
Speedster7t GPIO User Guide (UG112)

Speedster FPGAs

Preliminary Data



Copyrights, Trademarks and Disclaimers

Copyright © 2023 Achronix Semiconductor Corporation. All rights reserved. Achronix, Speedster and VectorPath are registered trademarks, and Speedcore and Speedchip are trademarks of Achronix Semiconductor Corporation. All other trademarks are the property of their prospective owners. All specifications subject to change without notice.

NOTICE of DISCLAIMER: The information given in this document is believed to be accurate and reliable. However, Achronix Semiconductor Corporation does not give any representations or warranties as to the completeness or accuracy of such information and shall have no liability for the use of the information contained herein. Achronix Semiconductor Corporation reserves the right to make changes to this document and the information contained herein at any time and without notice. All Achronix trademarks, registered trademarks, disclaimers and patents are listed at <http://www.achronix.com/legal>.

Preliminary Data

This document contains preliminary information and is subject to change without notice. Information provided herein is based on internal engineering specifications and/or initial characterization data.

Achronix Semiconductor Corporation

2903 Bunker Hill Lane
Santa Clara, CA 95054
USA

Website: www.achronix.com
E-mail : info@achronix.com

Table of Contents

Chapter - 1: Introduction	6
Chapter - 2: Device and Maximum Number of GPIO Ports	7
Chapter - 3: Supported Electrical Standards	8
Chapter - 4: GPIO Bank Features	9
GPIO Pin Types	9
Bank Features	9
Un-Registered I/O	9
Registered I/O	9
DDR Mode	10
SerDes Mode	10
Chapter - 5: I/O Ring and GPIO	11
I/O Ring Files	11
User Design Port List File	13
Chapter - 6: Speedster7t AC7t1500/1550 Ball Assignments by Bank	15
GPIO Pin Naming Convention	15
Ball Assignments	16
I/O Package Diagram	19
Chapter - 7: Speedster7t AC7t800 Ball Assignments by Bank	20
GPIO Pin Naming Convention	20
Simple I/O Bank Naming Convention	20
Ball Assignments	21
I/O Package Diagram	26
Chapter - 8: Speedster7t GPIO IP Bank Types	27
Speedster7t AC7t1500/1550 GPIO IP Bank Types	27
GPIO Bank Placements	28
Speedster7t AC7t800 GPIO IP Bank Types	29
GPIO Bank Placements	29
Simple I/O Bank	30
Chapter - 9: Configuring a GPIO Bank	32

Introduction	32
Step 1: Determine if PLL and NoC IP Are Required	32
Step 2: Create a GPIO Bank	32
Step 3: Creating a GPIO Bank IP Configuration	34
Chapter - 10: Configuring a Simple I/O Bank	47
Introduction	47
Create a Simple I/O Bank	47
Creating a Simple I/O Bank IP Configuration	49
Chapter - 11: Design Implementation and Simulation	54
Introduction	54
Default GPIO Behavior	54
GPIO Port Lists	54
Data and Auxiliary GPIO (Inputs)	55
Data and Auxiliary GPIO (Outputs)	56
Data and Auxiliary GPIO (Inout/ Bi-Directional)	56
Data GPIO with SerDes Mode (INOUT)	58
DDR Mode (Inputs)	58
DDR and SerDes Modes (Outputs)	58
Simulation	59
Chapter - 12: Dynamically Configuring Delays with Speedster7t AC7t1500/1550 GPIO ..	60
Introduction	60
Memory Addressing	61
Configuring the GPIO Bank	63
Pipeline Registers	63
Receive Path	63
Transmit Path	64
Revision History	65

Chapter - 1: Introduction

The Speedster®7t General Purpose I/O (GPIO) are digital pins that can be configured to either read data from external devices and/or output data to control external devices. GPIO pins on Speedster7t FPGAs are designed to provide a flexible, customizable, and efficient means to connect user designs to external components. To accomplish this, Achronix provides a variety of built-in features, options, supported I/O standards, and packages. This document describes the various features, how to configure them, any design considerations to be taken into account, and the tools required to implement them.

GPIO utilization requires knowledge of ACE, IP configuration, and the I/O Ring since the GPIO is part of the I/O Ring along with other interface subsystems such as PCIe and GDDR6. It is necessary to edit the IP configurations and generate I/O Ring files in order for ACE to understand the design requirements before implementing them.

There are two types of GPIO IP banks available in Speedster7t FPGAs:

1. The GPIO Bank
2. The Simple I/O Bank

The banks can be configured using the ACE IP configuration perspective. The type of GPIO IP banks available depends on the Speedster7t FPGA model. Speedster7t AC7t1500/AC7t1550 FPGAs only have the GPIO Bank, while Speedster7t AC7t800 FPGAs have both GPIO and Simple I/O banks. After configuring the GPIO IP bank and generating the I/O Ring files, signals may be added to their top-level port list.

GPIO banks have built-in features such as un-registered I/O, registered I/O, DDR mode, and SerDes mode, offering more options than traditional GPIO. This can simplify the design process and speed up production. Simple I/O banks are closer to traditional GPIO and do not include the registered I/O, DDR mode, or SerDes mode features. Simple I/O banks can only be found on Speedster7t AC7t800 FPGAs.

Speedster7t FPGAs offer various traditional options that can be applied to individual GPIOs, such as pull resistors, slew rate, target impedance, etc.

Speedster7t FPGAs support a range of electrical standards that are widely used in various industries and applications. This makes Speedster7t FPGAs versatile and suitable for a wide range of applications. These electrical standards may include but are not limited to, industry standards for voltage levels, signals, and communication protocols such as GMII, SPI, I²C, I³C, etc.

Speedster7t FPGAs have the ability to dynamically configure GPIO delays during runtime using delay-locked loops (DLL) allowing for more flexible control over timing of the GPIO signals. The built-in memory map provides a user-friendly interface for manipulating these delays dynamically.

By supporting these features, standards, and options, Speedster7t FPGA allows for easy integration and compatibility with existing systems and devices with GPIO.

A GPIO reference design is available from [How do I Download Demonstration and Reference Designs for Speedster7t and Speedcore Devices?](#) The GPIO reference design is a helpful resource for those looking to implement GPIO interfaces in their designs. It provides a tested and verified starting point for implementing GPIO features.

Chapter - 2: Device and Maximum Number of GPIO Ports

There are three types of GPIO available in Speedster7t FPGAs:

1. Clock GPIO – used as either input or output and can also serve as the reference clock for the GPIO bank with optional differential pairs.
2. Data GPIO – capable of being used as input, output, tri-state, or bi-directional ports with optional differential pairs.
3. Auxiliary GPIO – has the same features and usage as data GPIO including optional differential pairs.

Future releases of ACE may allow for the Speedster7t AC7t800 FPGA to enable auxiliary GPIO to serve as the reset source for the GPIO bank.

When evaluating the suitability of the Speedster7t FPGA for your design needs, refer to the following table for the available number of user GPIO ports.

Table 1: Maximum Number of Available GPIO Ports by Device

Device	Configuration	Clock Capable	Data	Auxiliary
AC7t800	Differential	2	34	2
	Single Ended	2	68	4
AC7t1500	Differential	6	22	4
	Single Ended	6	44	8
AC7t1550	Differential	6	22	4
	Single Ended	6	44	8

Note



The number of clock-capable GPIO ports in the preceding table is the same whether used as differential or single-ended. This is because the N side of a clock pair cannot be used as a single-ended port.

Chapter - 3: Supported Electrical Standards

Speedster7t FPGAs include GPIO ports to facilitate communication with external components. These ports support various I/O standards and have the ability to operate at multiple voltages. The following table details the supported I/O standards.

Table 2: Supported General-Purpose I/O Standards

Supported I/O Standard	Supported Voltage (V)	Single-Ended/ Differential
HSTL Class I	1.8	single-ended, differential
HSUL	1.2	
LVCMOS	1.1	
	1.2	
	1.35	
	1.5	
	1.8	
SSTL Class I	1.2	
	1.35	
	1.5	
	1.8	
SSTL Class II	1.8	

Note



LVCMOS is labeled as supporting differential signaling in the table. This is due to the Speedster7t FPGA GPIO pad design which implements a pseudo-differential signal when differential mode is enabled on LVCMOS signals.

Chapter - 4: GPIO Bank Features

The Speedster7t FPGA includes a variety of built-in features that can be applied to a set of GPIO ports within a GPIO Bank. The following is an overview of the Speedster7t AC7t800 FPGA built-in features such as un-registered I/O, registered I/O, SerDes mode, and DDR mode. Refer to [Configuring a GPIO Bank \(see page 32\)](#) for a complete guide on implementing these features. Since the GPIO bank can be found in both the Speedster7t AC7t800 and AC7t1500/AC7t1550 FPGAs, the features described are relevant to all Speedster7t FPGAs. Please note that these features are not relevant to the simple I/O bank found only in the Speedster7t AC7t800 FPGAs.

Before describing the GPIO bank features, it is important to first understand the GPIO pin types present within a GPIO bank.

GPIO Pin Types

There are three types of GPIO pins available in Speedster7t FPGAs:

1. Clock GPIO – can be used as either input or output and can also serve as the reference clock for the GPIO bank with optional differential pairs.
2. Data GPIO – capable of being used as inputs, outputs, tri-state, or bi-directional with optional differential pairs.
3. Auxiliary GPIO – same features and usage as data GPIO and can also be used as differential pairs.

Future releases of ACE may allow for the Speedster7t AC7t800 FPGA to enable auxiliary GPIO to serve as the reset source for the GPIO bank.

Bank Features

**Caution!**

When enabling any of the features in this section, that feature is applied to all the GPIO ports in that bank. It is not possible to apply these features to individual signals.

Un-Registered I/O

Un-registered I/O is the conventional form of GPIO where data or clock signals pass to/from the fabric RTL design asynchronously. This means that even if a clock GPIO is enabled and used as the reference clock for the GPIO bank, the signal is still passed asynchronously. There is no need to select a reference clock when only using unregistered data GPIO.

Registered I/O

Registered I/O provides the ability for GPIO signals to be registered within the GPIO bank. The data GPIO can then be conveniently read or written within the fabric RTL user design. This feature requires that the GPIO bank is given a reference clock, and the data signals are aligned with the bank reference clock.

DDR Mode

DDR Mode is only supported on Speedster7t AC7t800 FPGAs. In DDR Mode, the GPIO bank is capable of sampling and/or driving data on both the rising and falling edges of the clock signal, effectively doubling the amount of data that can be transferred per clock cycle on GPIO ports. DDR Mode is supported in combination with other features, including registered I/O and SerDes mode. This allows for even higher data transfer rates and more efficient use of FPGA resources.

SerDes Mode

SerDes mode allows the creating a SerDes bus with one or more of the ports in the GPIO bank. SerDes mode can be used to create an Rx and/or Tx path.

Chapter - 5: I/O Ring and GPIO

I/O Ring Files

Hardened IP is part of the I/O ring which surrounds the Speedster7t FGPA core. The I/O ring includes GPIO, PLLs, memory interfaces, etc. that are configurable if implemented. Therefore to utilize GPIO pins, the I/O ring must be configured with the IP configuration and I/O designer tools within ACE. After generating the I/O ring IP, timing constraints, placement constraints, etc. are created automatically. It is not necessary to instantiate GPIO macros in the design.

When the GPIO bank is configured, it is then necessary to generate the required I/O ring files in ACE. For additional details on generating I/O ring files by configuring IP, see the section, [Creating an IP Configuration](#), in the *Ace User Guide* (UG070).

The following table lists the relevant I/O ring generated files for GPIO, which may also contain information about the entire user design and not just the GPIO portion.

Table 3: I/O Ring GPIO File List

File	Description
<project_name>_ioring.pdc	Sets constraints for pin placements linking the top-level port list of the design with the I/O ring signals.
<project_name>_ioring.sdc	Sets the RTL design clock constraints.
<project_name>_pins.csv	CSV format file containing a list of GPIO pins to be placed by the I/O ring, with information including the names of the ball and bump, ball number, pad name, polarity, clock capability, reset capability, data capability, clock and reset usage.
<project_name>_pins.html	HTML format file containing information similar to the <project_name>_pins.csv file.
<project_name>_pins.txt	Text format file containing a list of GPIO pins to be placed by the I/O ring, including information similar to the <project_name>_pins.csv file.
<project_name>_sim_defines.f	Includes defines needed to set GPIO clocks and resets in the I/O ring. Must be included in the simulation file list.
<project_name>_sim_defines.svh	Calls configuration functions for any subsystems using full-chip RTL. Must be included in the test bench.

File	Description
<code><project_name>_user_design_port_bindings.svh</code>	Binds signal names in the RTL for signals traveling between the core and the I/O ring to core pins (i.e., a user-renamed I/O ring port <code>gpio_bank_1_clk_0</code> is a named version of a hardware port from the core, e.g., <code>i_user_11_09_mt_00[2]</code> , and must be bound so that it can be properly connected by the tool). See the Device Simulation Model section in the Design Flow User Guide (UG106) for more details.
<code><project_name>_user_design_port_list.svh</code>	A list of ports used to connect signals traveling to and from the fabric RTL and I/O ring from the perspective of the fabric RTL (e.g., SerDes parallel clock output, <code>gpio_bank_1_clk_0_parallel</code> , going to the fabric RTL would be listed as an input). Must be included in the top-level port list.
<code><project_name>_user_design_signal_list.svh</code>	Same as <code><project_name>_user_design_port_list.svh</code> but uses "logic" declaration. Can be included in the test bench.

Note

The file names shown in the table include `<project_name>` which is a placeholder for the ACE project name.

**Important!**

When using Speedster7t FPGAs, it is not possible to leave an enabled GPIO pin disconnected. This is because the I/O ring requires all ports to be constrained, meaning that the input or output pins must at least be connected at the top-level port list. Attempting to leave an enabled GPIO pin disconnected causes ACE to generate the following error message in the Tcl console:

```
Net <Instance Name> on top level Port <Instance Name> does not have a valid pad
connection. Unable to process this pin. Please contact Achronix customer support.
```

Where `<Instance Name>` is the name chosen in the GPIO Bank for a given signal.

User Design Port List File

The file, <project_name>_user_design_port_list.svh, is useful because it confirms the signal names that ACE expects in the top-level port list of the fabric RTL design. Use this file as a reference when creating the top-level port list. The following is an example of this file which uses all of the data and auxiliary GPIO set as inputs and the clock GPIO set as outputs for a Speedster7t AC7t1500/AC7t1550 FPGA.

```

////////////////////////////////////
// ACE GENERATED VERILOG INCLUDE FILE
// Generated on: 2022.12.05 at 15:35:49 PST
// By: ACE 8.8.2
// From project: gpio_user_guide_top
////////////////////////////////////
// User Design Port List Include File
////////////////////////////////////

// Ports for gpio_bank_1
input wire      gpio_bank_1_aux_0,
input wire      gpio_bank_1_aux_1,
input wire      gpio_bank_1_data_0,
input wire      gpio_bank_1_data_1,
input wire      gpio_bank_1_data_2,
input wire      gpio_bank_1_data_3,
input wire      gpio_bank_1_data_4,
input wire      gpio_bank_1_data_5,
input wire      gpio_bank_1_data_6,
input wire      gpio_bank_1_data_7,
output wire     gpio_bank_1_clk_0,
// Ports for gpio_bank_2
input wire      gpio_bank_2_aux_0,
input wire      gpio_bank_2_aux_1,
input wire      gpio_bank_2_data_0,
input wire      gpio_bank_2_data_1,
input wire      gpio_bank_2_data_2,
input wire      gpio_bank_2_data_3,
input wire      gpio_bank_2_data_4,
input wire      gpio_bank_2_data_5,
input wire      gpio_bank_2_data_6,
input wire      gpio_bank_2_data_7,
output wire     gpio_bank_2_clk_0,
// Ports for gpio_bank_3
input wire      gpio_bank_3_data_0,
input wire      gpio_bank_3_data_1,
input wire      gpio_bank_3_data_2,
input wire      gpio_bank_3_data_3,
input wire      gpio_bank_3_data_4,
input wire      gpio_bank_3_data_5,
output wire     gpio_bank_3_clk_0,
// Ports for gpio_bank_4
input wire      gpio_bank_4_aux_0,
input wire      gpio_bank_4_aux_1,
input wire      gpio_bank_4_data_0,
input wire      gpio_bank_4_data_1,
input wire      gpio_bank_4_data_2,
input wire      gpio_bank_4_data_3,
input wire      gpio_bank_4_data_4,

```

```
input wire      gpio_bank_4_data_5,
input wire      gpio_bank_4_data_6,
input wire      gpio_bank_4_data_7,
output wire     gpio_bank_4_clk_0,
// Ports for gpio_bank_5
input wire      gpio_bank_5_aux_0,
input wire      gpio_bank_5_aux_1,
input wire      gpio_bank_5_data_0,
input wire      gpio_bank_5_data_1,
input wire      gpio_bank_5_data_2,
input wire      gpio_bank_5_data_3,
input wire      gpio_bank_5_data_4,
input wire      gpio_bank_5_data_5,
input wire      gpio_bank_5_data_6,
input wire      gpio_bank_5_data_7,
output wire     gpio_bank_5_clk_0,
// Ports for gpio_bank_6
input wire      gpio_bank_6_data_0,
input wire      gpio_bank_6_data_1,
input wire      gpio_bank_6_data_2,
input wire      gpio_bank_6_data_3,
input wire      gpio_bank_6_data_4,
input wire      gpio_bank_6_data_5,
output wire     gpio_bank_6_clk_0,
```

```
////////////////////////////////////////
// End User Design Port List Include File
////////////////////////////////////////
```

Chapter - 6: Speedster7t AC7t1500/1550 Ball Assignments by Bank

GPIO Pin Naming Convention

Speedster7t AC7t1500/1550 FPGA GPIO pins are placed across six different banks. The following table describes the naming convention in ACE for the placement and ball names for GPIO Bank I/O; "c" is the chip side (north or south), "p" is the bank placement (i.e., 0, 1, or 2), and "b" is the individual GPIO number (i.e., 0, 1, 2, etc.) within a bank. Placement names refer to the functional group for which a particular pad is used (e.g., clock, data, auxiliary).

