PolarFire Family I/O User Guide

Introduction (Ask a Question)

This document describes the features and supported standards for each user I/O type, providing details about I/O banks and I/O naming conventions available in the PolarFire® family. User I/Os support multiple I/O standards while simultaneously providing the high bandwidth needed to maximize the internal logic capabilities of the device and achieve the required system-level performance. They are specifically designed for ease of use and rapid system integration.

The FPGA fabric is common to the PolarFire family, which consists of the following FPGA devices.

PolarFire FPGAs	Microchip's PolarFire® FPGAs are the fifth-generation family of non-volatile FPGA devices, built on state-of-the-art 28 nm non-volatile process technology. PolarFire FPGAs deliver the lowest power at mid-range densities. PolarFire FPGAs lower the cost of mid-range FPGAs by integrating the industry's lowest power FPGA fabric, lowest power 12.7 Gbps transceiver lane, built-in low power dual PCI Express Gen2 (EP/RP), and, on select data security (S) devices, an integrated low-power crypto co-processor.
PolarFire SoC FPGAs	Microchip's PolarFire SoC FPGAs are the fifth-generation family of non-volatile SoC FPGA devices, built on state-of-the-art 28 nm non-volatile process technology. The PolarFire SoC family offers the industry's first RISC-V®-based SoC FPGAs capable of running Linux®. It combines a powerful 64-bit, 5x core RISC-V Microprocessor Subsystem (MSS), based on SiFive's U54-MC family, with the PolarFire FPGA fabric in a single device.
RT PolarFire FPGAs	Microchip's RT PolarFire FPGAs combine our 60 years of spaceflight heritage with the industry's lowest-power PolarFire FPGA family to enable new capabilities for space and mission-critical applications. The RT PolarFire FPGA family includes RTPF500T, RTPF500TL, RTPF500TS, RTPF500TLS, RTPF500ZT, RTPF500ZTL, RTPF500ZTS, and RTPF500ZTLS devices.
RT PolarFire SoC FPGAs	Designed to enable high-performance data processing, our radiation-tolerant PolarFire SoC FPGA is the industry's first embedded, real-time, Linux-capable, RISC-V-based Microprocessor Subsystem (MSS) on the flight-proven RT PolarFire FPGA fabric. With our extensive Mi-V ecosystem, designers can develop lower-power solutions for the challenging thermal environments seen in space. RT PolarFire SoC FPGA family includes RTPFS160ZT, RTPFS160ZTL, RTPFS160ZTS, RTPFS160ZTLS, RTPFS460ZT, RTPFS460ZTL, RTPFS460ZTS, and RTPFS460ZTLS devices.

PolarFire family has two types of user I/Os:

- General-purpose I/O (GPIO)—supports a wide range of I/O standards operating with supplies between 1.2V to 3.3V nominal.
- High-speed I/O (HSIO)—supports I/O standards operating with supplies between 1.2V to 1.8V.

GPIO and HSIO are organized in I/O banks and each I/O bank has dedicated I/O supplies. The unused supplies are connected to ground to reduce noise leakage. In addition to GPIO and HSIO, a number of I/Os are associated with system controller, transceiver clocks, and data pads. These I/Os are powered up independently of other user I/O banks.

The following table summarizes the dedicated I/Os available in the PolarFire family.

Table 1. Dedicated I/Os

Dedicated I/Os	PolarFire FPGA (MPF)	PolarFire SoC FPGA (MPFS)	RT PolarFire FPGA (RTPF)	RT PolarFire SoC FPGA (RTPFS)
GPIO	✓	✓	✓	✓

Table 1. Dedicated I/Os (continued)

Dedicated I/Os	PolarFire FPGA (MPF)	PolarFire SoC FPGA (MPFS)	RT PolarFire FPGA (RTPF)	RT PolarFire SoC FPGA (RTPFS)
HSIO	✓	✓	✓	✓
MSSIO	—	✓	—	✓
MSS-DDR	—	✓	—	✓
MSS-SGMII I/O	—	✓	—	✓

References (Ask a Question)

- For more information about PolarFire FPGA GPIO and HSIO performance specifications, see PolarFire FPGA Datasheet.
- For more information about RT PolarFire FPGA GPIO and HSIO performance specifications, see RT PolarFire FPGA Datasheet.
- For more information about PolarFire SoC FPGA GPIO and HSIO performance specifications, see PolarFire SoC FPGA Datasheet.
- For more information about RT PolarFire SoC FPGA GPIO and HSIO performance specifications, see RT PolarFire SoC Datasheet.
- For more information about PolarFire SoC FPGA MSS, see PolarFire SoC FPGA MSS Technical Reference Manual.
- For information about clock conditioning circuitry (CCC), see PolarFire Family Clocking Resources User Guide.
- For information about transceiver, see PolarFire Family Transceiver User Guide.
- For information about external reference inputs, unused conditions of supply pins, cold sparing, and hot socketing, see respective PolarFire FPGA Board Design User Guide, RT PolarFire FPGA Board Design User Guide, or PolarFire SoC FPGA Board Design Guidelines User Guide.
- For information about programming and I/O states, see PolarFire FPGA and PolarFire SoC FPGA Programming User Guide or RT PolarFire FPGA Programming User Guide.
- For information about MSS Configurator, see Standalone MSS Configurator User Guide for PolarFire SoC.

