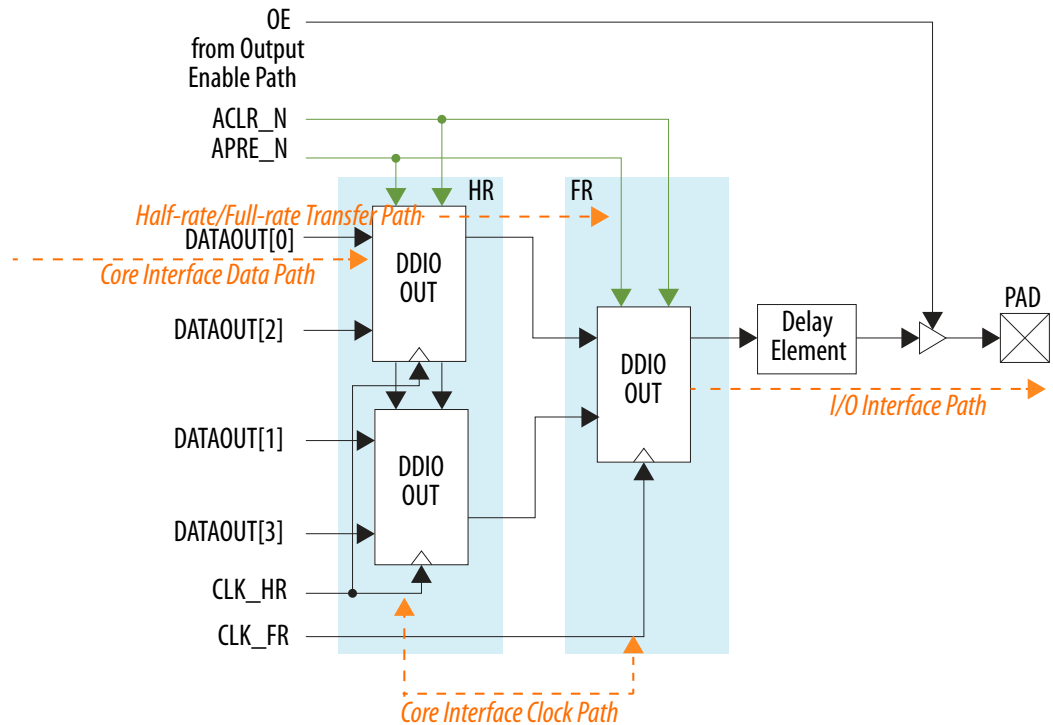
*Note:* The Timing Analyzer treats the path inside the DDIO\_IN and DDIO\_OUT blocks as black boxes.

**Figure 10. Input Path Timing Components**

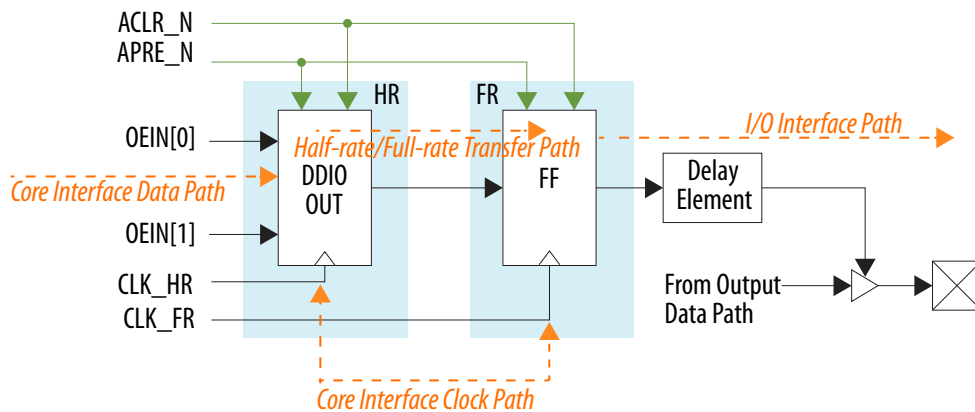




**Figure 11. Output Path Timing Components**



**Figure 12. Output Enable Path Timing Components**



## Delay Elements

The Intel Quartus Prime software does not automatically set delay elements to maximize slack in the I/O timing analysis. To close the timing or maximize slack, set the delay elements manually in the Intel Quartus Prime settings file (.qsf).