Table 4: GPIO Bank Ball Naming Convention for Speedster7t AC7t1500/1550 FPGAs

Placement Name	Ball/Bump Name	Type	Description
GPIO_[c]_B[p]_CLK_0	BANK_GPIO_[c]0_BYTE[p]_BIT_[b]	Clock	GPIO clock capable ball. This is the P side when used as a differential pair.
GPIO_[c]_B[p]_CLK_1	BANK_GPIO_[c]0_BYTE[p]_BIT_[b]	Clock	GPIO clock capable ball. This is the N side when used as a differential pair.
GPIO_[c]_B[p]_DATA_[b]	BANK_GPIO_[c]0_BYTE[p]_BIT_[b]	Data	Arbitrary user data.
GPIO_[c]_B[p]_AUX_[b]	BANK_GPIO_[c]0_BYTE[p]_BIT_[b]	Auxiliary	Arbitrary user data.

Ball Assignments

The following table lists the Speedster7t AC7t1500/1550 FPGA ball assignments for all available GPIO bank I/O.

Table 5: GPIO Bank Ball Assignments

Ball/Bump Name	Bank	GPIO Type	Ball
GPIO_N0_BYTE0_BIT_10	BANK_GPIO_N0_BYTE0	Auxiliary	AE16
GPIO_N0_BYTE0_BIT_11	BANK_GPIO_N0_BYTE0	Auxiliary	AE17
GPIO_N0_BYTE0_BIT_5	BANK_GPIO_N0_BYTE0	Clock	AA16
GPIO_N0_BYTE0_BIT_4	BANK_GPIO_N0_BYTE0	Clock	AA17
GPIO_N0_BYTE0_BIT_0	BANK_GPIO_N0_BYTE0	Data	W17
GPIO_N0_BYTE0_BIT_1	BANK_GPIO_N0_BYTE0	Data	W16
GPIO_N0_BYTE0_BIT_2	BANK_GPIO_N0_BYTE0	Data	Y17
GPIO_N0_BYTE0_BIT_3	BANK_GPIO_N0_BYTE0	Data	Y16
GPIO_N0_BYTE0_BIT_6	BANK_GPIO_N0_BYTE0	Data	AB17
GPIO_N0_BYTE0_BIT_7	BANK_GPIO_N0_BYTE0	Data	AC16
GPIO_N0_BYTE0_BIT_8	BANK_GPIO_N0_BYTE0	Data	AD16
GPIO_N0_BYTE0_BIT_9	BANK_GPIO_N0_BYTE0	Data	AD17
GPIO_N0_BYTE1_BIT_10	BANK_GPIO_N0_BYTE1	Auxiliary	AN16
GPIO_N0_BYTE1_BIT_11	BANK_GPIO_N0_BYTE1	Auxiliary	AR16
GPIO_N0_BYTE1_BIT_5	BANK_GPIO_N0_BYTE1	Clock	AJ17
GPIO_N0_BYTE1_BIT_4	BANK_GPIO_N0_BYTE1	Clock	AH16
GPIO_N0_BYTE1_BIT_0	BANK_GPIO_N0_BYTE1	Data	AF16
GPIO_N0_BYTE1_BIT_1	BANK_GPIO_N0_BYTE1	Data	AG17
GPIO_N0_BYTE1_BIT_2	BANK_GPIO_N0_BYTE1	Data	AG16
GPIO_N0_BYTE1_BIT_3	BANK_GPIO_N0_BYTE1	Data	AH17
GPIO_N0_BYTE1_BIT_6	BANK_GPIO_N0_BYTE1	Data	AJ16

Ball/Bump Name	Bank	GPIO Type	Ball
GPIO_N0_BYTE1_BIT_7	BANK_GPIO_N0_BYTE1	Data	AK17
GPIO_N0_BYTE1_BIT_8	BANK_GPIO_N0_BYTE1	Data	AL16
GPIO_N0_BYTE1_BIT_9	BANK_GPIO_N0_BYTE1	Data	AM16
GPIO_N0_BYTE2_BIT_5	BANK_GPIO_N0_BYTE2	Clock	AU19
GPIO_N0_BYTE2_BIT_4	BANK_GPIO_N0_BYTE2	Clock	AV19
GPIO_N0_BYTE2_BIT_0	BANK_GPIO_N0_BYTE2	Data	AP17
GPIO_N0_BYTE2_BIT_1	BANK_GPIO_N0_BYTE2	Data	AR17
GPIO_N0_BYTE2_BIT_2	BANK_GPIO_N0_BYTE2	Data	AT17
GPIO_N0_BYTE2_BIT_3	BANK_GPIO_N0_BYTE2	Data	AV18
GPIO_N0_BYTE2_BIT_6	BANK_GPIO_N0_BYTE2	Data	AR19
GPIO_N0_BYTE2_BIT_7	BANK_GPIO_N0_BYTE2	Data	AT20
GPIO_S0_BYTE0_BIT_10	BANK_GPIO_S0_BYTE0	Auxiliary	BL14
GPIO_S0_BYTE0_BIT_11	BANK_GPIO_S0_BYTE0	Auxiliary	BL15
GPIO_S0_BYTE0_BIT_5	BANK_GPIO_S0_BYTE0	Clock	BL8
GPIO_S0_BYTE0_BIT_4	BANK_GPIO_S0_BYTE0	Clock	BK9
GPIO_S0_BYTE0_BIT_0	BANK_GPIO_S0_BYTE0	Data	BL2
GPIO_S0_BYTE0_BIT_1	BANK_GPIO_S0_BYTE0	Data	BL3
GPIO_S0_BYTE0_BIT_2	BANK_GPIO_S0_BYTE0	Data	BL4
GPIO_S0_BYTE0_BIT_3	BANK_GPIO_S0_BYTE0	Data	BL5
GPIO_S0_BYTE0_BIT_6	BANK_GPIO_S0_BYTE0	Data	BL9
GPIO_S0_BYTE0_BIT_7	BANK_GPIO_S0_BYTE0	Data	BL11
GPIO_S0_BYTE0_BIT_8	BANK_GPIO_S0_BYTE0	Data	BL12
GPIO_S0_BYTE0_BIT_9	BANK_GPIO_S0_BYTE0	Data	BL13
GPIO_S0_BYTE1_BIT_10	BANK_GPIO_S0_BYTE1	Auxiliary	BL49

Ball/Bump Name	Bank	GPIO Type	Ball
GPIO_S0_BYTE1_BIT_11	BANK_GPIO_S0_BYTE1	Auxiliary	BL50
GPIO_S0_BYTE1_BIT_5	BANK_GPIO_S0_BYTE1	Clock	BL43
GPIO_S0_BYTE1_BIT_4	BANK_GPIO_S0_BYTE1	Clock	BK42
GPIO_S0_BYTE1_BIT_0	BANK_GPIO_S0_BYTE1	Data	BL39
GPIO_S0_BYTE1_BIT_1	BANK_GPIO_S0_BYTE1	Data	BL40
GPIO_S0_BYTE1_BIT_2	BANK_GPIO_S0_BYTE1	Data	BL41
GPIO_S0_BYTE1_BIT_3	BANK_GPIO_S0_BYTE1	Data	BL42
GPIO_S0_BYTE1_BIT_6	BANK_GPIO_S0_BYTE1	Data	BK44
GPIO_S0_BYTE1_BIT_7	BANK_GPIO_S0_BYTE1	Data	BL45
GPIO_S0_BYTE1_BIT_8	BANK_GPIO_S0_BYTE1	Data	BL46
GPIO_S0_BYTE1_BIT_9	BANK_GPIO_S0_BYTE1	Data	BL47
GPIO_S0_BYTE2_BIT_5	BANK_GPIO_S0_BYTE2	Clock	AY23
GPIO_S0_BYTE2_BIT_4	BANK_GPIO_S0_BYTE2	Clock	AT22
GPIO_S0_BYTE2_BIT_0	BANK_GPIO_S0_BYTE2	Data	AU20
GPIO_S0_BYTE2_BIT_1	BANK_GPIO_S0_BYTE2	Data	AW22
GPIO_S0_BYTE2_BIT_2	BANK_GPIO_S0_BYTE2	Data	AV22
GPIO_S0_BYTE2_BIT_3	BANK_GPIO_S0_BYTE2	Data	AU22
GPIO_S0_BYTE2_BIT_6	BANK_GPIO_S0_BYTE2	Data	AW23
GPIO_S0_BYTE2_BIT_7	BANK_GPIO_S0_BYTE2	Data	AR23

I/O Package Diagram

ACE includes an I/O Package Diagram view of placed balls. Hovering over a ball shows additional information for that site. The following image shows the ball locations for all GPIO and GPIO VDDIO for a Speedster7t AC7t1500 /1550 FPGA. The package diagram is visible by default, as shown. If not, find this view by selecting **Window** → **Show View...** → **Other...** → **I/O Package Diagram**. Additional information about the I/O Package Diagram can be found in the [ACE User Guide \(UG070\)](#).

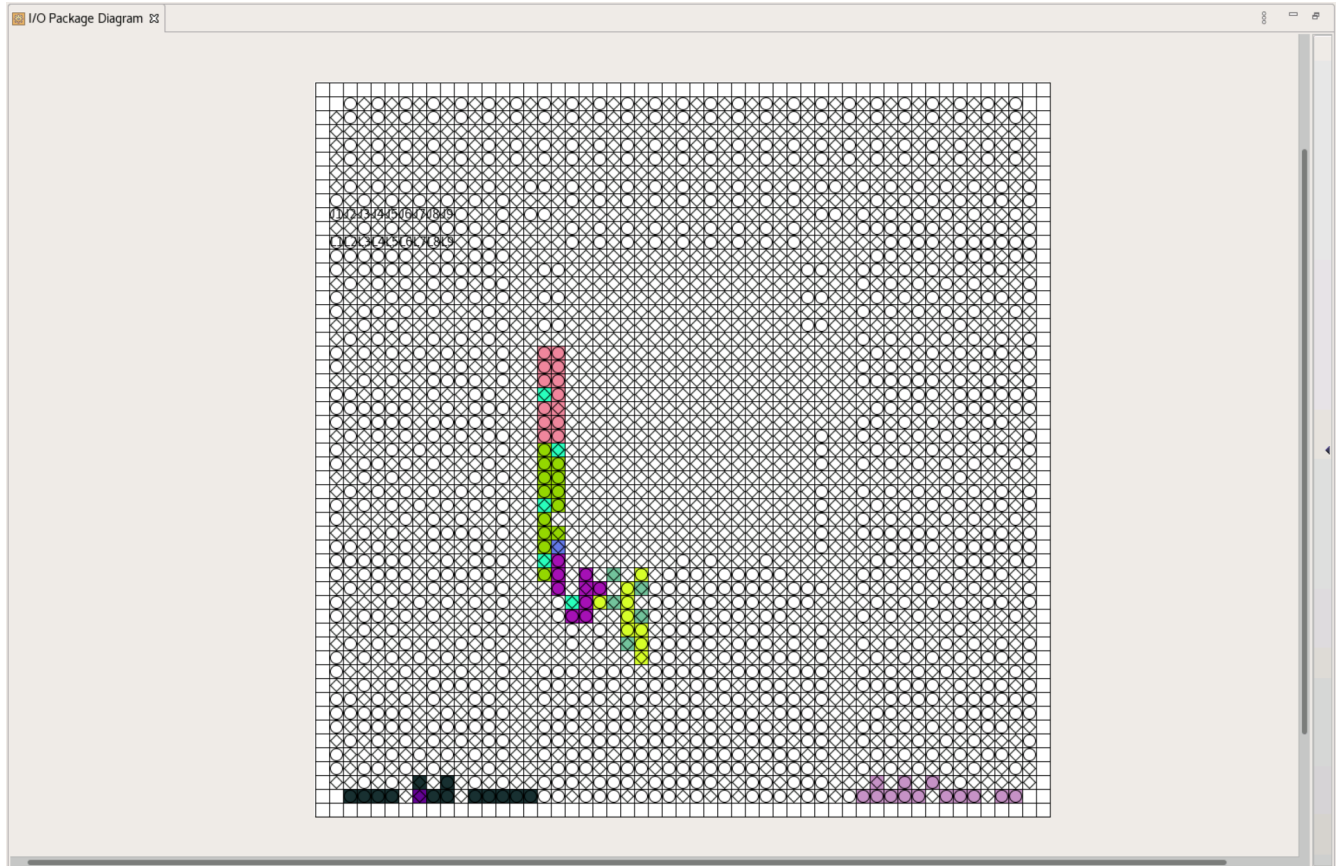


Figure 1: Ball Locations in ACE Package Diagram

Chapter - 7: Speedster7t AC7t800 Ball Assignments by Bank

GPIO Pin Naming Convention

The Speedster7t AC7t800 FPGA GPIO is placed across four different banks. Two of the four banks are GPIO and the other two are simple I/O banks.

The following table describes the ACE naming convention for the placement and ball names for GPIO bank I/O.

Table 6: GPIO Bank Ball Naming Convention for Speedster7t AC7t800 FPGAs

Placement Name	Ball/Bump Name	Type	Description
GPIO_0_B[p]_CLK_0	GPIO_0_BYTE[p]_BIT_[b]	Clock	GPIO clock-capable ball. This is the P side when used as a differential pair.
GPIO_0_B[p]_CLK_1	GPIO_0_BYTE[p]_BIT_[b]	Clock	GPIO clock capable ball. This is the N side when used as a differential pair.
GPIO_0_B[p]_DATA_[b]	GPIO_0_BYTE[p]_BIT_[b]	Data	Arbitrary user data.
GPIO_0_B[p]_AUX_[b]	GPIO_0_BYTE[p]_BIT_[b]	Auxiliary	Arbitrary user data.

"p" is the bank placement (e.g., 0 or 1), and "b" is the individual GPIO number (i.e., 0, 1, 2, etc.) within a bank. Placement names refer to a functional group for which a particular pad is used (e.g., clock, data, auxiliary).

Simple I/O Bank Naming Convention

The following table describes the ACE naming convention for the placement and ball names for simple I/O banks.

Table 7: Simple I/O Bank Ball Naming Convention for Speedster7t AC7t800 FPGAs

Placement Name	Ball/Bump Name	Type	Description
GPIO_[p]_B0_DATA_[b]	GPIO_[p]_[b]	Data	Arbitrary user data.

"p" is the bank placement (e.g., 0 or 1), and "b" is the individual GPIO number (i.e., 0, 1, 2, etc.) within the simple I/O banks.

Ball Assignments

The following table lists the Speedster7t AC7t800 FPGA ball assignments for all available GPIO bank I/O.

Table 8: GPIO Bank Ball Assignments

Ball/Bump Name	Bank	GPIO Type	Ball
GPIO_0_BYTE0_BIT_10	BANK_GPIO_0	Auxiliary	AF5
GPIO_0_BYTE0_BIT_11	BANK_GPIO_0	Auxiliary	AF7
GPIO_0_BYTE0_BIT_5	BANK_GPIO_0	Clock	AG8
GPIO_0_BYTE0_BIT_4	BANK_GPIO_0	Clock	AG6
GPIO_0_BYTE0_BIT_0	BANK_GPIO_0	Data	AH7
GPIO_0_BYTE0_BIT_1	BANK_GPIO_0	Data	AH6
GPIO_0_BYTE0_BIT_2	BANK_GPIO_0	Data	AH8
GPIO_0_BYTE0_BIT_3	BANK_GPIO_0	Data	AH10
GPIO_0_BYTE0_BIT_6	BANK_GPIO_0	Data	AG10
GPIO_0_BYTE0_BIT_7	BANK_GPIO_0	Data	AF2
GPIO_0_BYTE0_BIT_8	BANK_GPIO_0	Data	AF3
GPIO_0_BYTE0_BIT_9	BANK_GPIO_0	Data	AF4
GPIO_0_BYTE1_BIT_10	BANK_GPIO_0	Auxiliary	AE9
GPIO_0_BYTE1_BIT_11	BANK_GPIO_0	Auxiliary	AE10
GPIO_0_BYTE1_BIT_5	BANK_GPIO_0	Clock	AD2
GPIO_0_BYTE1_BIT_4	BANK_GPIO_0	Clock	AD1
GPIO_0_BYTE1_BIT_0	BANK_GPIO_0	Data	AF8
GPIO_0_BYTE1_BIT_1	BANK_GPIO_0	Data	AF9
GPIO_0_BYTE1_BIT_2	BANK_GPIO_0	Data	AE2
GPIO_0_BYTE1_BIT_3	BANK_GPIO_0	Data	AE3
GPIO_0_BYTE1_BIT_6	BANK_GPIO_0	Data	AE4
GPIO_0_BYTE1_BIT_7	BANK_GPIO_0	Data	AE5

Ball/Bump Name	Bank	GPIO Type	Ball
GPIO_0_BYTE1_BIT_8	BANK_GPIO_0	Data	AE6
GPIO_0_BYTE1_BIT_9	BANK_GPIO_0	Data	AE7

The following table lists the Speedster7t AC7t800 FPGA ball assignments for all available simple I/O pins.