Table of Contents

1. GPIO and HSIO Features (Ask a Question)

The PolarFire family supports different I/O features for GPIO and HSIO. The following is a summary of I/O features:

1.1. GPIO Features (Ask a Question)

- Supports 1.2V to 3.3V operation
- Single-ended input and output modes
- Flexible supply voltage for certain I/O standards
- Reference, differential, and complementary input receiver modes
- True current-based differential output driver modes and pseudo-differential complementary output modes
- Single-ended static or dynamic termination at 1.8V and 1.5V
- Differential static or dynamic termination of 100Ω
- Cold-sparing and hot swapping (hot plug-in or hot-socket) capabilities
- Process, voltage, and temperature (PVT)-compensated programmable drive strengths
- Supports full and reduced drive for SSTL18 as defined by JEDEC standards
- Built-in weak pull-up, pull-down, and bus-keeper circuits
- Programmable hysteresis
- DDR3 support at up to 1.066 Gbps

Note: For RT PolarFire FPGA, you must familiarize yourself with single event latch-up risks in GPIO banks before selecting 2.5V or 3.3V I/Os. See RT PolarFire Heavy Ion Radiation Single Event Latch-Up Test Report.

1.2. HSIO Features (Ask a Question)

- Supports 1.2V to 1.8V operation
- Single-ended input and output modes
- Mixed single-ended input modes for LVTTL/LVCMOS, regardless of power supply level
- Reference, differential, and complementary input receiver modes
- Pseudo-differential complementary output modes
- Single-ended static or dynamic termination at 1.8V, 1.5V, 1.35V, 1.1V, and 1.2V
- PVT-compensated programmable drive strengths
- Supports full and reduced drives for SSTL18 as defined by JEDEC standards
- Built-in weak pull-up, pull-down, and bus-keeper circuits
- DDR3 and LPDDR3 support up to 1333 Mbps, and DDR4 and LPDDR4 support up to 1.6 Gbps

2. Supported I/O Standards (Ask a Question)

GPIO and HSIO have configurable high-performance I/O drivers and receivers, supporting a wide variety of I/O standards.

The following table lists the I/O standards supported in the receiver and transmitter modes, respectively.

Table 2-1. Supported I/O Standards

I/O Standards	Receiver/Transmitter Modes	VDDI (Nominal) Required	Bank Types	Applications
Single-Ended Standards				
PCI	Receiver, Transmitter	3.3V	GPIO	PC and embedded systems
LVTTL[1]	Receiver	3.3V, 2.5V, 1.8V, 1.5V, 1.2V	GPIO	General purpose
	Transmitter	3.3V		
LVCMOS33[1]	Receiver	3.3V, 2.5V, 1.8V, 1.5V, 1.2V	GPIO	General purpose
	Transmitter	3.3V		
LVCMOS25[1]	Receiver	3.3V, 2.5V, 1.8V, 1.5V, 1.2V	GPIO	General purpose
	Transmitter	2.5V		
LVCMOS18[1]	Receiver	3.3V, 2.5V, 1.8V, 1.5V, 1.2V	GPIO, HSIO	General purpose
	Transmitter	1.8V		
LVCMOS15[1]	Receiver	3.3V, 2.5V, 1.8V, 1.5V, 1.2V	GPIO, HSIO	General purpose
	Transmitter	1.5V		
LVCMOS12[1]	Receiver	3.3V, 2.5V, 1.8V, 1.5V, 1.2V	GPIO, HSIO	General purpose
	Transmitter	1.2V		
SSTL25I, SSTL25II	Receiver	2.5V	GPIO	DDR1[2]
	Transmitter	2.5V	GPIO	DDR1[2]
SSTL18I, SSTL18II	Receiver, Transmitter	1.8V	GPIO, HSIO	DDR2[2]/RLDRAM2[2]
SSTL15I, SSTL15II	Receiver, Transmitter	1.5V	GPIO, HSIO	DDR3
SSTL135I, SSTL135II	Receiver, Transmitter	1.35V	HSIO	DDR3L
HSTL15I, HSTL15II	Receiver, Transmitter	1.5V	GPIO, HSIO	QDRII+
HSTL135I, HSTL135II	Receiver, Transmitter	1.35V	HSIO	RLDRAM3[2]
HSTL12I	Receiver, Transmitter	1.2V	HSIO	QDRII+
HSUL18I, HSUL18II	Receiver, Transmitter	1.8V	GPIO, HSIO	LPDDR[2]
HSUL12I, HSUL12II	Receiver, Transmitter	1.2V	HSIO	LPDDR2[2], LPDDR3
POD12I, POD12II	Receiver, Transmitter	1.2V	HSIO	DDR4

Table 2-1. Supported I/O Standards (continued)