**Table 13. Delay Elements .qsf Assignments**

Specify these assignments in the .qsf to access the delay elements.

Delay Element	.qsf Assignment
Input Delay Element	<code>set_instance_assignment -to &lt;PIN&gt; -name INPUT_DELAY_CHAIN &lt;0..63&gt;</code>
Output Delay Element	<code>set_instance_assignment -to &lt;PIN&gt; -name OUTPUT_DELAY_CHAIN &lt;0..15&gt;</code>
Output Enable Delay Element	<code>set_instance_assignment -to &lt;PIN&gt; -name OE_DELAY_CHAIN &lt;0..15&gt;</code>

## Timing Analysis

The Intel Quartus Prime software does not automatically generate the SDC timing constraints for the GPIO IP core. You must manually enter the timing constraints.

Follow the timing guidelines and examples to ensure that the Timing Analyzer analyzes the I/O timing correctly.

- To perform proper timing analysis for the I/O interface paths, specify the system level constraints of the data pins against the system clock pin in the .sdc file.
- To perform proper timing analysis for the core interface paths, define these clock settings in the .sdc file:
  - Clock to the core registers
  - Clock to the I/O registers for the simple register and DDIO modes

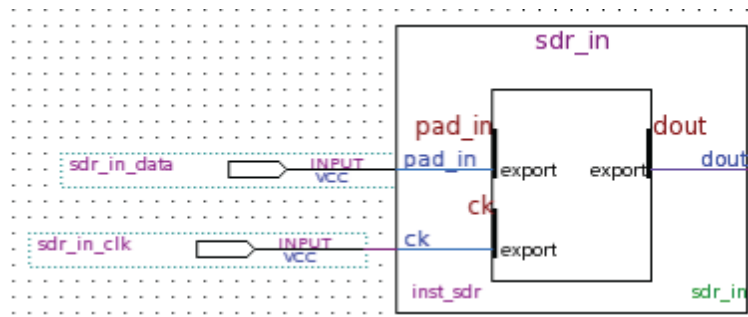
### Related Information

[AN 433: Constraining and Analyzing Source-Synchronous Interfaces](#)

Describes techniques for constraining and analyzing source-synchronous interfaces.

## Single Data Rate Input Register

**Figure 13. Single Data Rate Input Register**



**Table 14. Single Data Rate Input Register .sdc Command Examples**

Command	Command Example	Description
create_clock	<code>create_clock -name sdr_in_clk -period "100 MHz" sdr_in_clk</code>	Creates clock setting for the input clock.
set_input_delay	<code>set_input_delay -clock sdr_in_clk 0.15 sdr_in_data</code>	Instructs the Timing Analyzer to analyze the timing of the input I/O with a 0.15 ns input delay.

## Full-Rate or Half-Rate DDIO Input Register

The input side of the full-rate and half-rate DDIO input registers are the same. You can properly constrain the system by using a virtual clock to model the off-chip transmitter to the FPGA.