Table 9: Simple I/O Bank Ball Assignments

Ball/Bump Name	Bank	GPIO Type	Ball
GPIO_1_0	BANK_GPIO_1	Data	AD3
GPIO_1_1	BANK_GPIO_1	Data	AD4
GPIO_1_2	BANK_GPIO_1	Data	AD5
GPIO_1_3	BANK_GPIO_1	Data	AD7
GPIO_1_4	BANK_GPIO_1	Data	AD8
GPIO_1_5	BANK_GPIO_1	Data	AD9
GPIO_1_6	BANK_GPIO_1	Data	AD10
GPIO_1_7	BANK_GPIO_1	Data	AC3
GPIO_1_8	BANK_GPIO_1	Data	AC4
GPIO_1_9	BANK_GPIO_1	Data	AC6
GPIO_1_10	BANK_GPIO_1	Data	AC7
GPIO_1_11	BANK_GPIO_1	Data	AC8
GPIO_1_12	BANK_GPIO_1	Data	AC9
GPIO_1_13	BANK_GPIO_1	Data	AB2
GPIO_1_14	BANK_GPIO_1	Data	AB4
GPIO_1_15	BANK_GPIO_1	Data	AB5
GPIO_1_16	BANK_GPIO_1	Data	AB6
GPIO_1_17	BANK_GPIO_1	Data	AB8
GPIO_1_18	BANK_GPIO_1	Data	AA1
GPIO_1_19	BANK_GPIO_1	Data	Y1
GPIO_1_20	BANK_GPIO_1	Data	AA2
GPIO_1_21	BANK_GPIO_1	Data	AA4
GPIO_1_22	BANK_GPIO_1	Data	AA5

Ball/Bump Name	Bank	GPIO Type	Ball
GPIO_1_23	BANK_GPIO_1	Data	AA6
GPIO_1_24	BANK_GPIO_1	Data	AA7
GPIO_1_25	BANK_GPIO_1	Data	AA10
GPIO_2_0	BANK_GPIO_2	Data	R2
GPIO_2_1	BANK_GPIO_2	Data	R1
GPIO_2_2	BANK_GPIO_2	Data	T1
GPIO_2_3	BANK_GPIO_2	Data	U8
GPIO_2_4	BANK_GPIO_2	Data	U7
GPIO_2_5	BANK_GPIO_2	Data	U6
GPIO_2_6	BANK_GPIO_2	Data	U4
GPIO_2_7	BANK_GPIO_2	Data	U3
GPIO_2_8	BANK_GPIO_2	Data	U1
GPIO_2_9	BANK_GPIO_2	Data	V8
GPIO_2_10	BANK_GPIO_2	Data	V7
GPIO_2_11	BANK_GPIO_2	Data	V6
GPIO_2_12	BANK_GPIO_2	Data	V4
GPIO_2_13	BANK_GPIO_2	Data	V1
GPIO_2_14	BANK_GPIO_2	Data	W10
GPIO_2_15	BANK_GPIO_2	Data	W8
GPIO_2_16	BANK_GPIO_2	Data	W7
GPIO_2_17	BANK_GPIO_2	Data	W5
GPIO_2_18	BANK_GPIO_2	Data	W4
GPIO_2_19	BANK_GPIO_2	Data	W3
GPIO_2_20	BANK_GPIO_2	Data	Y9
GPIO_2_21	BANK_GPIO_2	Data	Y8

Ball/Bump Name	Bank	GPIO Type	Ball
GPIO_2_22	BANK_GPIO_2	Data	Y6
GPIO_2_23	BANK_GPIO_2	Data	Y5
GPIO_2_24	BANK_GPIO_2	Data	Y3
GPIO_2_25	BANK_GPIO_2	Data	Y2

I/O Package Diagram

ACE includes an I/O package diagram view of placed balls. Hovering over a ball shows additional information for that site. The following image shows the ball locations for all GPIO and GPIO VDDIO pins for a Speedster7t AC7t800 FPGA. The Package Diagram is visible by default, as shown. If not, find this view by selecting **Window** → **Show View...** → **Other...** → **I/O Package Diagram**. Additional information about the I/O package diagram can be found in the [ACE User Guide \(UG070\)](#)".

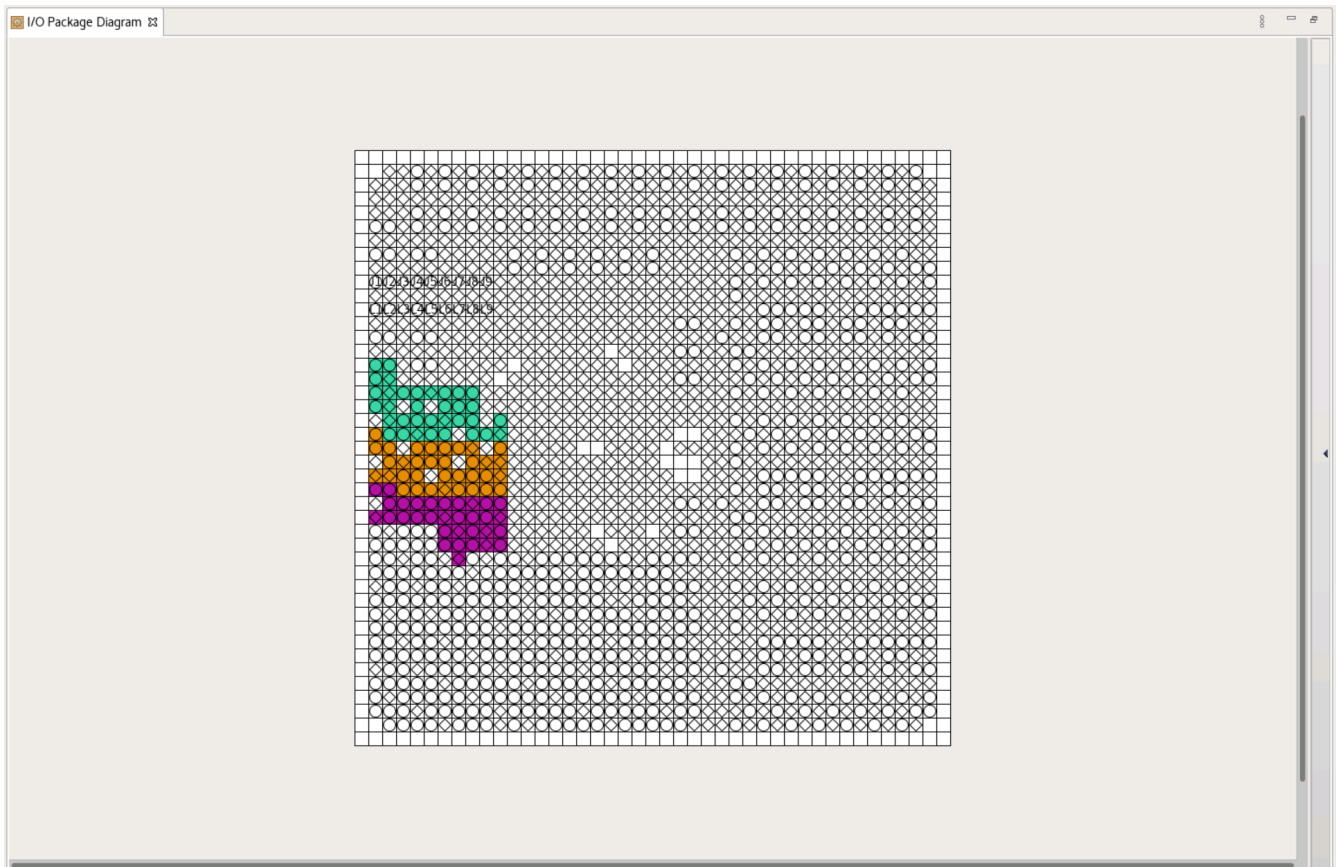


Figure 2: Ball Locations in ACE Package Diagram

Chapter - 8: Speedster7t GPIO IP Bank Types

There are two types of GPIO IP bank within Speedster7t FPGAs:

- GPIO bank
- Simple I/O bank

The Speedster7t AC7t800 FPGA includes both GPIO banks and simple I/O banks while the Speedster7t AC7t1500/1550 FPGAs only include GPIO banks.

Speedster7t AC7t1500/1550 GPIO IP Bank Types

The Speedster7t AC7t1500/1550 GPIO exists in two blocks on the chip. There is no difference in functionality between the two blocks. Each block consists of three banks for a total of six GPIO IP bank types. The Layout diagram is visible by default in ACE, as shown in the following figure. If it is not visible, it can be found by selecting **Window** → **Show View...** → **Other...** → **I/O Layout Diagram**.

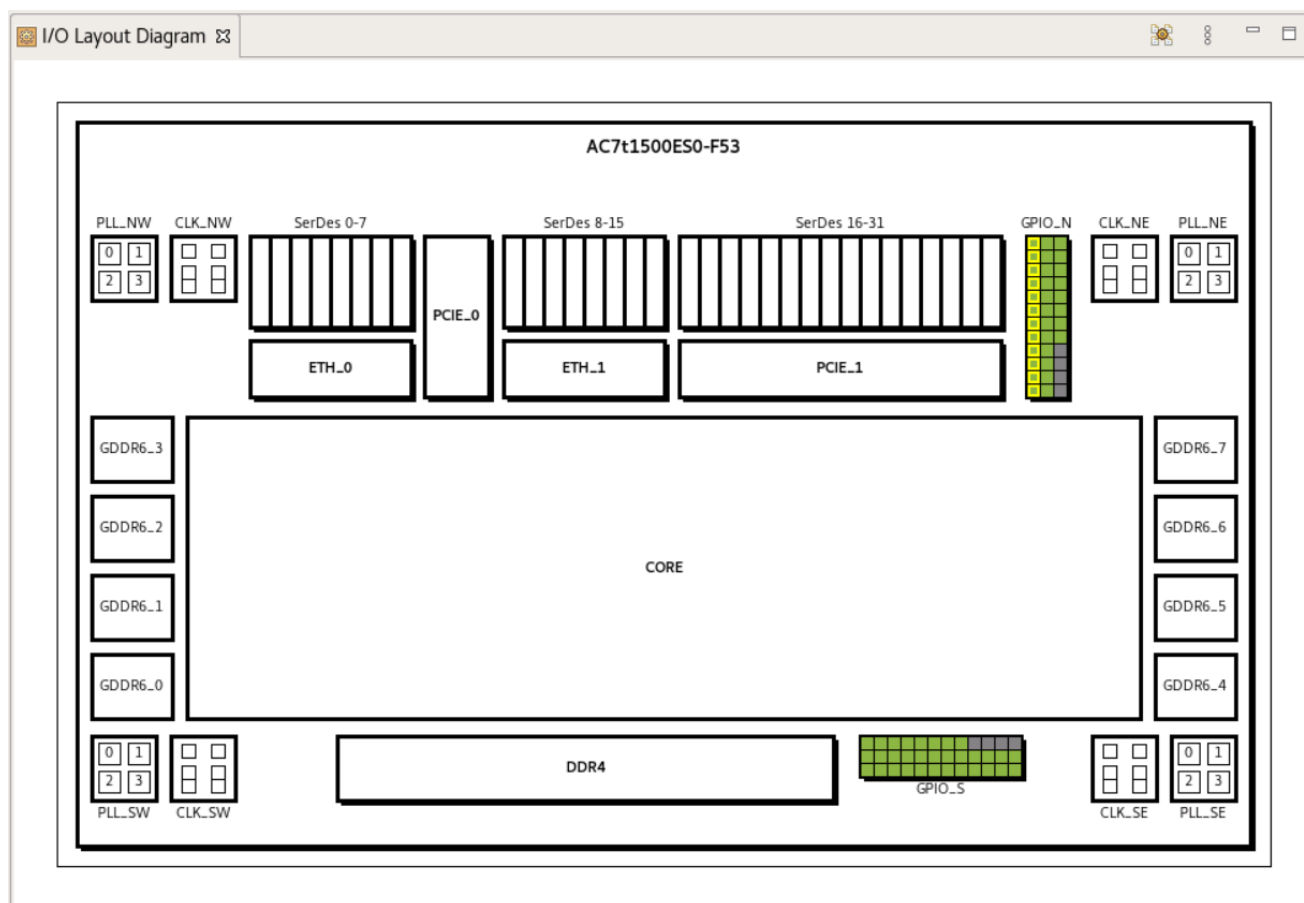


Figure 3: Speedster7t AC7t1500/1550 GPIO Blocks

GPIO Bank Placements

The GPIO bank IP configuration editor, shown in the following figure, is where the options for a given bank are selected. The **Placement** menu is expanded showing the six available GPIO Banks.



Caution!

When enabling any of the following features in the upper portion of the IP configuration editor, that feature is applied to all of the GPIO pins in that bank. It is not possible to apply these features to individual signals.

The screenshot shows the 'Speedster7t GPIO Bank' configuration window. The 'Placement' dropdown menu is open, displaying the following options: GPIO_N_B0, GPIO_N_B1, GPIO_N_B2, GPIO_S_B0, GPIO_S_B1, and GPIO_S_B2. The 'Placement' option is highlighted with a green border. The configuration window includes various settings such as Target Device (AC7t1500ES0), DDR Mode (No), Rx Register Mode (unchecked), Rx Edge Select (Positive edge), Bank Clock Signal Name (gpio_bank_1_clk_0), Bank Reset Source (Internal Reset from FCU), VREF Source (Internal VDD), SerDes Ratio, Tx Register Mode (unchecked), Tx Edge Select, Valid GPIO DLL refclk Frequency, Bank (Parallel) Clock Frequency, GPIO DLL Phase Shift Increment, Bank Global Reset Signal Name (bank_reset), and Bank Voltage Level (1.1).

Figure 4: Speedster7t AC7t1500/1550 Available GPIO Bank Placements

Maximum Number of GPIO Pins for GPIO Banks

Banks B0 and B1 differ slightly from B2 in the number of GPIO pins available. The following applies to both GPIO blocks on the FPGA.

Banks B0 and B1 include:

- 2 clock-capable balls (1 differential pair/1 single-ended)
- 8 data balls
- 2 auxiliary balls

Bank B2 includes:

- 2 clock-capable balls (1 differential pair/1 single-ended)
- 6 data balls

Speedster7t AC7t800 GPIO IP Bank Types

The Speedster7t AC7t800 GPIO exists in two blocks on the chip. These two blocks each contain one of two GPIO IP bank types. Two of these banks are of the GPIO bank type, and the remaining two are of the simple I/O bank type. The layout diagram is visible by default in ACE, as shown in the following figure. If it is not visible, it can be found by selecting **Window** → **Show View...** → **Other...** → **I/O Layout Diagram**.

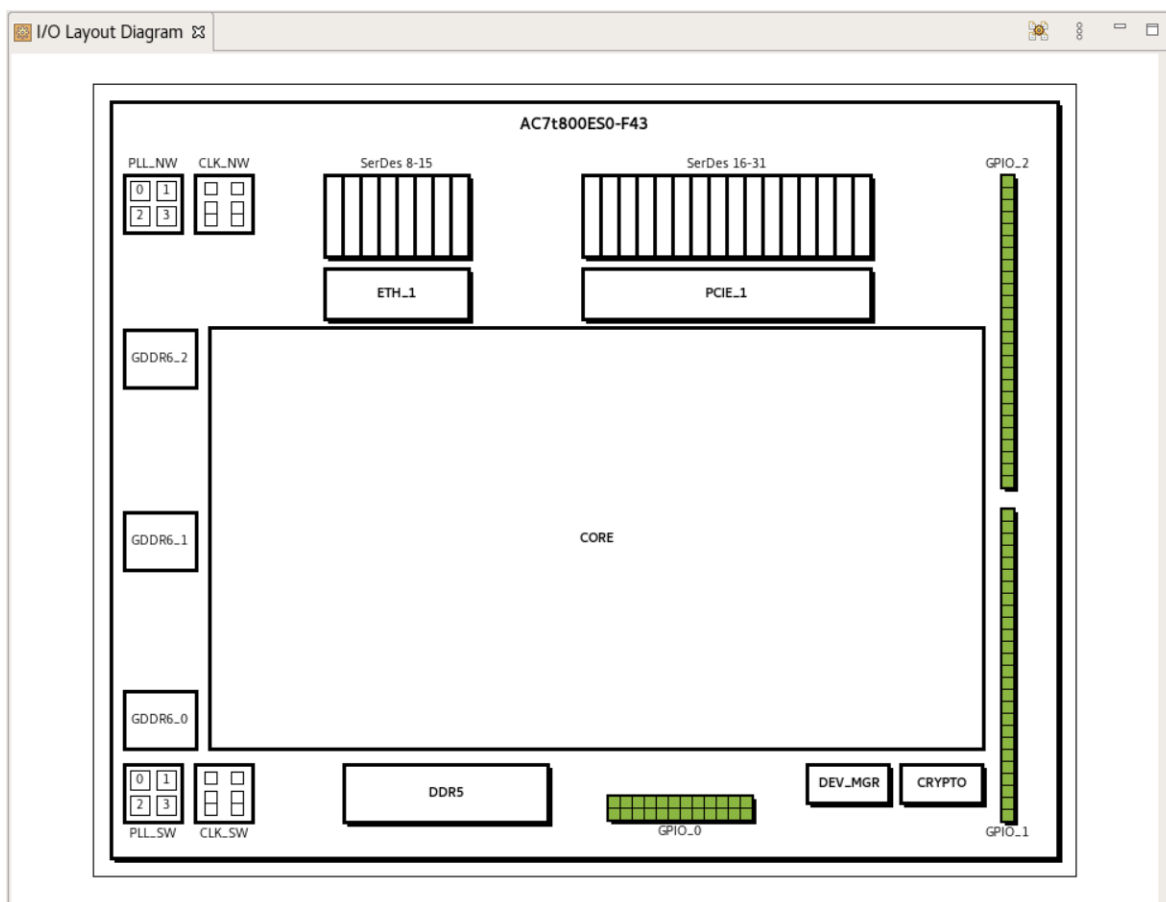


Figure 5: Speedster7t AC7t800 GPIO Blocks

GPIO Bank Placements

The GPIO bank IP configuration editor, shown in the following figure, is where the options for a given bank are selected. The **Placement** menu is expanded showing the two available GPIO banks.