I/O Standards	Receiver/Transmitter Modes	VDDI (Nominal) Required	Bank Types	Applications
LVSTLI LVSTLII	Receiver, Transmitter	1.1V	HSIO	LPDDR4
Differential Standards				
LVDS18G	Receiver	1.8V	GPIO	General purpose
	Transmitter[3]	1.8V	GPIO	General purpose
LVDS33	Receiver	3.3V	GPIO	General purpose
	Transmitter[3]	3.3V	GPIO	General purpose
LVDS25	Receiver	2.5V	GPIO	General purpose
	Transmitter[3]	2.5V	GPIO	General purpose
LVDS18[4,5]	Receiver	1.8V	HSIO	General purpose
RSDS33	Receiver	3.3V	GPIO	General purpose
	Transmitter[3]	3.3V	GPIO	General purpose
RSDS25	Receiver	2.5V	GPIO	General purpose
	Transmitter[3]	2.5V	GPIO	General purpose
RSDS18[5]	Receiver	1.8V	HSIO	General purpose
MINILVDS33	Receiver	3.3V	GPIO	General purpose
	Transmitter[3]	3.3V	GPIO	General purpose
MINILVDS25	Receiver	2.5V	GPIO	General purpose
	Transmitter[3]	2.5V	GPIO	General purpose
MINILVDS18[5]	Receiver	1.8V	HSIO	General purpose
SUBLVDS33	Receiver	3.3V	GPIO	General purpose
	Transmitter[3]	3.3V	GPIO	General purpose
SUBLVDS25	Receiver	2.5V	GPIO	General purpose
	Transmitter[3]	2.5V	GPIO	General purpose
SUBLVDS18[5]	Receiver	1.8V	HSIO	General purpose
PPDS33	Receiver	3.3V	GPIO	General purpose
	Transmitter[3]	3.3V	GPIO	General purpose
PPDS25	Receiver	2.5V	GPIO	General purpose
	Transmitter[3]	2.5V	GPIO	General purpose
PPDS18[5]	Receiver	1.8V	HSIO	General purpose
SLVS33	Receiver	3.3V	GPIO	General purpose
SLVS25	Receiver	2.5V	GPIO	General purpose
SLVS18	Receiver	1.8V	HSIO	General purpose
SLVSE15[6]	Transmitter	1.5V	GPIO, HSIO	General purpose
HCSL33	Receiver	3.3V	GPIO	General purpose
HCSL25	Receiver	2.5V	GPIO	General purpose
HCSL18	Receiver	1.8V	HSIO	General purpose
BUSLVDSE25[6]	Transmitter	2.5V	GPIO	Multipoint backplane applications
MLVDSE25[6]	Transmitter	2.5V	GPIO	Multipoint backplane applications
LVPECL33	Receiver	3.3V	GPIO	Video graphics and clock distribution
LVPECLE33[6]	Transmitter	3.3V	GPIO	Video graphics and clock distribution
MIPI25	Receiver	2.5V	GPIO	Consumer mobile applications
MIPIE25[6]	Transmitter	2.5V	GPIO	Consumer mobile applications, High–speed Mode

Notes:

1. Certain I/O standards are designed to support flexible VDDI assignment. See Mixed I/O in VDDI Banks.
2. This application is supported by the I/O Standard, however, PolarFire family offering does not include the specific memory controller solution.
3. Buffers configured for these standards are true-differential transmitters that do not support bidirectional operations.
4. For HSIO, native LVDS inputs are supported with a single external-differential termination 100Ω resistor, and LVDS transmit outputs are not supported in HSIO banks.
5. These standards require an external voltage reference (VREF) and two single-ended drivers with biasing through external resistors.
6. Buffers are configured as emulated-differential transmitters and also support bidirectional operations. However, they require an external board termination.

2.1. I/O Standard Descriptions (Ask a Question)

This section provides an overview for each of the I/O standards.

2.1.1. 3.3V Peripheral Component Interface (PCI) (Ask a Question)

GPIO supports the PCI I/O standards. The PCI standard uses an LVTTL input buffer and a push-pull output buffer. This standard is used for both 33 MHz and 66 MHz PCI bus applications.

2.1.2. Low-Voltage TTL (LVTTL) (Ask a Question)

LVTTL is a general-purpose standard (EIA/JESD8-B) for 3.3V applications. It uses an LVTTL input buffer and a push-pull output buffer. GPIO supports the LVTTL I/O standards, and the LVTTL output buffer can have up to six different programmable drive strengths. For more information about programmable drive strength control, see Table 7-6.

2.1.3. Low-Voltage CMOS (LVCMOS) (Ask a Question)

LVCMOS is a general-purpose standard implemented in CMOS transistors. Five different LVCMOS operational modes supported are:

- LVCMOS33—an extension of the LVCMOS standard (JESD8-B-compliant) is used for general-purpose 3.3V applications.
- LVCMOS25—an extension of the LVCMOS standard (JESD8-5-compliant) is used for general-purpose 2.5V applications.
- LVCMOS18—an extension of the LVCMOS standard (JESD8-7-compliant) is used for general-purpose 1.8V applications.
- LVCMOS15—an extension of the LVCMOS standard (JESD8-11-compliant) is used for general-purpose 1.5V applications.
- LVCMOS12—an extension of the LVCMOS standard (JESD8-26-compliant) is used for general-purpose 1.2V applications.

2.1.4. Stub Series Terminated Logic (SSTL) (Ask a Question)

SSTL is a general-purpose memory bus standard. Following are the SSTL operational modes supported:

- SSTL25I—SSTL Class I-standard with VDDI (nominal) = 2.5V
- SSTL25II—SSTL Class II-standard with VDDI (nominal) = 2.5V
- SSTL18I—SSTL Class I-standard with VDDI (nominal) = 1.8V
- SSTL18II—SSTL Class II-standard with VDDI (nominal) = 1.8V

- SSTL15I—SSTL Class I-standard with VDDI (nominal) = 1.5V
- SSTL15II—SSTL Class II-standard with VDDI (nominal) = 1.5V
- SSTL135I—SSTL Class I-standard with VDDI (nominal) = 1.35V
- SSTL135II—SSTL Class II-standard with VDDI (nominal) = 1.35V

SSTL25 is defined by the JEDEC standard, JESD8-9B, and used for DDR SDRAM and DDR1 memory interfaces. SSTL18 is defined by the JEDEC standard, JESD8, and used for DDR2 SDRAM memory interfaces. SSTL15 is used for DDR3 memory interfaces; SSTL135 is used for DDR3L memory interfaces.