Figure 14. Full-Rate or Half-Rate DDIO Input Register

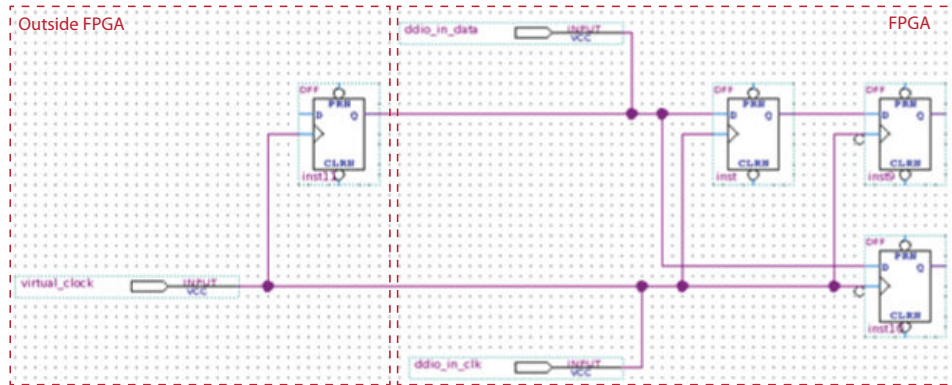


Table 15. Full-Rate or Half-Rate DDIO Input Register .sdc Command Examples

Command	Command Example	Description
create_clock	<pre>create_clock -name virtual_clock -period "200 MHz" create_clock -name ddio_in_clk -period "200 MHz" ddio_in_clk</pre>	Create clock setting for the virtual clock and the DDIO clock.
set_input_delay	<pre>set_input_delay -clock virtual_clock 0.25 ddio_in_data set_input_delay -add_delay -clock_fall -clock virtual_clock 0.25 ddio_in_data</pre>	Instruct the Timing Analyzer to analyze the positive clock edge and the negative clock edge of the transfer. Note the <code>-add_delay</code> in the second <code>set_input_delay</code> command.
set_false_path	<pre>set_false_path -fall_from virtual_clock -rise_to ddio_in_clk set_false_path -rise_from virtual_clock -fall_to ddio_in_clk</pre>	Instruct the Timing Analyzer to ignore the positive clock edge to the negative edge triggered register, and the negative clock edge to the positive edge triggered register.  <i>Note:</i> The <code>ck_hr</code> frequency must be half the <code>ck_fr</code> frequency. If the I/O PLL drives the clocks, you can consider using the <code>derive_pll_clocks .sdc</code> command.

## Single Data Rate Output Register

Figure 15. Single Data Rate Output Register

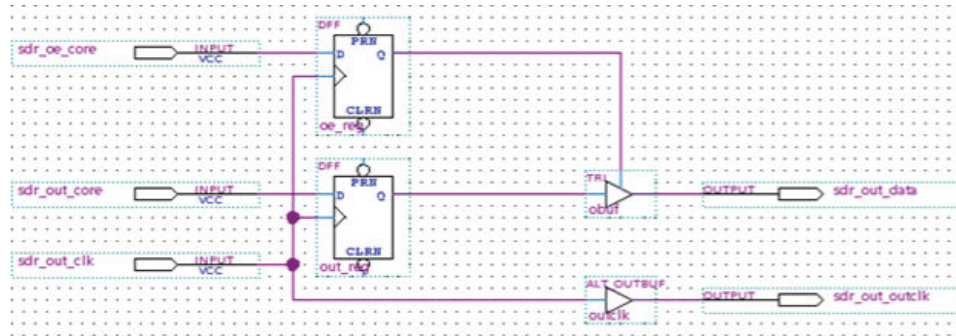


Table 16. Single Data Rate Output Register .sdc Command Examples

Command	Command Example	Description
create_clock and create_generated_clock	create_clock -name sdr_out_clk -period "100 MHz" sdr_out_clk create_generated_clock -source sdr_out_clk -name sdr_out_outclk sdr_out_outclk	Generate the source clock and the output clock to transmit.
set_output_delay	set_output_delay -clock sdr_out_clk 0.45 sdr_out_data	Instructs the Timing Analyzer to analyze the output data to transmit against the output clock to transmit.

## Full-Rate or Half-Rate DDIO Output Register

The output side of the full-rate and half-rate DDIO output registers are the same.