Caution!

When enabling any of the following features in the upper portion of the IP configuration editor, that feature is applied to all of the GPIO pins in that bank. It is not possible to apply these features to individual signals.

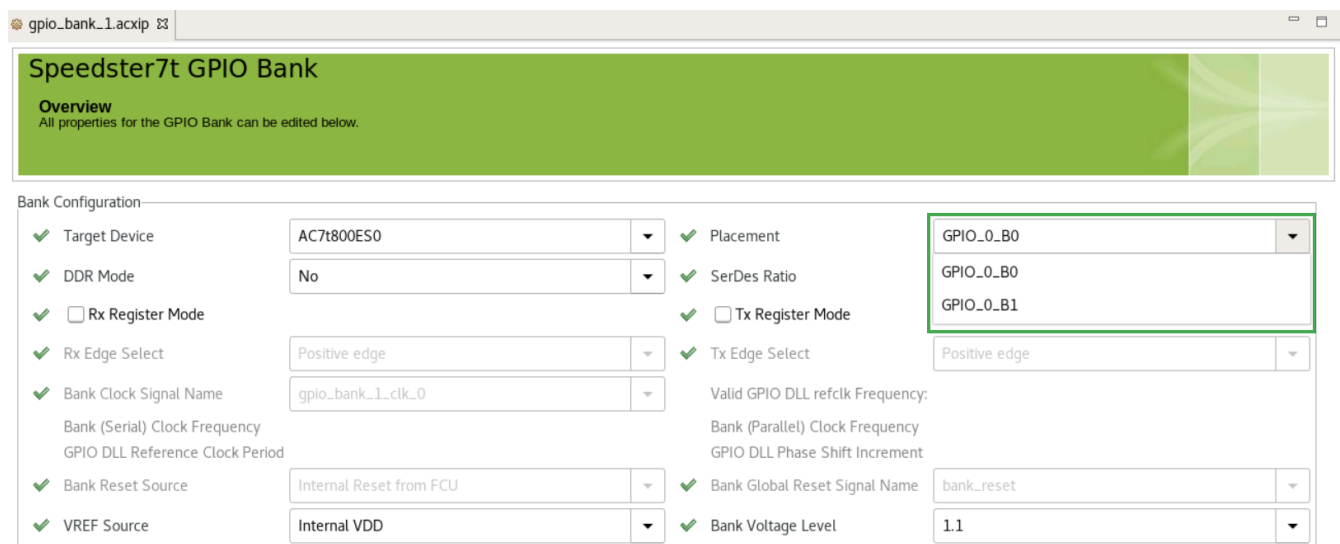


Figure 6: Speedster7t AC7t800 Available GPIO Bank Placements


Maximum Number of GPIO Pins for GPIO Banks

Banks B0 and B1 both include:

- 2 clock-capable balls (1 differential pair/1 single-ended)
- 8 data balls
- 2 auxiliary balls

Simple I/O Bank

The simple I/O bank IP configuration editor, shown in the following figure, is where the options for a given bank are selected. The **Placement** menu is expanded showing the two available simple I/O banks.

**Caution!**

When enabling any of the following features in the upper portion of the IP configuration editor, that feature is applied to all of the GPIO pins in that bank. It is not possible to apply these features to individual signals.

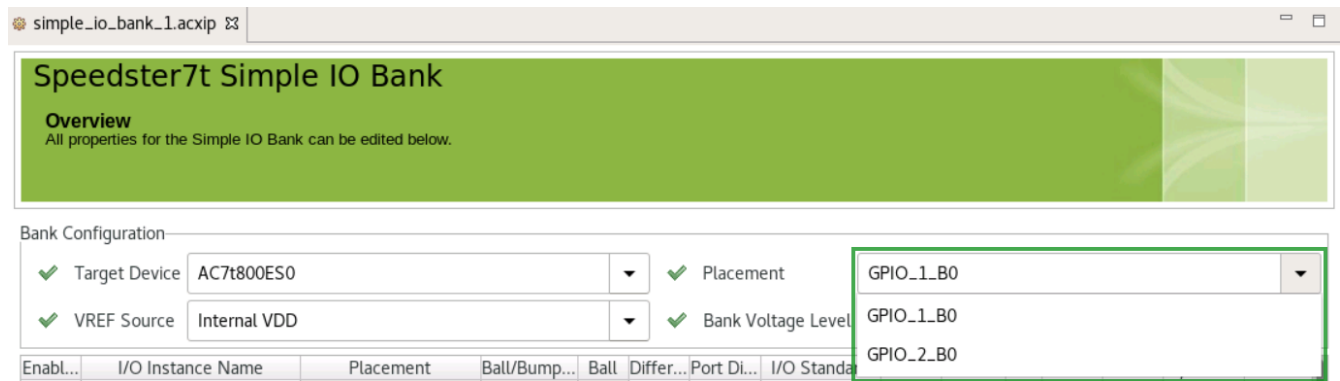


Figure 7: Speedster7t AC7t800 Available Simple I/O Bank Placements

Maximum Number of GPIO Pins for Simple I/O Banks

Banks GPIO_1_B0 and GPIO_2_B0 each include 26 data balls.

Chapter - 9: Configuring a GPIO Bank

Introduction

This section describes the steps required to utilize the GPIO types (clock, data, auxiliary) within a GPIO bank. The I/O ring files must first be generated with ACE by configuring one or more GPIO banks and generating the I/O ring files:

1. Create an ACE project. It is important to set the **Target Device** in the options view after creating a project. ACE only loads the corresponding IP libraries for the target device selected. This page first describes the prerequisite needed for generating PLL and 2D NoC IP for specific GPIO bank features.
2. Follow these steps to create a GPIO bank. Refer to the [ACE User Guide \(UG070\)](#) for additional details about configuring IP and generating I/O ring files.

Step 1: Determine if PLL and NoC IP Are Required

PLL and NoC IP are required if planning to use registered I/O or SerDes mode for Speedster7t AC7t1500/1550 FPGAs. The Speedster7t AC7t800 FPGA does not have this requirement. A PLL may still be used for Speedster7t AC7t800 FPGAs if it is intended to use its output as the reference clock for the GPIO Bank.

If requiring a PLL and 2D NoC only for the GPIO bank, the following steps describe the basic configuration for these IP blocks. All other default settings can be accepted. Otherwise, if planning a more complex design with the 2D NoC, refer to the [Speedster7t 2D Network on Chip User Guide \(UG089\)](#) for detailed information.

1. Configure a PLL IP. The PLL requires an external reference clock and at least one clock output. Select a reference clock from the list of valid clocks. The valid clock options are displayed in the drop-down menu.
2. Enable one output clock with a frequency of 200 MHz. 200 MHz is the minimum required frequency for the 2D NoC.
3. Configure the 2D NoC IP. Select the reference clock name from the list of valid clocks which is the output clock of the PLL chosen in the previous step.

Step 2: Create a GPIO Bank

1. In ACE, navigate to the IP configuration perspective in the top navigation pane.
2. Under **Speedster7t** in the **IP Libraries** pane on the left, navigate to the **IO Ring** section.
3. Double-click **GPIO Bank**.
4. Select a suitable name for the instance of the GPIO bank. The chosen bank name becomes the prefix of the default names of all I/O instances within this bank.

This opens the GPIO bank .acxip file to configure using the IP configuration editor:

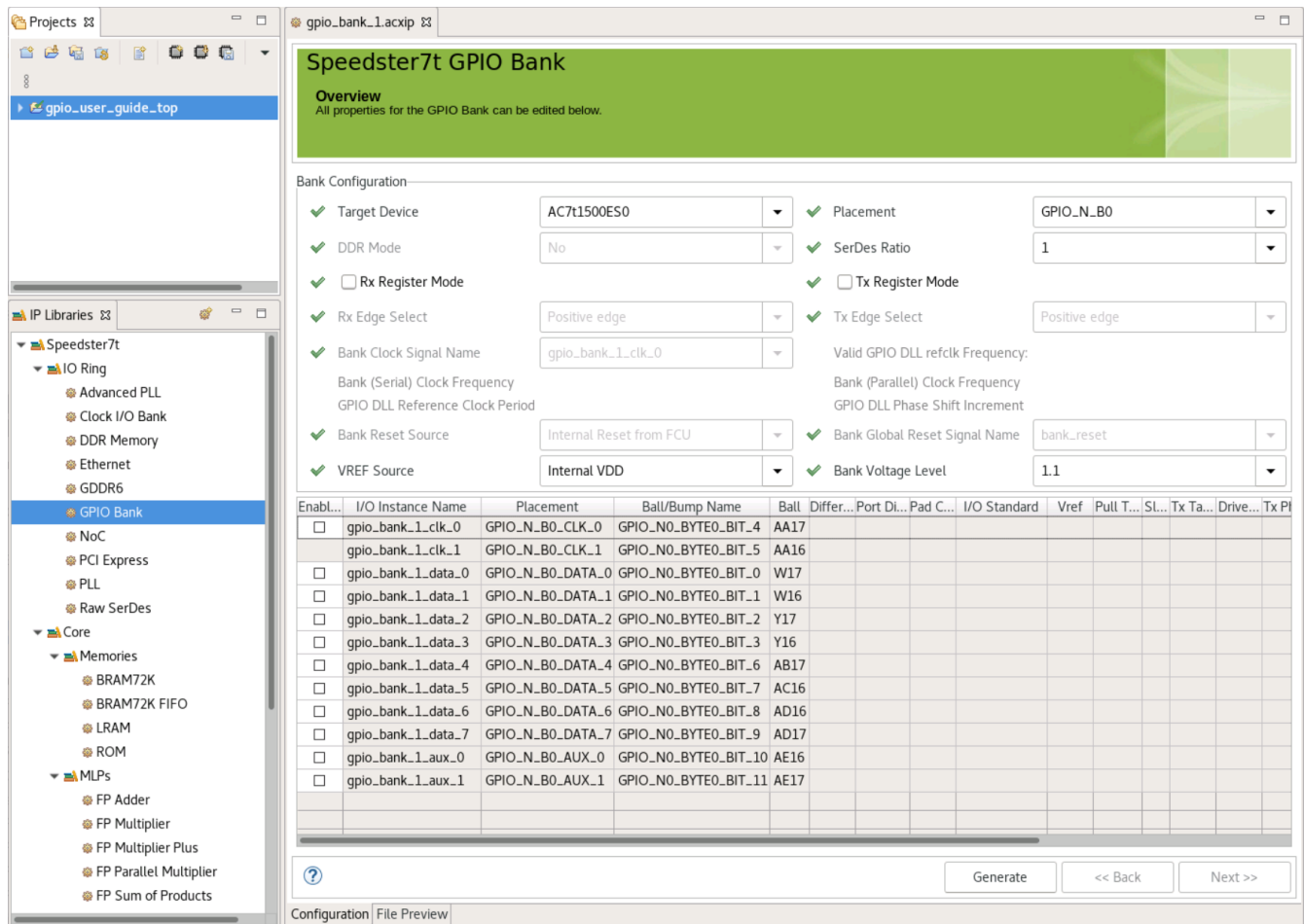



Figure 8: Speedster7t AC7t1500/1550 GPIO Bank IP Configuration Editor

Step 3: Creating a GPIO Bank IP Configuration

Follow these steps in order. Working from left to right and top to bottom is important because some options depend on a prior selected value. Working in the opposite direction may override that value.

**Caution!**

When enabling any of the following features in the upper portion of the IP configuration editor, that feature is applied to all of the GPIO pins in that bank. It is not possible to apply these features to individual signals.

gpio_bank_1.acxp

Speedster7t GPIO Bank

Overview
All properties for the GPIO Bank can be edited below.

Bank Configuration

✓ Target Device

AC7t1500ES0

✓ DDR Mode

No

✓ ☐ Rx Register Mode

✓ Rx Edge Select

Positive edge

✓ Bank Clock Signal Name

pll_clkout

Bank (Serial) Clock Frequency

GPIO DLL Reference Clock Period

✓ Bank Reset Source

Internal Reset from FCU

✓ VREF Source

Internal VDD

✓ Placement

GPIO_N_B0

✓ SerDes Ratio

1

✓ ☐ Tx Register Mode

✓ Tx Edge Select

Positive edge

Valid GPIO DLL refclk Frequency:

Bank (Parallel) Clock Frequency

GPIO DLL Phase Shift Increment

✓ Bank Global Reset Signal Name

clock_io_bank_1_refio_p_0

✓ Bank Voltage Level

1.1

Figure 9: Global Bank Feature Settings

1. Verify the **Target Device**. This is the Speedster7t FPGA model to be targeted. The target device should have been selected when creating an ACE project in the options view. If the target device selected in the bank IP configuration editor does not match the device for the project, ACE shows a "Target device mismatch" error in the **IP Problems** tab. If there is an error in the tab due to a target device conflict, click the error in question for further details to understand the cause of the issue.

2. Select the GPIO bank **Placement** desired:

- For Speedster7t AC7t1500/1550 FPGAs, the placement can be one of six available GPIO banks:
 1. GPIO_N_B0
 2. GPIO_N_B1
 3. GPIO_N_B2
 4. GPIO_S_B0
 5. GPIO_S_B1
 6. GPIO_S_B2
- For Speedster7t AC7t800 FPGAs, the placement can be one of two available GPIO banks:
 1. GPIO_0_B0
 2. GPIO_0_B1.

Refer to [Speedster7t GPIO IP Bank Types \(see page 27\)](#) for specific information about the number of GPIO pins available on the banks mentioned above.

3. Select **yes** or **no** from the drop-down menu to enable or disable the **DDR Mode** if the feature is required.**Note**

DDR mode is always grayed out for Speedster7t AC7t1500/1550 FPGAs because it is not supported for these models.

Enabling the DDR mode feature reduces the maximum SerDes ratio to 4 in the next step, which means that the maximum data transfer rate for SerDes mode is limited. This is because a model without DDR mode supports a maximum SerDes ratio of 8. Therefore, it is important to consider the trade-offs between the benefits of DDR mode and SerDes mode.

4. Select a **SerDes Ratio** if the use of the SerDes mode feature is required.**Note**

All enabled GPIO pins in the GPIO bank share a single SerDes ratio configuration. It is not possible to have multiple SerDes ratio configurations within a single GPIO bank. Additionally, it is not possible to have both SerDes GPIO and non-SerDes GPIO in the same GPIO bank.

- To utilize SerDes mode, select the desired SerDes ratio when configuring the GPIO bank. When the SerDes ratio is set to 1, the SerDes mode is disabled. When the SerDes ratio is set to a value between 2 and 8, the SerDes mode is enabled with the selected ratio. The maximum SerDes ratio available is 8 with DDR mode disabled and 4 for when DDR mode is enabled. The SerDes ratio chosen indicates the serial-to-parallel (Rx) and parallel-to-serial (Tx) ratio to be used for any data or auxiliary GPIO enabled. That means, when the SerDes mode has been enabled, all signals in the bank that are enabled are then included in an Rx or Tx SerDes interface. The GPIO direction indicates if the SerDes bus is an Rx or Tx bus. It is up to the user to choose which of the available GPIO are enabled and thus used for the SerDes interface.

The SerDes ratio can implement various ratios up to an Rx bus of 1:8 or a Tx bus of 8:1 for when DDR mode is disabled. When DDR mode is enabled, the maximum ratio is up to an Rx bus of 1:4 or a Tx bus of 4:1.

For an Rx SerDes bus, the GPIO bank implements a serial-to-parallel path. A single external serial input to the FPGA appears parallel to the fabric RTL design. An example of the top-level signals for a data GPIO that is configured as an input with a SerDes ratio of 8, follows. Notice the single data GPIO enabled in the bank IP configuration editor is now presented to the fabric RTL design as a vector of width 8.

```
input wire      gpio_bank_1_clk_0_parallel
input wire [7:0] gpio_bank_1_data_0
```

For a Tx SerDes bus, the GPIO bank implements a parallel-to-serial path. A single output from the FPGA combines a parallel bus from the fabric RTL design. An example of the top-level signals for a data GPIO that is configured as an output with a SerDes ratio of 8, follows. Notice the single data GPIO enabled in the bank IP configuration editor is now presented to the fabric RTL design as a vector of width 8.

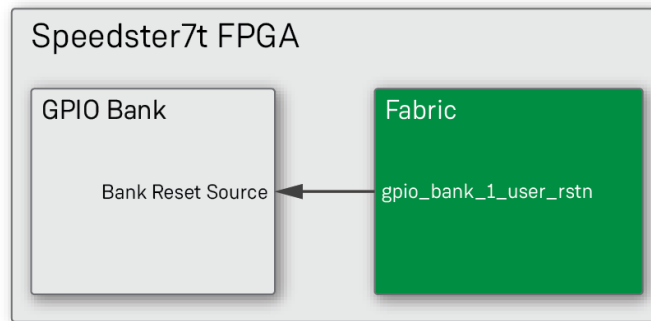
```
input wire      gpio_bank_1_clk_0_parallel
output wire [7:0] gpio_bank_1_data_0
output wire      gpio_bank_1_data_0_oe
```

The SerDes ratio indicates the ratio of the serial clock used to drive the interface and the parallel clock presented to the fabric RTL design. The parallel clock is available to use in the fabric top-level port list and must then be used to drive the SerDes interface. For example, with a SerDes ratio of 4 and a reference clock of 500 MHz, the GPIO bank creates a parallel clock to be used in the fabric RTL of 125 MHz. Notice how in both of the previous examples, the parallel clock is an input presented to the fabric RTL which drives the user SerDes interface. Please see step 8 for how to select a GPIO Bank reference clock. There is a delay between the bank reference clock provided by the user and the parallel clock created by the GPIO bank. The delay between the provided reference clock and the parallel clock is $(2 * \text{serdes_ratio}) + 4$ clock cycles.