For more information about signal levels for the various SSTL I/O standards, see PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet.

2.1.5. High-Speed Transceiver Logic (HSTL) (Ask a Question)

HSTL is a general-purpose, high-speed bus standard (EIA/JESD8-6) with a signaling range between 0V and 1.5V, and signals can either be single-ended or differential. This standard is used in memory bus interfaces with data switching capabilities of up to 1.267 GHz.

Following are the HSTL operational modes supported:

- HSTL15I—HSTL Class I-standard with VDDI (nominal) = 1.5V
- HSTL15II—HSTL Class II-standard with VDDI (nominal) = 1.5V
- HSTL135I—HSTL Class I-standard with VDDI (nominal) = 1.35V
- HSTL135II—HSTL Class II-standard with VDDI (nominal) = 1.35V
- HSTL12I—HSTL Class I-standard with VDDI (nominal) = 1.2V
- HSTL12I—HSTL Class II-standard with VDDI (nominal) = 1.2V

For more information about signal levels for the various HSTL I/O standards, see Table 2-1. Also, see PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet.

Important: HSTL135 and HSTL12 are not part of the JEDEC specification. They are, scaled from HSTL15. For more information about HSTL signal levels, see respective PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet.

2.1.6. High-Speed Unterminated Logic (HSUL) (Ask a Question)

HSUL, as specified by the JEDEC standard JESD8-22, is a standard for LPDDR2 and LPDDR3 memory buses. HSUL I/O standards are supported in both HSIO and GPIO.

2.1.7. Pseudo Open Drain (POD) (Ask a Question)

POD standards are intended for DDR4, DDR4L, and LLDRAM3 applications. HSIO supports both POD receive and transmit modes.

2.1.8. Low-Voltage Differential Signal (LVDS) (Ask a Question)

LVDS (ANSI/TIA/EIA-644) is a high-speed, differential I/O standard. The voltage swing between two signal lines is approximately 350 mV. GPIO supports LVDS receive and transmit modes. HSIO supports LVDS receive mode with an external 100Ω board termination, see I/O External Termination for more information.

2.1.9. Reduced-Swing Differential Signal (RSDS) (Ask a Question)

RSDS is similar to an LVDS high-speed interface using differential signaling, but with a smaller voltage swing and requiring a parallel termination resistor. RSDS is only intended for point-to-point applications. For more information about RSDS Voltage Swing, see the respective PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet.

While GPIO supports RSDS receive and trasmit modes, HSIO supports RS,DS receive mode with an external 100Ω on-board termination.

2.1.10. Mini-LVDS (Ask a Question)

Mini-LVDS is a unidirectional interface from the timing controller to the column drivers in TFT LCD displays, and is specified in Texas Instruments standard, SLDA007A. GPIO supports mini-LVDS in both receive and transmit modes. HSIO supports mini-LVDS only in the receive mode and requires an external resistor.

2.1.11. Sub-LVDS (Ask a Question)

Sub-LVDS is a differential low-voltage standard that is a subset of LVDS, and uses a reduced-voltage swing and lower common-mode voltage compared to LVDS. For sub-LVDS, the maximum differential swing is less than 350 mV, as in LVDS. For more information on the maximum differential swing values, see the respective device datasheet. The nominal common-mode voltage for sub-LVDS is 0.9V, while it is 1.25V for LVDS. GPIO supports sub-LVDS in both receive and transmit modes. HSIO supports sub-LVDS only in the receive mode and requires an external resistor. The common mode voltage and differential voltage swing are key differences between Sub-LVDS and LVDS. PolarFire/PolarFire SoC to PolarFire/PolarFire SoC, RT PolarFire to RT PolarFire, and RT PolarFire SoC to RT PolarFire SoC interfaces allow LVDS to sub-LVDS and sub-LVDS to LVDS as the data sheet specifications permit these levels to operate within their specified ranges.

2.1.12. Point-to-Point Differential Signaling (PPDS) (Ask a Question)

PPDS is the next generation of the RSDS standards introduced by National Semiconductor Corporation, and is used to interface to next-generation LCD row and column drivers. PPDS inputs require a parallel termination resistor.

GPIO supports PPDS in both receive and transmit modes. HSIO supports PPDS only in receive mode and requires an external resistor.

2.1.13. Scalable Low-Voltage Signaling (SLVS) (Ask a Question)

SLVS is a chip-to-chip signaling standard designed for maximum performance with minimum power consumption, inheriting low noise susceptibility from LVDS. The standard features a scaled-down 400 mV signal swing, versus the 700 mV swing of LVDS, and includes a ground reference. GPIO and HSIO banks support the SLVS I/O standards, but an external resistor is required for transmitter mode. For more information, see Implementing Emulated Standards for Outputs.

2.1.14. High-Speed Current Steering Logic (HCSL) (Ask a Question)

HCSL is a differential output standard used in PCI Express applications. Both GPIO and HSIO support the HCSL I/O standards (receive-only mode). Although, the common mode range for this standard is from 250 mV to 550 mV, HCSL I/O receivers support a wider range of 50 mV to 2.4V.