Table 17. DDIO Output Register .sdc Command Examples

Command	Command Example	Description
create_clock and create_generated_clock	create_clock -name ddio_out_fr_clk -period "200 MHz" ddio_out_fr_clk create_generated_clock -source ddio_out_fr_clk -name ddio_out_fr_outclk ddio_out_fr_outclk	Generate the clocks to the DDIO and the clock to transmit.
set_output_delay	set_output_delay -clock ddio_out_fr_outclk 0.55 ddio_out_fr_data set_output_delay -add_delay -clock_fall -clock ddio_out_fr_outclk 0.55 ddio_out_fr_data	Instruct the Timing Analyzer to analyze the positive and negative data against the output clock.
set_false_path	set_false_path -rise_from ddio_out_fr_clk -fall_to ddio_out_fr_outclk set_false_path -fall_from ddio_out_fr_clk -rise_to ddio_out_fr_outclk	Instruct the Timing Analyzer to ignore the rising edge of the source clock against the falling edge of the output clock, and the falling edge of source clock against rising edge of output clock

## Timing Closure Guidelines

For the GPIO input registers, the input I/O transfer is likely to fail the hold time if you do not set the input delay chain. This failure is caused by the clock delay being larger than the data delay.

To meet the hold time, add delay to the input data path using the input delay chain. In general, the input delay chain is around 60 ps per step at the -1 speed grade. To get an approximate input delay chain setting to pass the timing, divide the negative hold slack by 60 ps.

However, if the I/O PLL drives the clocks of the GPIO input registers (simple register or DDIO mode), you can set the compensation mode to source synchronous mode. The Fitter will attempt to configure the I/O PLL for a better setup and hold slack for the input I/O timing analysis.

For the GPIO output and output enable registers, you can add delay to the output data and clock using the output and output enable delay chains.

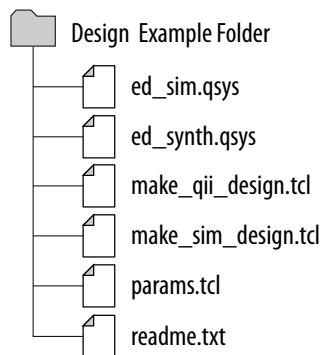
- If you observe setup time violation, you can increase the output clock delay chain setting.
- If you observe hold time violation, you can increase the output data delay chain setting.

## GPIO Intel FPGA IP Design Examples

The GPIO IP core can generate design examples that match your IP configuration in the parameter editor. You can use these design examples as references for instantiating the IP core and the expected behavior in simulations.

You can generate the design examples from the GPIO IP core parameter editor. After you setting the parameters that you want, click **Generate Example Design**. The IP core generates the design example source files in the directory you specify.

**Figure 16. Source Files in the Generated Design Example Directory**



**Note:** The `.qsys` files are for internal use during design example generation only. You cannot edit these `.qsys` files.

## GPIO IP Core Synthesizable Intel Quartus Prime Design Example

The synthesizable design example is a compilation-ready Platform Designer system that you can include in an Intel Quartus Prime project.

### Generating and Using the Design Example

To generate the synthesizable Intel Quartus Prime design example from the source files, run the following command in the design example directory:

```
quartus_sh -t make_qii_design.tcl
```

To specify an exact device to use, run the following command:

```
quartus_sh -t make_qii_design.tcl [device_name]
```

The TCL script creates a `qii` directory that contains the `ed_synth.qpf` project file. You can open and compile this project in the Intel Quartus Prime software.

## GPIO IP Core Simulation Design Example

The simulation design example uses your GPIO IP core parameter settings to build the IP instance connected to a simulation driver. The driver generates random traffic and internally checks the legality of the out going data.

Using the design example, you can run a simulation using a single command, depending on the simulator that you use. The simulation demonstrates how you can use the GPIO IP core.

### Generating and Using the Design Example

To generate the simulation design example from the source files for a Verilog simulator, run the following command in the design example directory:

```
quartus_sh -t make_sim_design.tcl
```

To generate the simulation design example from the source files for a VHDL simulator, run the following command in the design example directory:

```
quartus_sh -t make_sim_design.tcl VHDL
```

The TCL script creates a `sim` directory that contains subdirectories—one for each supported simulation tool. You can find the scripts for each simulation tool in the corresponding directories.