6. GPIO has the ability to register signals within the GPIO bank. GPIO data can then be conveniently read or written in the user fabric RTL design. This is referred to this as registered I/O. Select **Rx Register Mode** and/or **Tx Register Mode** if the use of the registered I/O feature is required and consider the following settings:
 - If the box for **Rx Register Mode** is checked, only signals that are inputs are registered. If the box for **Tx Register Mode** is checked, only signals that are outputs are registered. When a signal is used as inout, it is allowed to be registered in either or both directions. If the **Rx Register Mode** box is checked, only the Rx path of the inout signal is registered, and if the **Tx Register Mode** box is checked, only the Tx path of the inout signal is registered. To register both Rx and Tx paths for inout signals, both the **Rx Register Mode** and the **Tx Register Mode** boxes must be checked.
 - Registered I/O applies to data and auxiliary GPIO and does not register clock GPIO.
 - Either box next to **Rx Register Mode** or **Tx Register Mode** should not be checked if planning to use un-registered I/O which is the more conventional form of GPIO.
7. Select the edge type for **Rx Edge Select** and **Tx Edge Select** if using registered I/O, SerDes mode, or DDR mode. Select whether the GPIO bank is preferred to sample data on the positive or negative edge of the reference clock by selecting from the drop-down menu either **Rx Edge Select** or **Tx Edge Select**.
8. Select a **Bank Clock Signal Name** for the GPIO bank. If using clock GPIO, registered I/O, SerDes mode, or DDR mode features, a reference clock is required and is used as the reference for the entire GPIO bank. Select a reference clock from the drop-down menu next to **Bank Clock Signal Name**. A list of valid clock sources is populated automatically in the drop-down menu. It is possible to select a reference clock of two types:
 - Global Clock – this is a clock generated from the output of a general-purpose PLL. This is not to be confused with clocks generated in the user fabric RTL design.
 - Clock GPIO – this is a clock enabled by the GPIO bank itself. When the GPIO clock is enabled in the lower portion of the bank IP configuration editor, it is available to select from the **Bank Clock Signal Name** drop-down menu as the bank reference clock by checking the box in the **Enable** column next to the GPIO clock **I/O Instance Name**. This clock GPIO can be either an input or an output. If using the clock GPIO as an input, provide the clock from an external pad. If using as an output, provide the clock from the fabric RTL design. Refer to step 13 for details about the possible ways to configure clock GPIO. Additionally, when using a GPIO clock as the reference clock it cannot come from another GPIO bank or simple I/O bank.

Ensure that the data GPIO to be registered is synchronized to the reference clock chosen. Additionally, the allowed reference clock frequency range is 0.75 to 500 MHz.

9. If using registered I/O, SerDes mode, or DDR mode features, a **Bank Reset Source** is required. The bank reset source defaults to the internal reset from the FCU. However, three reset source types can be chosen from the following:
- **Global Reset from IOs** – reset sourced from a global clock I/O bank. A clock I/O reset can be created when configuring a clock I/O bank. From the clock I/O bank IP configuration editor, select **Reset** for the **Signal** type column. For additional details about the clock I/O IP configuration, refer to the [Speedster7t Clock and Reset Architecture User Guide \(UG083\)](#).
 - **Internal Reset from FCU** – this is the default option and is a reset sourced from the FCU. The internal reset from the FCU is released automatically after entering user mode. The FPGA enters user mode shortly after powering up.
 - **Local Reset From Core** – local reset from the fabric RTL design. When this source is chosen, an additional output port is added to the user fabric RTL design, sharing the name of the GPIO bank, but with the suffix `_user_rstn`.



120563169-05.2023.03.24

Figure 10: Local Reset From Core as Reset Source Example

10. If **Global Reset from IOs** is selected as the **Bank Reset Source**, the **Bank Global Reset Signal Name** box is enabled to select the required reset name from a clock I/O bank reset source. A list of valid reset sources is populated automatically in the drop-down menu.

Note



GPIO bank DLLs do not use the **Global Reset from IOs** source. GPIO DLL resets can only be dynamically set from CSR registers. Refer to the [Dynamically Configuring Delays with Speedster7t AC7t1500/1550 GPIO \(see page 60\)](#) section for how to use GPIO DLLs and the DLL reset register.

11. Select the **VREF Source**. This option allows sourcing the VREF from an internal hardware macro that uses VDD. Using VDD as the reference is preferred to reduce IR drop. Alternatively, the VREF source can be taken from an external input. There are two VREF source types to choose from:
- **Internal VDD** – using the internal VDD as the reference is preferred to reduce IR drop.
 - **External Bump** – connects to an external input on the board.

The reference voltage source that is selected applies to the entire bank. However, the source can be configured differently for each individual bank if needed.

12. Select the **Bank Voltage Level** to set the voltage level for a given bank. The I/O standards available later in this section depend on the voltage level selected. The bank voltage must be the same for all GPIO banks within a block. Refer to the [Speedster7t GPIO IP Bank Types \(see page 27\)](#) section for details about the available blocks on Speedster7t FPGAs.

In the lower portion of the bank IP configuration editor with rows and columns, select from the additional options that apply to individual signals (I/O instance name). Table cells with a light background are editable, while those with a dark background are read-only. Click in editable table cells to change their values, working from left to right. Altering table cell values often changes whether table cells become editable further to the right (and, when dealing with differential pairs in some cases, the next row down).



Caution!

Working from left to right is important because some options depend on a value selected in a prior column. Working in the opposite direction may override a previously set value.

Similarly, the bank configuration values should be completed before any of the options for the individual data and auxiliary signals are adjusted, because the bank configuration options affect the available configuration options for the individual signals. Going back and changing a bank configuration value may override prior choices made to individual signals in the table (when the values in the table of individual signals are no longer valid for the newly changed bank configuration).

Enable	I/O Instance Name	Placement	Ball/Bump Name	Ball	Differ...	Port ...	Pad C...	I/O Stand...	Vref	Pull T...	SL...	Tx Ta...	Drive...	Tx Ph...
<input type="checkbox"/>	gpio_bank_1_clk_0	GPIO_N_B0_CLK_0	GPIO_N0_BYTE0_BIT_4	AA17										
	gpio_bank_1_clk_1	GPIO_N_B0_CLK_1	GPIO_N0_BYTE0_BIT_5	AA16										
<input type="checkbox"/>	gpio_bank_1_data_0	GPIO_N_B0_DATA_0	GPIO_N0_BYTE0_BIT_0	W17										
<input type="checkbox"/>	gpio_bank_1_data_1	GPIO_N_B0_DATA_1	GPIO_N0_BYTE0_BIT_1	W16										
<input type="checkbox"/>	gpio_bank_1_data_2	GPIO_N_B0_DATA_2	GPIO_N0_BYTE0_BIT_2	Y17										
<input type="checkbox"/>	gpio_bank_1_data_3	GPIO_N_B0_DATA_3	GPIO_N0_BYTE0_BIT_3	Y16										
<input type="checkbox"/>	gpio_bank_1_data_4	GPIO_N_B0_DATA_4	GPIO_N0_BYTE0_BIT_6	AB17										
<input type="checkbox"/>	gpio_bank_1_data_5	GPIO_N_B0_DATA_5	GPIO_N0_BYTE0_BIT_7	AC16										
<input type="checkbox"/>	gpio_bank_1_data_6	GPIO_N_B0_DATA_6	GPIO_N0_BYTE0_BIT_8	AD16										
<input type="checkbox"/>	gpio_bank_1_data_7	GPIO_N_B0_DATA_7	GPIO_N0_BYTE0_BIT_9	AD17										
<input type="checkbox"/>	gpio_bank_1_aux_0	GPIO_N_B0_AUX_0	GPIO_N0_BYTE0_BIT_10	AE16										
<input type="checkbox"/>	gpio_bank_1_aux_1	GPIO_N_B0_AUX_1	GPIO_N0_BYTE0_BIT_11	AE17										

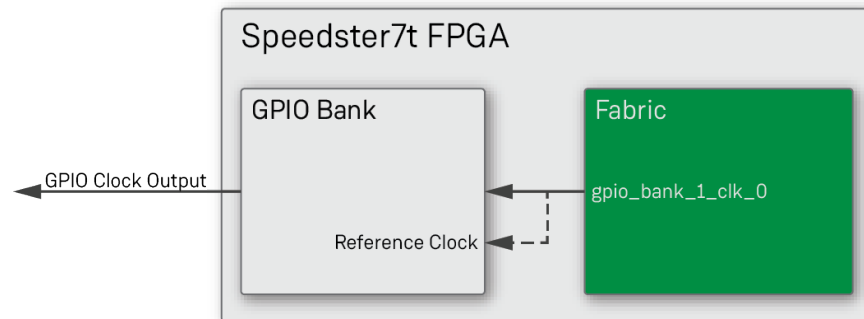
Figure 11: Individual Signal Bank Feature Settings

13. Select the checkbox under the **Enable** column for a given GPIO intended to be enabled. This step should also be repeated for all clock, data, and auxiliary GPIO intended to be used. The capabilities and limitations of clock, data, and auxiliary GPIO are:

Clock GPIO

Use clock GPIO to synchronize user data with external devices or components. In the following images, the dashed line indicates that the GPIO clock output is being utilized as the reference clock for the GPIO bank.

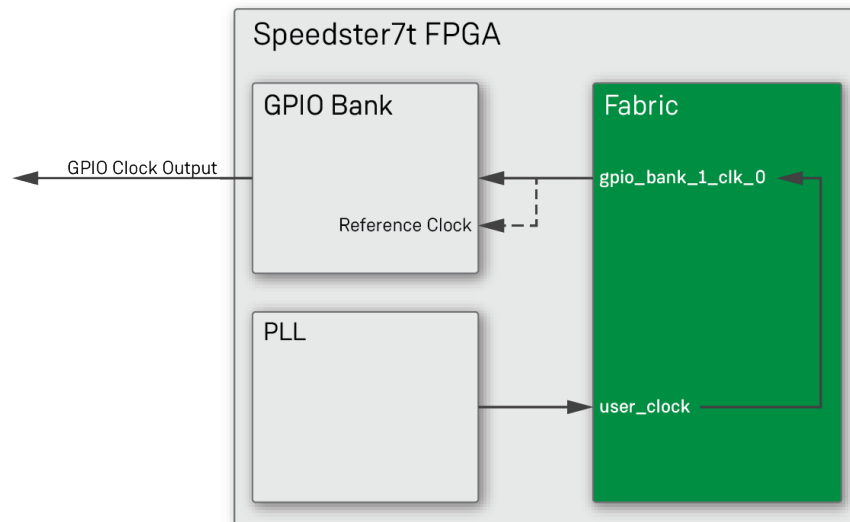
- Capable of being used as an output. Clock GPIO used as an output originates from the fabric RTL design. Voltage standards and other properties within the GPIO bank can be selected for this clock to be utilized with external interfaces.



120563169-01.2023.03.15

Figure 12: GPIO Clock Output Example**Note**

Clock GPIO should not be confused with clock I/O generated by PLLs. While they are not the same, they can be connected to take advantage of the dedicated paths provided by clock I/O. This can be accomplished by assigning the GPIO clock output to the clock I/O in the fabric RTL design.

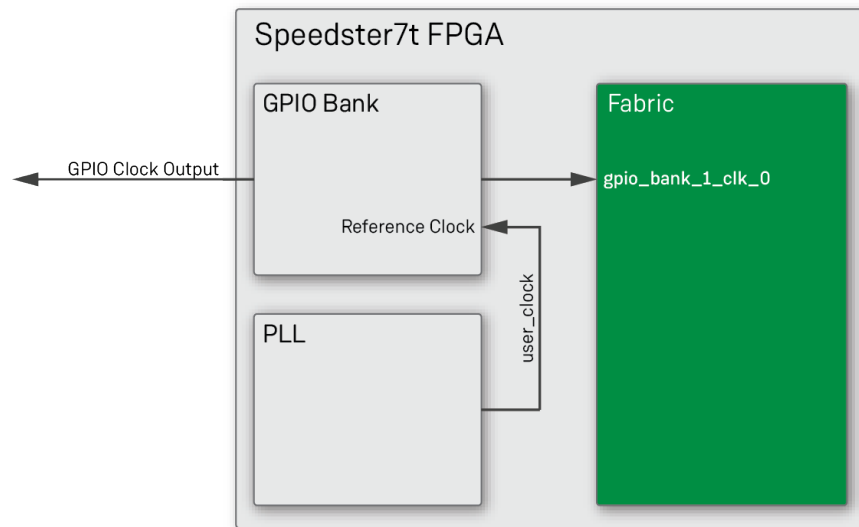


120563169-02.2023.03.15

Figure 13: GPIO Clock Output From a PLL Example**Note**

Using the GPIO clock as the reference clock for the GPIO bank is an available configuration option but is not required.

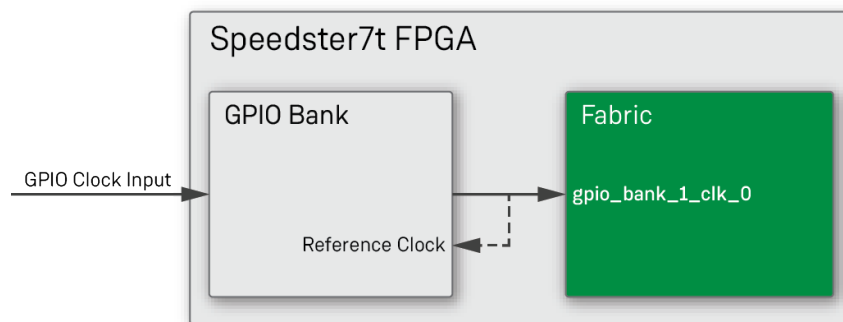
- Capable of being driven by a PLL clock. If a chosen GPIO reference clock (such as `user_clock` in the following image) is driven by a PLL, the clock signal does not go through the fabric, but stays within the I/O ring. This can provide a dedicated clock path the GPIO bank. When the GPIO reference clock is driven by a PLL, the clock signal is driven to both the pad (such as a GPIO clock output) and into the fabric on a GPIO clock input. This can help to simplify the overall clocking scheme of the user design.



120563169-06.2023.03.27

Figure 14: GPIO Bank With PLL Reference Clock Example

- Capable of being used as an input. Clock GPIO used as an input originates from external devices chosen to interface with the Speedster7t FPGA.



120563169-03.2023.03.15

Figure 15: GPIO Clock Input Example

- Capable of being used as the GPIO bank reference clock. When GPIO bank features (e.g., registered I/O, SerDes mode, etc.) are used, a reference clock is required for GPIO banks. A clock GPIO can be utilized as the GPIO bank reference clock, whether as an input or output. After enabling a clock GPIO in the lower section of the GPIO bank IP configuration editor, it can be chosen as the reference clock for the GPIO bank and used to drive its features.

Note

The frequency range allowed for the GPIO bank reference clock is from 0.75 to 500 MHz. Furthermore, GPIO clocks can be incorporated into the fabric RTL design while being used as the reference clock for the GPIO bank.

- Capable of being used as either a differential pair or single-ended. When using a single-ended clock, it is only possible to use the P side. The N side of a clock pair can only be used in differential mode and only to support the P side.
- Not capable of carrying data.

Data GPIO

Use data GPIO for interfacing user data with external devices or components.

- Capable of being used as inputs, outputs, tri-state, or bi-directionally.
- Capable of being used as a differential pair.
- When using data GPIO as outputs (on Speedster7t AC7t1500/1550 FPGAs only), bi-directionally, or tri-state (on Speedster7t AC7t800 FPGAs only), the user design must drive a corresponding output enable signal. The GPIO bank generates this signal for each output, tri-state, or bi-directional GPIO, and makes it available to be controlled in the top-level design. To output data, the output enable should be driven high for data GPIO when necessary. To utilize data GPIO bi-directionally, the output enable should be toggled between low and high when receiving or sending data respectively.

Auxiliary GPIO

Use auxiliary GPIO for interfacing user data with external devices or components. For Speedster7t AC7t1500/1550 FPGAs, auxiliary GPIO has the same features and usage as the data GPIO (future releases of ACE may incorporate functionality for the Speedster7t AC7t800 FPGA that enable auxiliary GPIO to serve as the reset source for the GPIO bank).

- Capable of being used as inputs, outputs, tri-state, or bi-directionally.
- Capable of being used as differential pairs.
- When using data GPIO as outputs (on Speedster7t AC7t1500/1550 FPGAs only), bi-directionally, or tri-state (on Speedster7t AC7t800 FPGAs only), the user design must drive a corresponding output enable signal. The GPIO bank generates this signal for each output, tri-state, or bi-directional GPIO, and makes it available to be controlled in the top-level design. To output data, the output enable should be driven high for data GPIO when necessary. To utilize data GPIO bi-directionally, the output enable should be toggled between low and high when receiving or sending data respectively.

14. The desired **I/O Instance Name** can be selected for the instances that have been enabled. Left-click the box under the **I/O Instance Name** column for the signal requiring the name to be edited.

15. All GPIO pin types (e.g., clock, data, auxiliary) have the ability to be used as a differential pair. Select the **Differential** check box in the options for a given instance. After selecting an instance to be **Differential**, ACE automatically enables its corresponding N signal. GPIO data configured as differential outputs might require the driving both P and N output enable signals depending on the FPGA type. For Speedster7t AC7t1500/1550 FPGAs, the fabric RTL design must provide output enable signals for both the P and N side of a differential pair as well as the single-ended data GPIO. The following is an example of the top-level signals for a data GPIO pin set as differential and configured for output.

```
output wire gpio_bank_1_data_0_out
output wire gpio_bank_1_data_0_n_oe
output wire gpio_bank_1_data_0_p_oe
```

For Speedster7t AC7t800 FPGAs, the fabric RTL does not need to provide an output enable signal when port direction is set as output. The following is an example of the top-level signals for a data GPIO pin set as differential and configured as an output.

```
output wire gpio_bank_1_data_0_out
```

16. Select the signal **Port Direction**.

INPUT

Use this option for GPIO signals that are input externally to the FPGA.