2.1.15. Bus-LVDS (B-LVDS)/Multipoint LVDS (M-LVDS) (Ask a Question)

B-LVDS refers to bus interface circuits based on the LVDS technology with the M-LVDS specification extending the LVDS standard to high-performance multipoint bus applications. Multidrop and multipoint bus configurations may contain any combination of drivers, receivers, and transceivers. LVDS drivers provide the higher drive current required by B-LVDS and M-LVDS to accommodate bus loading. These drivers require series terminations for better signal quality and voltage swing control.

The drivers can be located anywhere on the bus, and therefore, termination is also required at both ends of the bus.

GPIO supports B-LVDS and M-LVDS in receive mode. For transmit mode, however, external board termination is required. For more information about various BLVDS standards, see Bus-LVDS Emulated (BLVDSE25) Output Mode and Multipoint Low-Voltage Emulated (MLVDSE25) Output Mode.

2.1.16. Low-Voltage Positive Emitter-Coupled Logic (LVPECL) (Ask a Question)

LVPECL is a 3.3V differential signal standard that transmits one data bit over a pair of signal lines, thus requiring two pins per input or output. The voltage swing between the two signal lines is approximately 850 mV. While LVPECL input is supported for GPIO, external board termination is required for the LVPECL outputs. For more information about LVPECL33, see LVPECL Emulated (LVPECLE33) Output Mode.

2.1.17. Mobile Industry Processor Interface (MIPI) D-PHY (Ask a Question)

MIPI is a serial communication interface used in camera and display applications. GPIO bank supports implementation of the MIPI D-PHY standards using an external termination. For more information, see Implementing MIPI D-PHY.

3. PolarFire Family I/O Banks (Ask a Question)

PolarFire SoC and RT PolarFire SoC FPGA devices have eight user I/O banks, whereas PolarFire and RT PolarFire FPGA devices have five, six, or eight user I/O banks depending upon the device size. In PolarFire SoC and RT PolarFire SoC FPGA devices, the banks assigned to the MSS are only for dedicated use with the MSS block and are not accessible to the FPGA fabric. MSS_DDR, MSS_IO, MSS_SGMII, and MSS_REFCLK are only used with MSS block. For information about MSS bank, see PolarFire SoC FPGA Packaging and Pin Descriptions User Guide.

The I/O banks on the north side of the device support only HSIO. Each I/O bank has dedicated I/O supplies and grounds. Each I/O within a given bank shares the same VDDI power supply and VREF reference voltage. Only compatible I/O standards can be assigned to a given I/O bank.

Each bank contains a bank power detector and a bank receiver reference voltage generator to create an internally generated reference voltage, VREF. Each bank also interfaces with a PVT controller to calibrate the I/O buffer output driver strengths and termination values (needed only for certain I/O standards). The PVT controller generates a set of codes to control the source driver and the sink driver, and also calibrates the HSIO output slew. Each I/O buffer has individual drive-strength programmability to multiply the PVT digital code value by a drive setting to create the desired drive, impedance, or termination settings. For more information, see I/O Analog (IOA) Buffer Programmable Features.

Figure 3-1 through Figure 3-5 show simplified floor plans for each device, including the bank locations. These figures also show the corner block and transceiver block. The corner block includes CCCs, two PLLs, and two DLLs each, providing flexible clock management and synthesis for the FPGA fabric, external system, and I/Os. All banks are not available in all devices, see I/O Lanes in Each Bank for more information. For more information about CCC and transceivers, see PolarFire Family Clocking Resources User Guide and PolarFire Family Transceiver User Guide.

Figure 3-1. PolarFire FPGA MPF300T/MPF300XT/MPF500T and RT PolarFire FPGA RTPF500T I/O Banks