## IP Migration Flow for Arria V, Cyclone V, and Stratix V Devices

The IP migration flow allows you to migrate the `ALTDDIO_IN`, `ALTDDIO_OUT`, `ALTDDIO_BIDIR`, and `ALTIOBUF` IP cores of Arria V, Cyclone V, and Stratix V devices to the GPIO IP core of Intel Arria 10 and Intel Cyclone 10 GX devices.

This IP migration flow configures the GPIO IP core to match the settings of the `ALTDDIO_IN`, `ALTDDIO_OUT`, `ALTDDIO_BIDIR`, and `ALTIOBUF` IP cores, allowing you to regenerate the IP core.

*Note:* Some IP cores support the IP migration flow in specific modes only. If your IP core is in a mode that is not supported, you may need to run the IP Parameter Editor for the GPIO IP core and configure the IP core manually.

## Migrating Your ALTDDIO\_IN, ALTDDIO\_OUT, ALTDDIO\_BIDIR, and ALTIobuf IP Cores

To migrate your ALTDDIO\_IN, ALTDDIO\_OUT, ALTDDIO\_BIDIR, and ALTIobuf IP cores to the GPIO Intel FPGA IP IP core, follow these steps:

1. Open your ALTDDIO\_IN, ALTDDIO\_OUT, ALTDDIO\_BIDIR, or ALTIobuf IP core in the IP Parameter Editor.
2. In the **Currently selected device family**, select **Intel Arria 10** or **Intel Cyclone 10 GX**.
3. Click **Finish** to open the GPIO IP Parameter Editor.  
The IP Parameter Editor configures the GPIO IP core settings similar to the ALTDDIO\_IN, ALTDDIO\_OUT, ALTDDIO\_BIDIR, or ALTIobuf core settings.
4. If there are any incompatible settings between the two, select **new supported settings**.
5. Click **Finish** to regenerate the IP core.
6. Replace your ALTDDIO\_IN, ALTDDIO\_OUT, ALTDDIO\_BIDIR, or ALTIobuf IP core instantiation in RTL with the GPIO IP core.

*Note:* The GPIO IP core port names may not match the ALTDDIO\_IN, ALTDDIO\_OUT, ALTDDIO\_BIDIR, or ALTIobuf IP core port names. Therefore, simply changing the IP core name in the instantiation may not be sufficient.

### Related Information

[Input and Output Bus High and Low Bits](#) on page 12

## Guideline: Swap datain\_h and datain\_l Ports in Migrated IP

When you migrate your GPIO IP from previous devices to the GPIO IP core, you can turn on **Use legacy top-level port names** option in the GPIO IP core parameter editor. However, the behavior of these ports in the GPIO IP core is different than in the IP cores used for the Stratix V, Arria V, and Cyclone V devices.

The GPIO IP core drives these ports to the output registers on these clock edges:

- datain\_h—on the rising edge of outclock
- datain\_l—on the falling edge of outclock

If you migrated your GPIO IP from Stratix V, Arria V, and Cyclone V devices, swap the datain\_h and datain\_l ports when you instantiate the IP generated by the GPIO IP core.