OUTPUT

For Speedster7t AC7t1500/1550 FPGAs, use this option for GPIO signals that are output from the fabric RTL design and require tri-state functionality. When this source is chosen, an additional fabric output port is added to the design, sharing the name of the GPIO signal, but with the suffix `_oe`. This output enable allows tri-stating an output.

For Speedster7t AC7t800 FPGAs, use this option for GPIO signals that are pure outputs from the fabric RTL design and do not require a separate output enable signal. There is no additional output enable signal added and the output cannot be configured as tri-state.

TRISTATE

This option is only available for Speedster7t AC7t800 FPGAs. Use this option for GPIO signals that are output from the fabric RTL design and require tri-state functionality. When this source is chosen, an additional fabric output port is added to the design, sharing the name of the GPIO signal, but with the suffix `_oe`. This output enable allows the RTL design to tri-state the associated output signal.

INOUT (Bi-Directional)

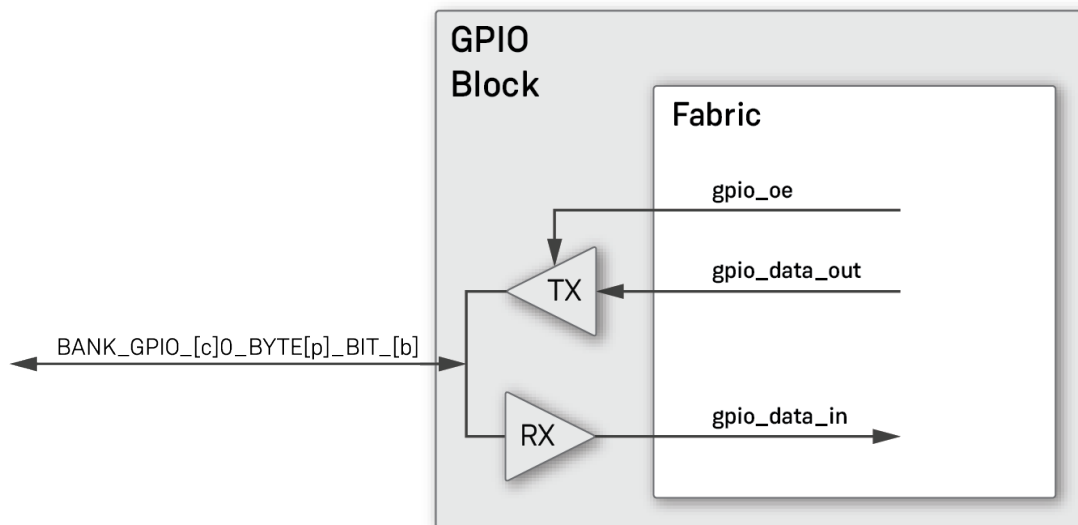
Data and auxiliary GPIO pin types have the ability to be used bi-directionally. When a bi-directional GPIO pin has been enabled, ACE separately creates an input and output signal to use in the design. Use the **I/O Instance Name** with `_in` and `_out` appended to it in the top-level design port list as shown in the following figure.

The following is an example of the top-level signals for a Speedster7t AC7t1500/1550 FPGA with a data GPIO pin set as inout and differential.

```
input  wire gpio_bank_1_data_0_in
output wire gpio_bank_1_data_0_out
output wire gpio_bank_1_data_0_n_oe
output wire gpio_bank_1_data_0_p_oe
```

The following is an example of the top-level signals for a Speedster7t AC7t800 FPGA with a data GPIO pin set as inout and differential.

```
input  wire gpio_bank_1_data_0_in
output wire gpio_bank_1_data_0_out
output wire gpio_bank_1_data_0_oe
```



120563169-04.2023.03.18

Figure 16: Bi-directional GPIO and Output Enable for Speedster7t AC7t800 FPGA

17. Select the signal **I/O Standard**. This option applies an electrical standard for a given GPIO pin. The available electrical standards to choose from are a subset of the bank voltage level selected from the global bank configuration settings. For example, if a global bank setting of 1.35 is selected for the bank voltage level, two options are available in the **I/O Standard** drop-down menu:

- **LVC MOS_135**
- **SSTL135_I**

Refer to the [Supported Electrical Standards \(see page 8\)](#) page for a list of all supported GPIO electrical standards for Speedster7t FPGAs.

18. Select **Pull Type** for the signal. This option determines whether to apply a pull resistor to a given GPIO pin. The available options are:

- **Pullup**
- **Pulldown**
- **None**

19. Select **Slew Rate** for the signal. This option applies a slew rate to a given GPIO pin being used as an output. The available options are:

- **0** (slowest)
- **1**
- **2**
- **3** (fastest)

The fastest rate of 3 is recommended for HSUL, HSTL, and SSTL for $R_{out} = 50$ Ohms, as well as for LVC MOS.

20. Select **Tx Target Impedance** for the signal. This option is the desired termination value in Ohms for a given GPIO pin used as an output. This value is used to improve the signal quality and must match the input impedance of the receiving device. The available termination values depend on the selected I/O standard and the bank voltage level. When not using LVC MOS, it is important to ensure that all I/O have the same termination value set for the Tx target impedance.

21. Select **Tx Phase Shift Factor** for the signal. The Tx phase shift determines the phase shift of the data signal in relation to the bank clock. It can be calculated as $Tx\ Phase\ Shift\ Factor * (Bank\ Clock\ Frequency\ period / 256)$. Setting this to a non-zero value uses the DLL to phase shift the data signal using the bank clock as the master DLL reference clock. Setting **Tx Phase Shift Factor** to zero effectively disables the DLL for a given signal. Tx phase shifts are not allowed when **Advanced Debug Loopback** is enabled. In SerDes mode, the serial clock is used as the bank clock, not the parallel clock. The available **Tx Phase Shift Factor** options are from 0 to 255.

22. Select **Hysteresis Type** for the signal. This option indicates whether a given GPIO pin being used as an input should utilize a Schmidt trigger.

23. Select **ODT Mode** for the signal. This option applies on-die termination to a given GPIO pin. For LVC MOS signals, it is generally recommended to select **Disable ODT Mode**. The available options are:

- **Split-Thevenin**
- **Single_Terminated_ODT_To_VDDH**

24. Select **RX Target Impedance** for the signal. This option is the desired termination value in Ohms for a given GPIO pin used as an input. This value is used to improve the signal quality and must match the output impedance of the sending device. The available termination values depend on the selected I/O Standard and the bank voltage level.

25. Select **Rx Phase Shift Factor** for the signal. The Rx phase shift determines the phase shift of the data signal in relation to the bank clock. It can be calculated as $\text{Rx Phase Shift Factor} * (\text{Bank Clock Frequency period} / 256)$. Setting this to a non-zero value uses the DLL to phase shift the data signal using the bank clock as the master DLL reference clock. Setting **Rx Phase Shift Factor** to zero effectively disables the DLL for a given signal. In SerDes mode, the serial clock is used as the bank clock, not the parallel clock. The available options are from 0 to 255.
26. Ensure that there are no errors or warnings in the **IP Problems** tab. If so, click the problem in question for further details to understand the cause of the issue.
27. Select **Generate**. This step generates all of the necessary IP files including files for other I/O ring configurations configured and saved such as for clock I/O, PLLs, Ethernet, etc. For that reason, it is only required to select **Generate** after all GPIO banks and configurations for a design have been configured.
28. A new window, **Generate IO Ring Design Files**, appears. Set **Directory** to a path in which to store the I/O ring files and then click **Finish**.

Note

Since the fabric RTL design does not include any I/O ring instances, no wrappers or macros need to be generated. Simply connect their top-level ports to the GPIO bank signal names selected in the bank IP configuration editor. The **I/O Instance Name** selected in the bank IP configuration editor should match what is listed in the top-level port list. Refer to the [Design Flow User Guide \(UG106\)](#) for details on using interfaces within the I/O Ring.

Chapter - 10: Configuring a Simple I/O Bank

Introduction

This section describes the steps required to utilize the GPIO within a Speedster7t AC7t800 FPGA simple I/O bank. To begin, the I/O ring files must be generated with ACE by configuring one or more simple I/O banks and generating the I/O ring files:

1. Create an ACE project. Set **Target Device** in the options view after creating the project. This is important as ACE only loads the corresponding IP libraries for the target device selected. Refer to the [ACE User Guide \(UG070\)](#) for additional details about configuring IP and generating I/O ring files.
2. Follow these steps to create a simple I/O bank.

Create a Simple I/O Bank

1. In ACE, navigate to the IP configuration perspective in the top navigation pane.
2. Under **Speedster7t** in the **IP Libraries** pane on the left, navigate to the **IO Ring** section.
3. Double-click **Simple IO Bank**.
4. Select a suitable name for the simple I/O bank instance. The chosen bank name becomes the prefix of the default names of all I/O instances within this bank.

These steps open the simple I/O bank .acxip file to be configured in the bank IP configuration editor:

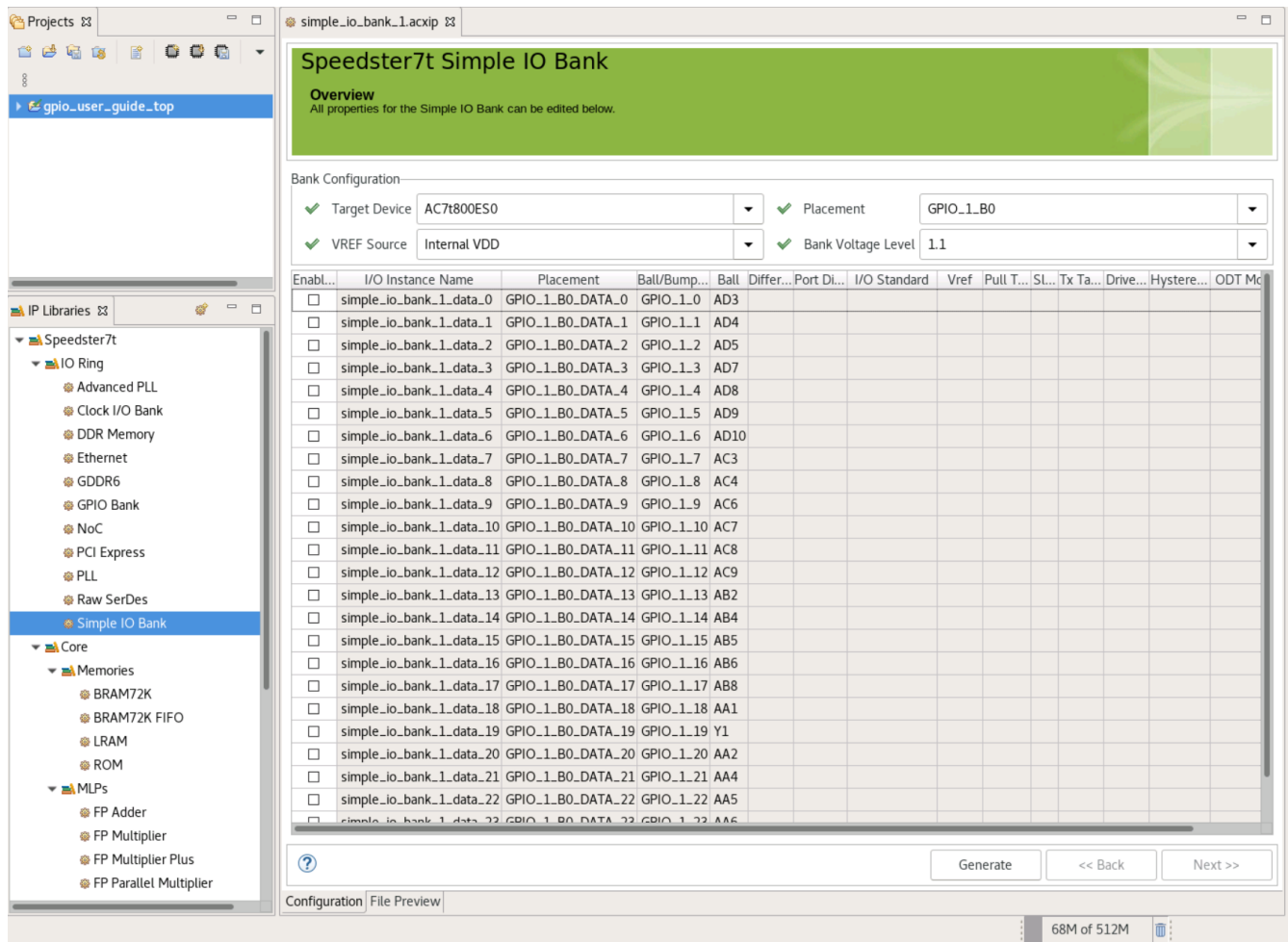


Figure 17: Simple I/O Bank IP Configuration Editor

Creating a Simple I/O Bank IP Configuration

Follow these steps in order. Working from left to right and top to bottom is important because some options depend on previously selected values. Working in the opposite direction might override a previously set value.



Caution!

When enabling any of the following features in the upper portion of the IP configuration editor, that feature is applied to all of the GPIO pins in that bank. It is not possible to apply these features to individual signals.

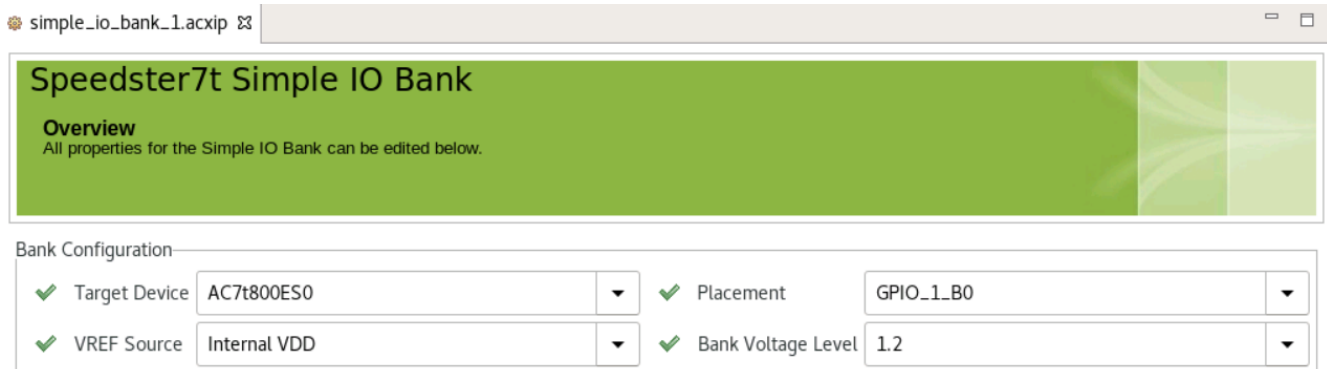


Figure 18: >Global Bank Feature Settings

1. Verify the **Target Device**. This is the Speedster7t FPGA model to be targeted. The target device should have been selected when creating an ACE project in the options view. If the target device selected in the bank IP configuration editor does not match the device for the project, a "Target device mismatch" error appears in the **IP Problems** tab. If there is an error in the IP problems tab due to a target device conflict, click the error in question for further details to understand the cause of the issue.
2. In **Placement**, select the GPIO bank desired. The placement can be one of two available GPIO banks:
 1. GPIO_1_B0
 2. GPIO_2_B0
 Refer to the [Speedster7t GPIO IP Bank Types \(see page 27\)](#) section for specific information about the number of GPIO pins available on these banks.
3. Select **VREF Source**. This option allows sourcing the VREF from an internal hardware macro that uses VDD. Using VDD as the reference is preferred to reduce IR drop. Alternatively, the VREF source can be from an external source. There are two VREF source types to choose from:
 - **Internal VDD** - using the internal VDD as the reference is preferred to reduce IR drop.
 - **External Bump** - connects to an external bump on the board.

The reference voltage source selected applies to the entire bank. However, it can be configured differently for each individual bank if needed.

4. Select **Bank Voltage Level**. this sets the voltage level for a given bank. The I/O standards available later in this section depend on the voltage level selected. The bank voltage must be the same for all GPIO banks within a block. Refer to the [Speedster7t GPIO IP Bank Types \(see page 27\)](#) section for details about the available blocks on Speedster7t FPGAs.

In the lower portion of the bank IP configuration editor with rows and columns, select from the additional options that apply to individual signals (I/O instance name). Table cells with a light background are editable, while table cells with a dark background are read-only. Click in editable table cells to change their values, working from left to right. Altering table cell values often changes whether table cells become editable further to the right (and, when dealing with differential pairs in some cases, the next row down).



Caution!

Working from left to right is important because some options depend on a value selected in a column before it. Working in the opposite direction may override a previously set value.

Similarly, the Bank Configuration values should be completed before any of the options for the individual data and auxiliary signals are adjusted, because the bank configuration options affect the available configuration options for the individual signals. Going back and changing a bank configuration value might override prior choices made to individual signals in the table (when the values in the table of individual signals are no longer valid for the newly changed bank configuration).