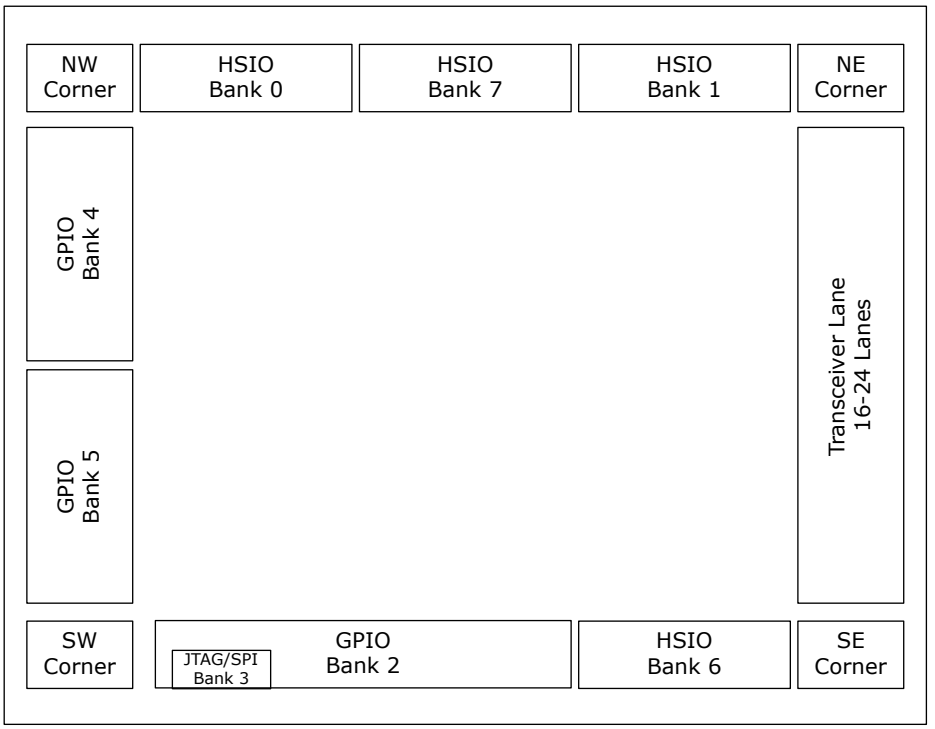


Figure 3-2. PolarFire FPGA MPF200T Device I/O Banks

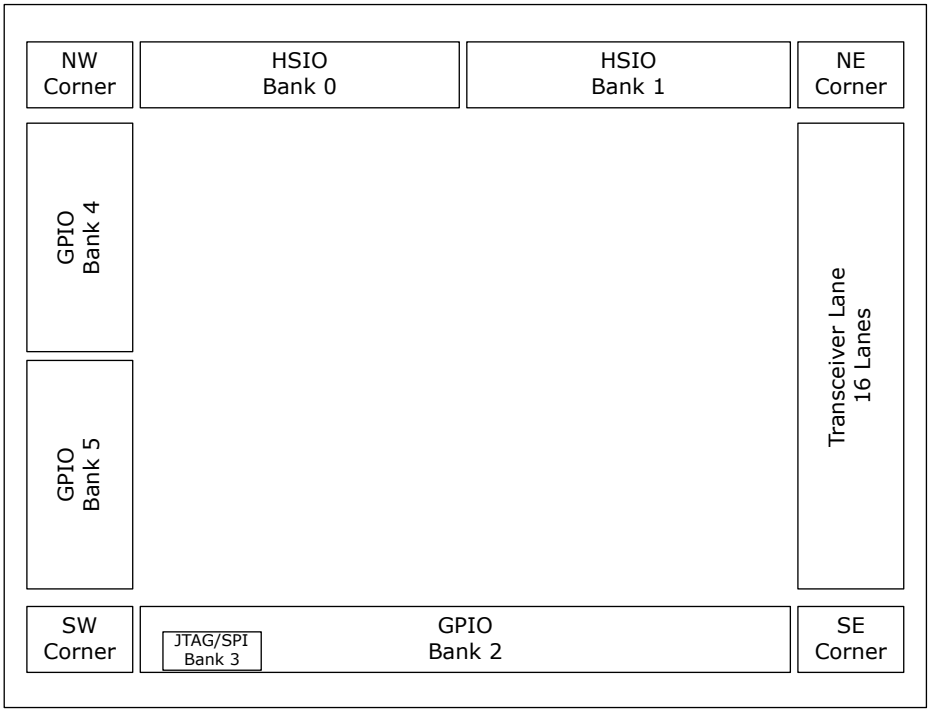


Figure 3-3. PolarFire FPGA MPF050/MPF100T Device I/O Banks

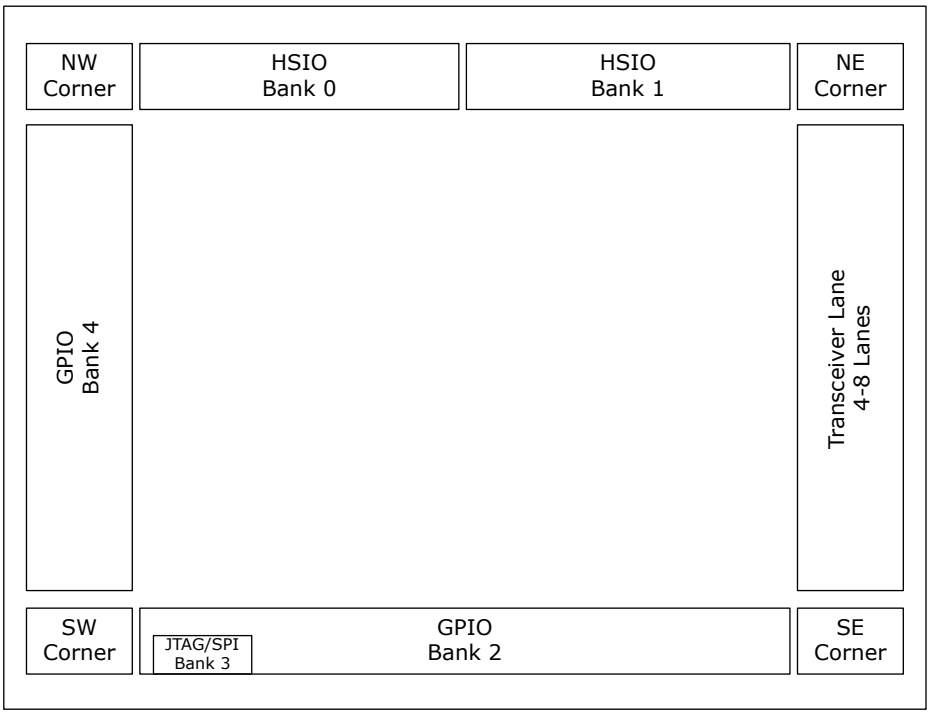