### Related Information

[Input and Output Bus High and Low Bits](#) on page 12

## GPIO Intel FPGA IP User Guide Archives

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
20.4	<a href="#">GPIO Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices</a>
19.3	<a href="#">GPIO Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices</a>
18.1	<a href="#">GPIO Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices</a>
18.0	<a href="#">GPIO Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices</a>
17.1	<a href="#">Intel FPGA GPIO IP Core User Guide</a>
17.0	<a href="#">Altera GPIO IP Core User Guide</a>
16.1	<a href="#">Altera GPIO IP Core User Guide</a>
16.0	<a href="#">Altera GPIO IP Core User Guide</a>
14.1	<a href="#">Altera GPIO Megafunction User Guide</a>
13.1	<a href="#">Altera GPIO Megafunction User Guide</a>

## Document Revision History for GPIO Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices

Document Version	Intel Quartus Prime Version	Changes
2021.03.29	21.1	Updated the GPIO IP version number to 20.0.0.
2021.03.12	20.4	Updated the IP migration guideline to specify that the GPIO IP drives <code>datain_h</code> on the rising edge and <code>datain_l</code> on the falling edge.
2019.10.01	19.3	Corrected typographical error in the <code>.qsif</code> assignment codes in the topic about delay elements.
2019.03.04	18.1	In the topics about the input path, and output and output enable paths: <ul style="list-style-type: none"> <li>Corrected the notes in the topics to specify that the GPIO Intel FPGA IP does not support dynamic calibration of bidirectional pins.</li> <li>Added links to the <a href="#">PHY Lite for Parallel Interfaces Intel FPGA IP Core User Guide: Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX Devices</a> for more information about applications that require dynamic calibration for bidirectional pins.</li> </ul>
2018.08.28	18.0	<ul style="list-style-type: none"> <li>Retitled the document from <i>Intel FPGA GPIO IP Core User Guide</i> to <i>GPIO Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices</i>.</li> <li>Added a link to the Intel Stratix 10 GPIO IP user guide.</li> <li>Renamed the IP from "Intel FPGA GPIO" to "GPIO Intel FPGA IP".</li> <li>Corrected instances of "clk_fr" and "clk_hr" to "ck_fr" and "ck_hr".</li> <li>Updated the GPIO IP input path and output paths diagrams to show the actual IP core signal names.</li> </ul>



Date	Version	Changes
November 2017	2017.11.06	<ul style="list-style-type: none"> <li>• Added support for Intel Cyclone 10 GX devices.</li> <li>• Updated the signal names in figures to match the signal names in the GPIO IP core.</li> <li>• Added the output path waveform.</li> <li>• Renamed "Altera GPIO IP core" to "Intel FPGA GPIO IP core".</li> <li>• Renamed "Altera IOPLL IP core" to "Intel FPGA IOPLL IP core".</li> <li>• Renamed "TimeQuest Timing Analyzer" to "Timing Analyzer".</li> <li>• Renamed "Qsys" to "Platform Designer".</li> <li>• Clarified that the <code>ASET</code> and <code>ACLR</code> signals are active high.</li> </ul>
May 2017	2017.05.08	<ul style="list-style-type: none"> <li>• Updated the table listing the GPIO buffer parameters to specify the conditions for the <b>Use bus-hold circuitry</b> parameter option.</li> <li>• Rebranded as Intel.</li> </ul>
October 2016	2016.10.31	<ul style="list-style-type: none"> <li>• Updated the input path waveform.</li> <li>• Added a topic describing the high and low bits in the <code>din</code> and <code>dout</code> buses.</li> </ul>
August 2016	2016.08.05	<ul style="list-style-type: none"> <li>• Added notes about dynamic OCT support in the GPIO IP core.</li> <li>• Updated the topic about parameter settings to improve accuracy and clarity.</li> <li>• Updated the section about generating the design example.</li> <li>• Added a guideline topic about behavior of the legacy ports when you migrate to the GPIO IP core from Stratix V, Arria V, and Cyclone V devices.</li> <li>• Rewrote and restructured the document to improve clarity and for ease of reference.</li> <li>• Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li> </ul>
August 2014	2014.08.18	<ul style="list-style-type: none"> <li>• Added timing information.</li> <li>• Added register packing information.</li> <li>• Added <b>Use legacy top-level port names</b> parameter. This is a new parameter.</li> <li>• Added register packing information.</li> <li>• Replaced the term megafunction with IP core.</li> </ul>
November 2013	2013.11.29	Initial release.