Enable	I/O Instance Name	Placement	Ball/Bump...	Ball	Differ...	Port Di...	I/O Standard	Vref	Pull T...	SL...	Tx Ta...	Drive...	Hyste
<input type="checkbox"/>	simple_io_bank_1_data_0	GPIO_1_B0_DATA_0	GPIO_1_0	AD3									
<input type="checkbox"/>	simple_io_bank_1_data_1	GPIO_1_B0_DATA_1	GPIO_1_1	AD4									
<input type="checkbox"/>	simple_io_bank_1_data_2	GPIO_1_B0_DATA_2	GPIO_1_2	AD5									
<input type="checkbox"/>	simple_io_bank_1_data_3	GPIO_1_B0_DATA_3	GPIO_1_3	AD7									
<input type="checkbox"/>	simple_io_bank_1_data_4	GPIO_1_B0_DATA_4	GPIO_1_4	AD8									
<input type="checkbox"/>	simple_io_bank_1_data_5	GPIO_1_B0_DATA_5	GPIO_1_5	AD9									
<input type="checkbox"/>	simple_io_bank_1_data_6	GPIO_1_B0_DATA_6	GPIO_1_6	AD10									
<input type="checkbox"/>	simple_io_bank_1_data_7	GPIO_1_B0_DATA_7	GPIO_1_7	AC3									
<input type="checkbox"/>	simple_io_bank_1_data_8	GPIO_1_B0_DATA_8	GPIO_1_8	AC4									
<input type="checkbox"/>	simple_io_bank_1_data_9	GPIO_1_B0_DATA_9	GPIO_1_9	AC6									
<input type="checkbox"/>	simple_io_bank_1_data_10	GPIO_1_B0_DATA_10	GPIO_1_10	AC7									
<input type="checkbox"/>	simple_io_bank_1_data_11	GPIO_1_B0_DATA_11	GPIO_1_11	AC8									
<input type="checkbox"/>	simple_io_bank_1_data_12	GPIO_1_B0_DATA_12	GPIO_1_12	AC9									
<input type="checkbox"/>	simple_io_bank_1_data_13	GPIO_1_B0_DATA_13	GPIO_1_13	AB2									
<input type="checkbox"/>	simple_io_bank_1_data_14	GPIO_1_B0_DATA_14	GPIO_1_14	AB4									
<input type="checkbox"/>	simple_io_bank_1_data_15	GPIO_1_B0_DATA_15	GPIO_1_15	AB5									
<input type="checkbox"/>	simple_io_bank_1_data_16	GPIO_1_B0_DATA_16	GPIO_1_16	AB6									
<input type="checkbox"/>	simple_io_bank_1_data_17	GPIO_1_B0_DATA_17	GPIO_1_17	AB8									
<input type="checkbox"/>	simple_io_bank_1_data_18	GPIO_1_B0_DATA_18	GPIO_1_18	AA1									

Figure 19: Individual Signal Bank Feature Settings

5. Select the checkbox under the **Enable** column for a given GPIO pin to be enabled. Repeat this step for all GPIO pins to be used.

6. The desired **I/O Instance Name** can be selected for the instances that have been enabled. Left-click the box under the **I/O Instance Name** column for the signal name requiring editing. This can be found in the table in the lower portion of the bank IP configuration editor. After clicking the signal to rename, it is highlighted and ready to edit.

Note

All fields in the lower portion of the bank IP configuration editor with a white background are editable, while fields with a darker background are read-only.

7. Select the **Differential** check box in the options for a given instance to be used as a differential signal.
8. Select the signal **Port Direction**:

INPUT

Use this option for GPIO signals that are input externally to the FPGA.

OUTPUT

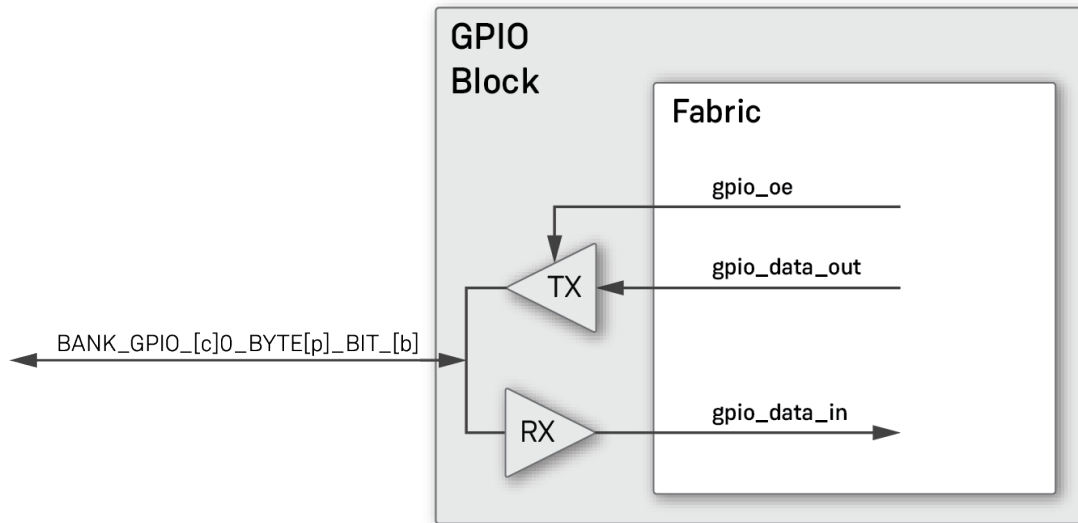
Use this option for GPIO signals that are pure outputs from the fabric RTL design and do not require a separate output enable signal. There is no additional output enable signal added and the output cannot be tri-stated.

TRISTATE

Use this option for GPIO signals that are output from the fabric RTL design and require tri-state functionality. When this source is chosen, an additional fabric output port is added to the design, sharing the name of the GPIO signal, but with the suffix `_oe`. This output enable allows the RTL design to tri-state the associated output signal.

INOUT (Bi-Directional)

Use this option for GPIO signals that are to be used bi-directionally. When a bi-directional GPIO pin has been enabled, ACE separately creates an input and output signal to use in the design. Use the **I/O Instance Name** with `_in` and `_out` appended to it in the top-level design port list as shown in the following figure.



120563169-04.2023.03.18

Figure 20: Bi-directional GPIO and Output Enable for Speedster7t AC7t800 FPGA

9. Select **I/O Standard** for the signal. This option applies an electrical standard for a given GPIO pin. The available electrical standards to choose from are a subset of the bank voltage level selected from the global bank configuration settings. For example, if a global bank setting of 1.35 is selected for the bank voltage level, two options are available in the **I/O Standard** drop-down menu:

- **LVC MOS_135**
- **SSTL135_I**

Refer to the [Supported Electrical Standards \(see page 8\)](#) page for a list of all supported GPIO electrical standards for Speedster7t FPGAs.

10. Select **Pull Type** for the signal. This option determines whether to apply a pull resistor to a given GPIO pin. The available settings are:
 - **Pullup**
 - **Pulldown**
 - **None**

11. Select **Slew Rate** for the signal. This option applies a slew rate to a given GPIO pin being used as an output. The available selections are:

- **0** (slowest)
- **1**
- **2**
- **3** (fastest)

The fastest rate of 3 is recommended for HSUL, HSTL, and SSTL for $R_{out} = 50$ Ohms, as well as for LVCMOS.

12. Select **Tx Target Impedance** for the signal. This option is the desired termination value in Ohms for a given GPIO used as an output. This value is used to improve the signal quality and must match the input impedance of the receiving device. The available termination values depend on the selected I/O standard and the bank voltage level. When not using LVCMOS, it is important to ensure that all I/O pins have the same termination value set for **Tx Target Impedance**.
13. Select **Hysteresis Type** for the signal. This option indicates whether a given GPIO pin being used as an input should utilize a Schmidt trigger.
14. Select **ODT Mode** for the signal. This option applies on-die termination (ODT) to a given GPIO pin. For LVCMOS signals, it is generally recommended to disable ODT. The available option are:
- **Split-Thevenin**
 - **Single_Terminated_ODT_To_VDDH**
15. Select **RX Target Impedance** for the signal. This option is the desired termination value in Ohms for a given GPIO pin used as an input. This value is used to improve the signal quality and must match the output impedance of the sending device. The available termination values depend on the selected I/O standard and the bank voltage level.
16. Ensure that there are no errors or warnings in the **IP Problems** tab. If there are any warnings or errors in the IP problems tab, click the problem in question for further details to understand the cause of the issue.
17. Select **Generate**. This step generates all of the necessary IP files including files for other I/O ring configurations that have been configured and saved (e.g., I/O, PLLs, Ethernet, etc). For that reason, it is only required to select **Generate** after all GPIO banks and configurations for a design have been configured.
18. A new **Generate IO Ring Design Files** window appears. Set **Directory** to a path in which to store the I/O ring files and then choose **Finish**.

Note



Since the fabric RTL design does not include any I/O Ring instances, no wrappers or macros need to be generated. Simply connect the top-level ports to the GPIO bank signal names selected in the bank IP configuration editor. The **I/O Instance Name** selected in the bank IP configuration editor should match what is listed in the top-level port list. Refer to the [Design Flow User Guide \(UG106\)](#) for details on using interfaces within the I/O Ring.

Chapter - 11: Design Implementation and Simulation

Introduction

This section provides information on how to implement and simulate with GPIO on Speedster7t FPGAs. It explains the default GPIO behavior before the FPGA is configured with a bitstream or when a GPIO pin has not been enabled in a GPIO bank. This chapter includes examples of how to implement GPIO in the fabric RTL design using inputs, outputs, bi-directional pins, or with SerDes mode.

A GPIO reference design is available from the knowledge base article, [How do I Download Demonstration and Reference Designs for Speedster7t and Speedcore Devices?](#) The GPIO reference design is a helpful resource for those looking to implement GPIO interfaces in their designs. It provides a tested and verified starting point for implementing GPIO features in a user design.

Default GPIO Behavior

By default, all GPIO pins for both GPIO banks and simple I/O banks are disabled on Speedster7t FPGAs. It is up to the user to enable the pins as needed. When the Speedster7t FPGA is not configured with a bitstream or a given GPIO has not been enabled, device pins are in the high-impedance state. It is worth noting that when using the Speedster7t family of FPGAs, it is not possible to leave an enabled GPIO pin disconnected. This is because the I/O ring requires all ports to be constrained, which means that the input or output pins must at least be connected at the top-level port list. Attempting to leave an enabled GPIO pin disconnected causes ACE to generate the following error message in the Tcl console:

```
Net <Instance Name> on top level Port <Instance Name> does not have a valid pad connection.  
Unable to process this pin. Please contact Achronix customer support.
```

...where <Instance Name> is the name chosen in the GPIO bank for a given signal.

GPIO Port Lists

When a GPIO IP bank is configured and the I/O ring files have been generated, ACE expects the chosen instance names to be present in the top-level port list of the target fabric RTL design. The following section includes examples of how to implement GPIO in the top-level port list when used as inputs, outputs, inout, or SerDes Mode.

Data and Auxiliary GPIO (Inputs)

The following is an example of a top-level port list using all of the data and auxiliary GPIO set as inputs and the clock GPIO set as outputs.

```
module gpio_user_guide_top #(
    // Ports for gpio_bank_1
    input wire      gpio_bank_1_aux_0,
    input wire      gpio_bank_1_aux_1,
    input wire      gpio_bank_1_data_0,
    input wire      gpio_bank_1_data_1,
    input wire      gpio_bank_1_data_2,
    input wire      gpio_bank_1_data_3,
    input wire      gpio_bank_1_data_4,
    input wire      gpio_bank_1_data_5,
    input wire      gpio_bank_1_data_6,
    input wire      gpio_bank_1_data_7,
    output wire     gpio_bank_1_clk_0,
    // Ports for gpio_bank_2
    input wire      gpio_bank_2_aux_0,
    input wire      gpio_bank_2_aux_1,
    input wire      gpio_bank_2_data_0,
    input wire      gpio_bank_2_data_1,
    input wire      gpio_bank_2_data_2,
    input wire      gpio_bank_2_data_3,
    input wire      gpio_bank_2_data_4,
    input wire      gpio_bank_2_data_5,
    input wire      gpio_bank_2_data_6,
    input wire      gpio_bank_2_data_7,
    output wire     gpio_bank_2_clk_0,
    // Ports for gpio_bank_3
    input wire      gpio_bank_3_data_0,
    input wire      gpio_bank_3_data_1,
    input wire      gpio_bank_3_data_2,
    input wire      gpio_bank_3_data_3,
    input wire      gpio_bank_3_data_4,
    input wire      gpio_bank_3_data_5,
    output wire     gpio_bank_3_clk_0,
    // Ports for gpio_bank_4
    input wire      gpio_bank_4_aux_0,
    input wire      gpio_bank_4_aux_1,
    input wire      gpio_bank_4_data_0,
    input wire      gpio_bank_4_data_1,
    input wire      gpio_bank_4_data_2,
    input wire      gpio_bank_4_data_3,
    input wire      gpio_bank_4_data_4,
    input wire      gpio_bank_4_data_5,
    input wire      gpio_bank_4_data_6,
    input wire      gpio_bank_4_data_7,
    output wire     gpio_bank_4_clk_0,
    // Ports for gpio_bank_5
    input wire      gpio_bank_5_aux_0,
    input wire      gpio_bank_5_aux_1,
    input wire      gpio_bank_5_data_0,
    input wire      gpio_bank_5_data_1,
    input wire      gpio_bank_5_data_2,
```

```

input wire      gpio_bank_5_data_3,
input wire      gpio_bank_5_data_4,
input wire      gpio_bank_5_data_5,
input wire      gpio_bank_5_data_6,
input wire      gpio_bank_5_data_7,
output wire     gpio_bank_5_clk_0,
// Ports for gpio_bank_6
input wire      gpio_bank_6_data_0,
input wire      gpio_bank_6_data_1,
input wire      gpio_bank_6_data_2,
input wire      gpio_bank_6_data_3,
input wire      gpio_bank_6_data_4,
input wire      gpio_bank_6_data_5,
output wire     gpio_bank_6_clk_0
);

```

Data and Auxiliary GPIO (Outputs)

The following is an example of a top-level port list that uses the data and auxiliary GPIO set as outputs on GPIO bank 1. Notice that ACE generated an additional signal for each data and auxiliary GPIO pin. This additional signal is an output enable that must drive this signal from the user design in order to function. Setting the output enable to one enables the data to pass through on the corresponding GPIO pin.

An example of using a data GPIO with an output enable would be when implementing an I²C bus. In the case of using the Speedster7t FPGA as an I²C initiator, the user design would toggle the output enable while constantly driving the data GPIO pin to zero. This means that the output enable is used to control the direction of data flow, while the data input is constantly driven with a logic low (0). When the output enable is driven to one, the data GPIO pin drives a zero. When the output enable is driven to zero, the data GPIO pin enters a high impedance state which allows external devices to pull up the data GPIO signal.

```

module gpio_user_guide_top #(
(
// Ports for gpio_bank_1
output wire      gpio_bank_1_aux_0,
output wire      gpio_bank_1_aux_0_oe,
output wire      gpio_bank_1_aux_1,
output wire      gpio_bank_1_aux_1_oe,
output wire      gpio_bank_1_clk_0,
output wire      gpio_bank_1_data_0,
output wire      gpio_bank_1_data_0_oe,
output wire      gpio_bank_1_data_1,
output wire      gpio_bank_1_data_1_oe,
output wire      gpio_bank_1_data_2,
output wire      gpio_bank_1_data_2_oe,
output wire      gpio_bank_1_data_3,
output wire      gpio_bank_1_data_3_oe,
output wire      gpio_bank_1_data_4,
output wire      gpio_bank_1_data_4_oe,
output wire      gpio_bank_1_data_5,
output wire      gpio_bank_1_data_5_oe,
output wire      gpio_bank_1_data_6,
output wire      gpio_bank_1_data_6_oe,
output wire      gpio_bank_1_data_7,
output wire      gpio_bank_1_data_7_oe
);

```


Data and Auxiliary GPIO (Inout/ Bi-Directional)

The following is an example of a top-level port list using the data and auxiliary GPIO pins set as inout on GPIO bank 1. Notice that ACE generated three signals for each data and auxiliary GPIO pin. These three additional signals are:

- an input labeled <I/O Instance Name>_in.
- an output labeled <I/O Instance Name>_out.
- an output enable labeled <I/O Instance Name>_oe. Setting the output enable to one or zero is what determines if the GPIO pin is to be used as an output or an input at that time. Use the output enable in conjunction with the logic that drives the bi-directional data intended for the GPIO pin. The I/O instance name is chosen when configuring the GPIO bank. Refer to the [Configuring a GPIO Bank \(see page 32\)](#) section for how to edit the I/O instance name.

```
module gpio_user_guide_top #()
(
    // Ports for gpio_bank_1
    input wire      gpio_bank_1_aux_0_in,
    input wire      gpio_bank_1_aux_1_in,
    input wire      gpio_bank_1_data_0_in,
    input wire      gpio_bank_1_data_1_in,
    input wire      gpio_bank_1_data_2_in,
    input wire      gpio_bank_1_data_3_in,
    input wire      gpio_bank_1_data_4_in,
    input wire      gpio_bank_1_data_5_in,
    input wire      gpio_bank_1_data_6_in,
    input wire      gpio_bank_1_data_7_in,
    output wire     gpio_bank_1_aux_0_oe,
    output wire     gpio_bank_1_aux_0_out,
    output wire     gpio_bank_1_aux_1_oe,
    output wire     gpio_bank_1_aux_1_out,
    output wire     gpio_bank_1_clk_0,
    output wire     gpio_bank_1_data_0_oe,
    output wire     gpio_bank_1_data_0_out,
    output wire     gpio_bank_1_data_1_oe,
    output wire     gpio_bank_1_data_1_out,
    output wire     gpio_bank_1_data_2_oe,
    output wire     gpio_bank_1_data_2_out,
    output wire     gpio_bank_1_data_3_oe,
    output wire     gpio_bank_1_data_3_out,
    output wire     gpio_bank_1_data_4_oe,
    output wire     gpio_bank_1_data_4_out,
    output wire     gpio_bank_1_data_5_oe,
    output wire     gpio_bank_1_data_5_out,
    output wire     gpio_bank_1_data_6_oe,
    output wire     gpio_bank_1_data_6_out,
    output wire     gpio_bank_1_data_7_oe,
    output wire     gpio_bank_1_data_7_out
);
```