Figure 3-4. PolarFire SoC FPGA MPFS250–FCG1152 I/O Banks

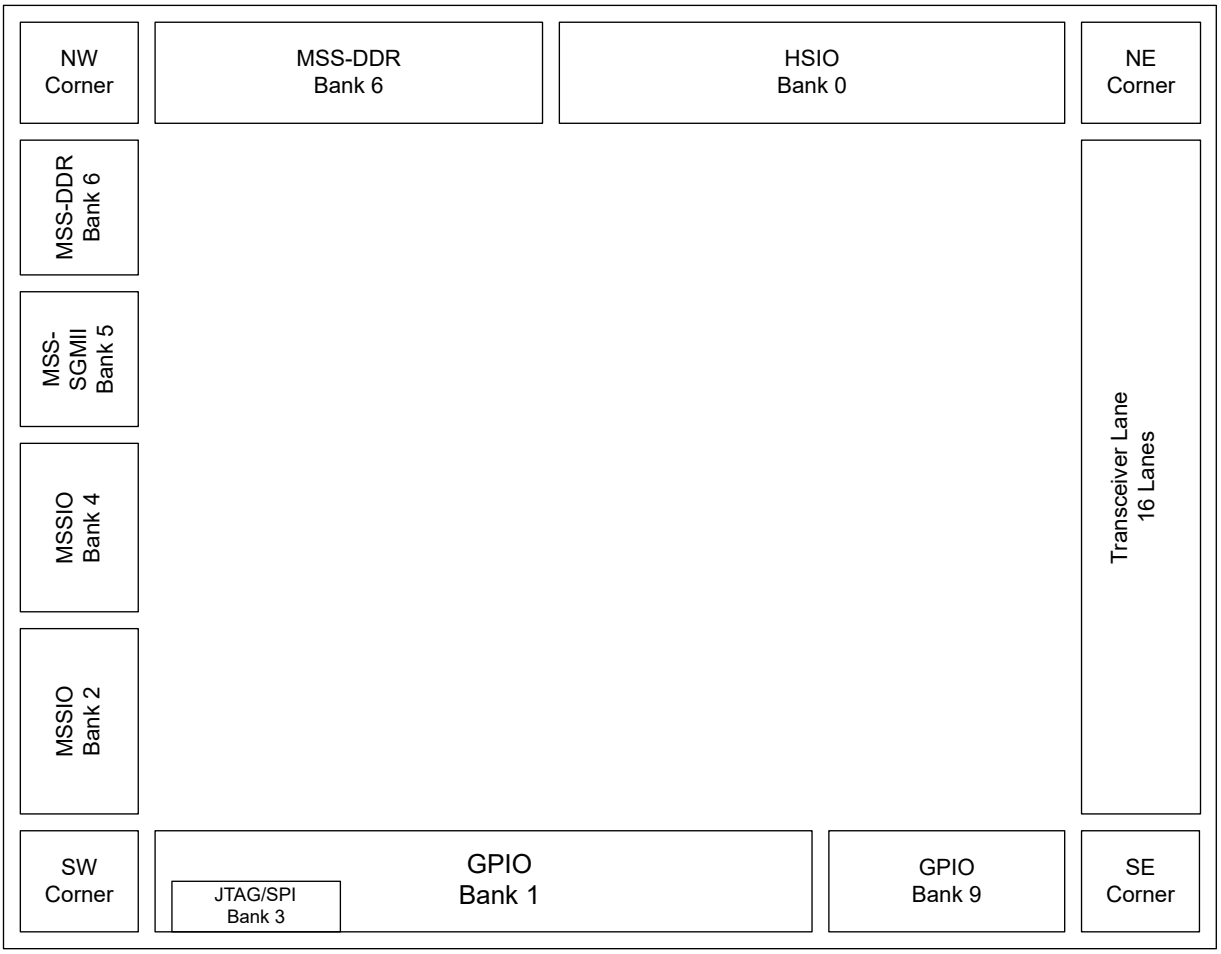


Figure 3-5. PolarFire SoC FPGA MPFS250–FCVG484 I/O Banks

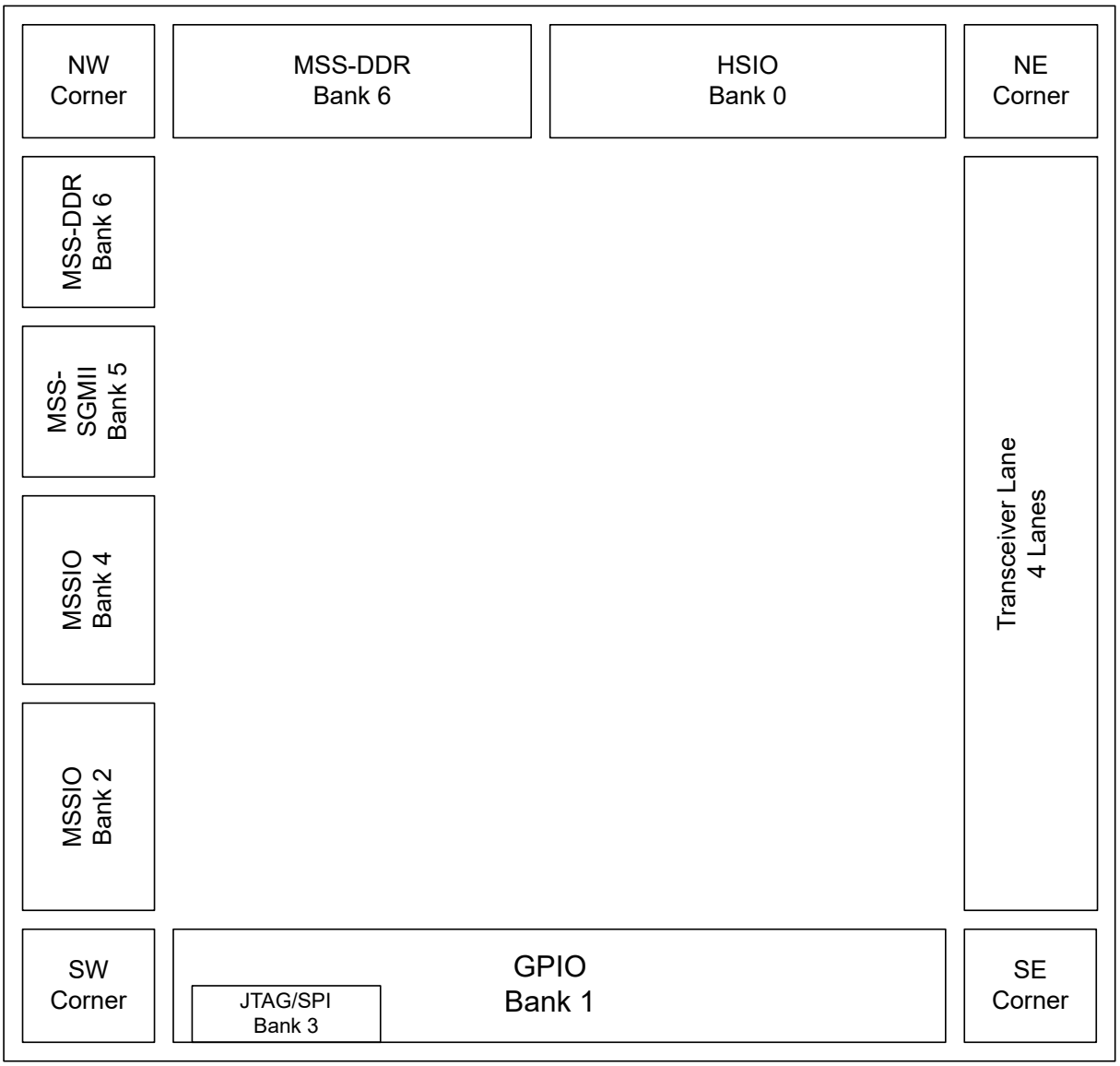