Data GPIO with SerDes Mode (INOUT)

The following is an example of a top-level port list that uses four data GPIO pins set as inout with a SerDes ratio of 4 on GPIO bank 1. Notice that ACE generated a 4-bit bus for each of the I/O instances. Additionally, a new input labeled <I/O Instance Name>_<reference clock>_parallel is created. Where the <reference clock> is chosen when configuring the GPIO bank. Refer to the [Configuring a GPIO Bank \(see page 32\)](#) section for how to choose a reference clock. Use this parallel clock created to drive the SerDes interface in your design.

```
module gpio_user_guide_top #()
(
    // Ports for gpio_bank_1
    input wire      gpio_bank_1_clk_0_parallel,
    input wire [3:0] gpio_bank_1_data_0_in,
    input wire [3:0] gpio_bank_1_data_1_in,
    input wire [3:0] gpio_bank_1_data_2_in,
    input wire [3:0] gpio_bank_1_data_3_in,
    input wire [3:0] gpio_bank_1_data_4_in,
    output wire      gpio_bank_1_clk_0,
    output wire      gpio_bank_1_data_0_oe,
    output wire [3:0] gpio_bank_1_data_0_out,
    output wire      gpio_bank_1_data_1_oe,
    output wire [3:0] gpio_bank_1_data_1_out,
    output wire      gpio_bank_1_data_2_oe,
    output wire [3:0] gpio_bank_1_data_2_out,
    output wire      gpio_bank_1_data_3_oe,
    output wire [3:0] gpio_bank_1_data_3_out,
    output wire      gpio_bank_1_data_4_oe,
    output wire [3:0] gpio_bank_1_data_4_out
);
```

DDR Mode (Inputs)

The following is an example of a top-level port list that uses five data GPIO pins set as inputs with DDR mode enabled on GPIO bank 1. Notice that ACE generated two inputs for each enabled GPIO pin with the _l and _h suffixes.

```
module gpio_user_guide_top #()
(
    // Ports for gpio_bank_1
    input wire      gpio_bank_1_clk_0,
    input wire      gpio_bank_1_data_0_h,
    input wire      gpio_bank_1_data_0_l,
    input wire      gpio_bank_1_data_1_h,
    input wire      gpio_bank_1_data_1_l,
    input wire      gpio_bank_1_data_2_h,
    input wire      gpio_bank_1_data_2_l,
    input wire      gpio_bank_1_data_3_h,
    input wire      gpio_bank_1_data_3_l,
    input wire      gpio_bank_1_data_4_h,
    input wire      gpio_bank_1_data_4_l
);
```

DDR and SerDes Modes (Outputs)

The following is an example of a top-level port list that uses five data GPIO pins set as outputs with DDR mode enabled and a SerDes ratio of 2 on GPIO bank 1. Notice that ACE generated a 2-bit bus for both the low and high enabled DDR mode signals.

```
module gpio_user_guide_top #()
(
    input wire          gpio_bank_1_clk_0_parallel,
    output wire         gpio_bank_1_clk_0,
    output wire [1:0]   gpio_bank_1_data_0_h,
    output wire [1:0]   gpio_bank_1_data_0_l,
    output wire [1:0]   gpio_bank_1_data_1_h,
    output wire [1:0]   gpio_bank_1_data_1_l,
    output wire [1:0]   gpio_bank_1_data_2_h,
    output wire [1:0]   gpio_bank_1_data_2_l,
    output wire [1:0]   gpio_bank_1_data_3_h,
    output wire [1:0]   gpio_bank_1_data_3_l,
    output wire [1:0]   gpio_bank_1_data_4_h,
    output wire [1:0]   gpio_bank_1_data_4_l
);
```

Simulation

Achronix provides various design simulation modes. These modes take into consideration the hardened IP (i.e., GPIO bank) that can be included in the user design and are cycle accurate to the hardware. Simulating with GPIO is only supported with full-chip RTL mode. Full-Chip BFM (bus functional model) or standalone simulation modes do not exist for GPIO as for other Achronix IP. If following the Achronix reference design and simulation project flows, refer to the Reference Design Simulation chapter included in most Achronix reference design documentation for how to use full-chip RTL mode.

Chapter - 12: Dynamically Configuring Delays with Speedster7t AC7t1500/1550 GPIO

Introduction

GPIO delays can be configured dynamically by writing to the appropriate control and status registers (CSR). The CSR space is just one of the Speedster7t FPGA peripherals that can be accessed through the 2D NoC. The CSR space takes up a few of the 42-bit addresses as specified in the 2D NoC global address map. As can be seen in table 3, chapter 6 of the *Speedster7t Network on Chip User Guide (UG089)*, the most significant 8 bits of the CSR address must be "0010_0000".

Writing to the CSR can be accomplished by issuing the API command, `csr_write_named`, in the ACE Tcl console as described in table 5 of the application note, *Runtime Programming of Speedster FPGAs (AN025)*. Modifying CSR values can also be accomplished by utilizing 2D NoC access points (NAPs) within the fabric as described in chapter 7, "Network-on-Chip (NoC) Primitive", of the *Speedster7t Component Library User Guide (UG086)*.

Because the CSR can configure various 2D NoC peripherals, the target ID field points to the registers that configure the GPIO subsystems.

Each GPIO subsystem has three banks. The following table lists which GPIO banks are configured by their respective IP ID bits and the specific target ID values for the various GPIO subsystems on a Speedster7t FPGA.

Each 42-bit CSR address has the following format:

Table 10: CSR Address Bits

CSR Address Bits	Address Type	Possible Values
41:34	2D NoC destination	0010_0000
33:28	Target ID	<ul style="list-style-type: none"> North GPIO subsystem: 01_1101 South GPIO subsystem: 00_1011
27:24	IP ID	<ul style="list-style-type: none"> Bank 1: 0000 Bank 2: 0001 Bank 3: 0010
23:0	Memory address	(See the following table)

Memory Addressing

The least 24 significant bits of the CSR address correspond to the different IOB configuration registers that configure the DLLs.

Table 11: Memory Addresses of GPIO Bank DLL Configuration Registers

Register Name	Register Field	Bit	Type	Reset Value	Description	Address [23:0]
TX_DRIVE_EDGE_SEL	TX_DRIVE_EDGE_SEL	0	R/W	0	Write 1 to enable transmit data to be driven to the pad on the negative edge of the clock.	000020
TX_DLL_ENABLE	TX_DLL_ENABLE	0	R/W	0	Write 1 to enable DLL in the transmit path.	000024
RX_SAMPLE_EDGE_SEL	RX_SAMPLE_EDGE_SEL	0	R/W	0	Write 1 to sample data on the falling edge of the clock in the receive direction.	000028
TX_FLOP_CONTROL	TX_FLOP_ENABLE	0	R/W	0	Write 1 to enable a register in the transmit path.	00002C
	TX_PIPE_BYPASS_ENABLE	1	R/W	0	Write 1 to bypass the pipeline stage in the transmit path.	
RX_DLL_ENABLE	RX_DLL_ENABLE	0	R/W	0	Write 1 to enable a DLL in the receive path.	000030
RX_FLOP_CONTROL	RX_FLOP_ENABLE	0	R/W	0	Write 1 to enable a register in the receive path.	000034
	RX_PIPE_BYPASS_ENABLE	1	R/W	0	Write 1 to bypass the pipeline stage in the receive path.	
DLL_RESET	SOFT_RESET	1	R/W	0	Write: 0 to assert reset to the DLL. 1 to release hard reset to the DLL.	000038
	HARD_RESET	0	R/W	0	Write: 0 to assert reset to the DLL. 1 to release soft reset to the DLL.	
SDLL0_DELAY_REG_1 ⁽¹⁾	SDLL0_DELAY_REG_1	31:0	R/W	0	32-bit value programmed for the delay required from responder DLL 0. The delay value is used by the DLL to set delays for the four channels, IOB pads 1–4: [7:0] – channel 0: core-to-pad data for IOB PAD1 [15:8] – channel 1: core-to-pad data for IOB PAD2 [23:16] – channel 2: core-to-pad data for IOB PAD3 [31:24] – channel 3: core-to-pad data for IOB PAD4	00003C
SDLL0_DELAY_REG_2 ⁽¹⁾	SDLL0_DELAY_REG_2	31:0	R/W	0	32-bit value programmed for the delay required from responder DLL 0. The delay value is used by the DLL to set delays for the four channels, IOB pads 7–10: [7:0] – channel 4: core-to-pad data for IOB PAD7 [15:8] – channel 5: core-to-pad data for IOB PAD8 [23:16] – channel 6: core-to-pad data for IOB PAD9 [31:24] – channel 7: core-to-pad data for IOB PAD10	000040
SDLL1_DELAY_REG_1 ⁽¹⁾	SDLL1_DELAY_REG_1	31:0	R/W	0	32-bit value programmed for the delay required from responder DLL 1. The delay value is used by the DLL to set delays for the four channels, IOB pads 1–4: [7:0] – channel 0: IOB PAD1 OEN [15:8] – channel 1: IOB PAD2 OEN [23:16] – channel 2: IOB PAD3 OEN [31:24] – channel 3: IOB PAD4 OEN	000044

Register Name	Register Field	Bit	Type	Reset Value	Description	Address [23:0]
SDLL1_DELAY_REG_2 ⁽¹⁾	SDLL1_DELAY_REG_2	31:0	R/W	0	32-bit value programmed for the delay required from responder DLL 1. The delay value is used by the DLL to set delays for the four channels, IOB pads 7–10: [7:0] – channel 0: IOB PAD7 OEN [15:8] – channel 1: IOB PAD8 OEN [23:16] – channel 2: IOB PAD9 OEN [31:24] – channel 3: IOB PAD10 OEN	000048
SDLL2_DELAY_REG_1 ⁽¹⁾	SDLL2_DELAY_REG_1	31:0	R/W	0	32-bit value programmed for the delay required from responder DLL 1. The delay value is used by the DLL to set delays for the two channels, IOB pads 11–12: [7:0] – channel 0: IOB PAD11 OEN [15:8] – channel 1: IOB PAD12 OEN [23:16] – channel 2: core-to-pad data for IOB PAD11 [31:24] – channel 3: core-to-pad data for IOB PAD12	00004C
SDLL2_DELAY_REG_2 ⁽¹⁾	SDLL2_DELAY_REG_2	31:0	R/W	0	32-bit value programmed for the delay required from responder DLL 1. The delay value is used by the DLL to set delays for the two channels, IOB pads 11–12: [7:0] – channel 4: pad-to-core data for IOB PAD11 [15:8] – channel 5: pad-to-core data for IOB PAD12 [23:16] – channel 6: Reserved for future use [31:24] – channel 7: Reserved for future use	000050
SDLL3_DELAY_REG_1 ⁽¹⁾	SDLL3_DELAY_REG_1	31:0	R/W	0	32-bit value programmed for the delay required from responder DLL 1. The delay value is used by the DLL to set delays for the four channels, IOB pads 1–4: [7:0] – channel 0: pad-to-core data IOB PAD0 [15:8] – channel 1: pad-to-core data IOB PAD1 [23:16] – channel 2: pad-to-core data IOB PAD2 [31:24] – channel 3: pad-to-core data IOB PAD3	000054
SDLL3_DELAY_REG_2 ⁽¹⁾	SDLL3_DELAY_REG_2	31:0	R/W	0	32-bit value programmed for the delay required from responder DLL 1. The delay value is used by the DLL to set delays for the four channels, IOB pads 7–10: [7:0] – channel 4: pad-to-core data IOB PAD7 [15:8] – channel 5: pad-to-core data IOB PAD8 [23:16] – channel 6: pad-to-core data IOB PAD9 [31:24] – channel 7: pad-to-core data IOB PAD10	000058
MDLL_LOCK_DETECT	LOCK_DETECT	0	R	0	1 indicates that the initiator DLL has completed the delay calibration.	00005C
MDLL_LOCK_CONTROL	LOCK_COUNT	7:0	R/W	0	Number of DLL clock cycles for which lock must be maintained.	000068
	LOCK_THRESHOLD	15:8	R/W	0	Number of DLL clock cycles for which phase difference must be measured.	

Table Notes

- For these registers, the following equation defines the output delay:

$$\text{output delay} = \text{unit_cycle} \times 0.4 + i_delay_code / 256 \times \text{unit_cycle}$$
 where the unit cycle is one clock cycle.

Configuring the GPIO Bank

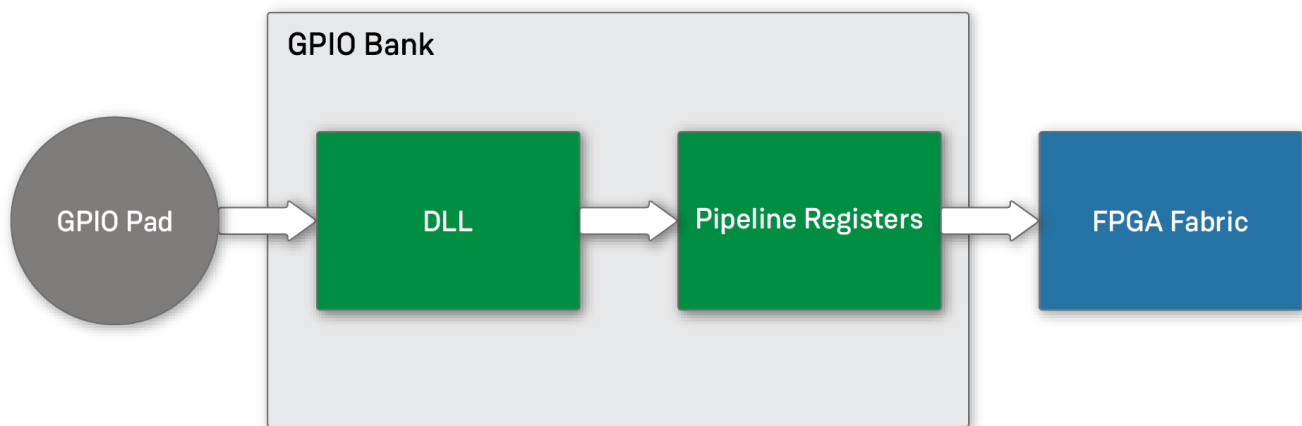
In order to make use of the DLL delays, a reference clock between 300 and 500 MHz must be applied to the GPIO bank.

An asynchronous active-low reset must be asserted through the register `DLL_RESET`. After a delay code value is written to the appropriate SDLL delay register, it takes up to 50 reference clock cycles for the DLL to lock, applying the appropriate delays to the I/O pins. To apply the delays on the receive path, the least significant bit of the value stored in register `RX_DLL_ENABLE` must be set to 1. To apply the delays on the transmit path, the least significant bit of the value stored in register `TX_DLL_ENABLE` must be set to 1.

Pipeline Registers

Receive Path

The following figure illustrates the path of the data inputs from the GPIO pad to the FPGA fabric.



120563164-01.2022.12.16

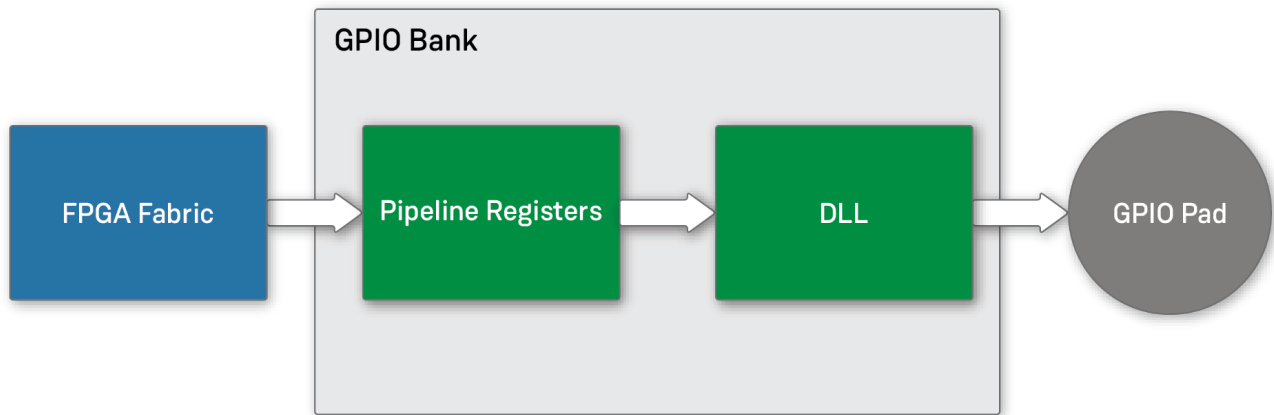
Figure 21: Data Path for GPIO Inputs on the Receive Path

As in the figure, the delayed data is registered by the pipeline flops in the GPIO bank. The configured delay is applied to the receive signal. When registering the signal in the GPIO bank, it is observed at zero to one clock-cycle delay from the fabric.

To observe the receive DLL delays on the fabric as described in the output delay equation, the receive flops must be disabled by writing the value `x00000000` to register `RX_FLOP_CONTROL`.

Transmit Path

The following figure illustrates the path of the data outputs from the FPGA fabric to the GPIO pad.



120563164-02.2022.12.16

Figure 22: Data Path for GPIO Outputs on the Transmit Path

As in the figure, the delay is applied following the registers in the GPIO bank and directly to the device output pads. Therefore, the delayed data as described by the output delay equation can be observed on the output pads regardless of whether the pipeline flops are enabled.

Revision History

Version	Date	Description
1.0	14 Aug 2023	<ul style="list-style-type: none">Initial Achronix release.