Figure 3-6. PolarFire SoC FPGA MPFS250/MPFS460 and RT PolarFire SoC FPGA RTPFS460 I/O Banks

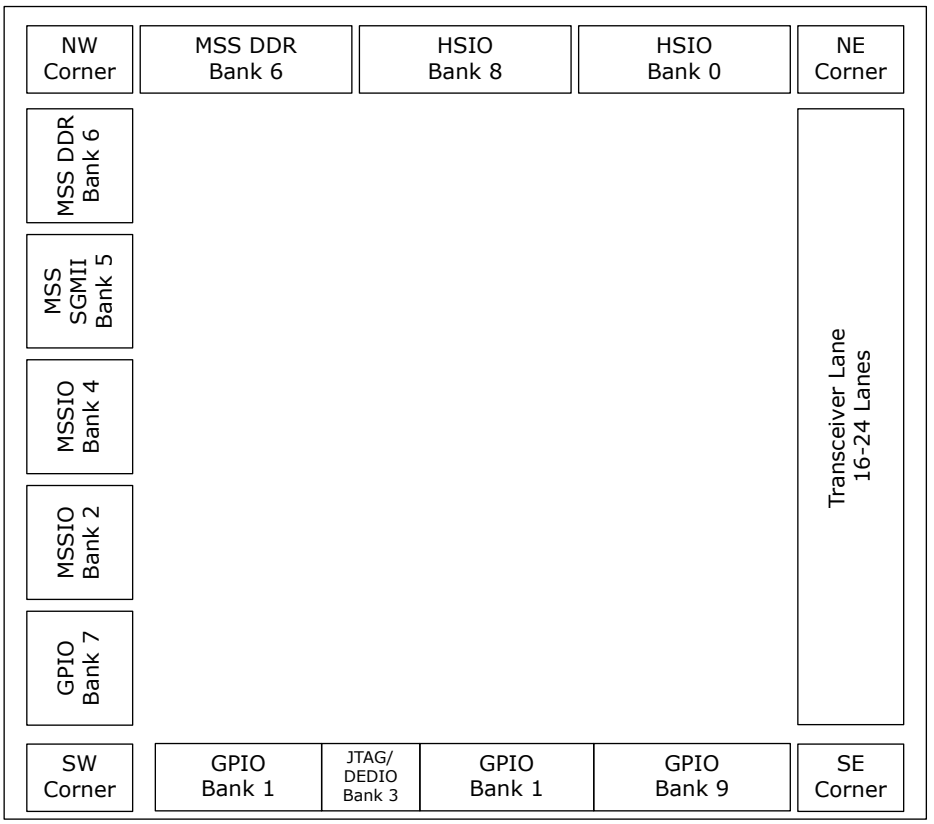


Figure 3-7. PolarFire SoC FPGA MPFS160 and RT PolarFire SoC FPGA RTPFS160 I/O Banks

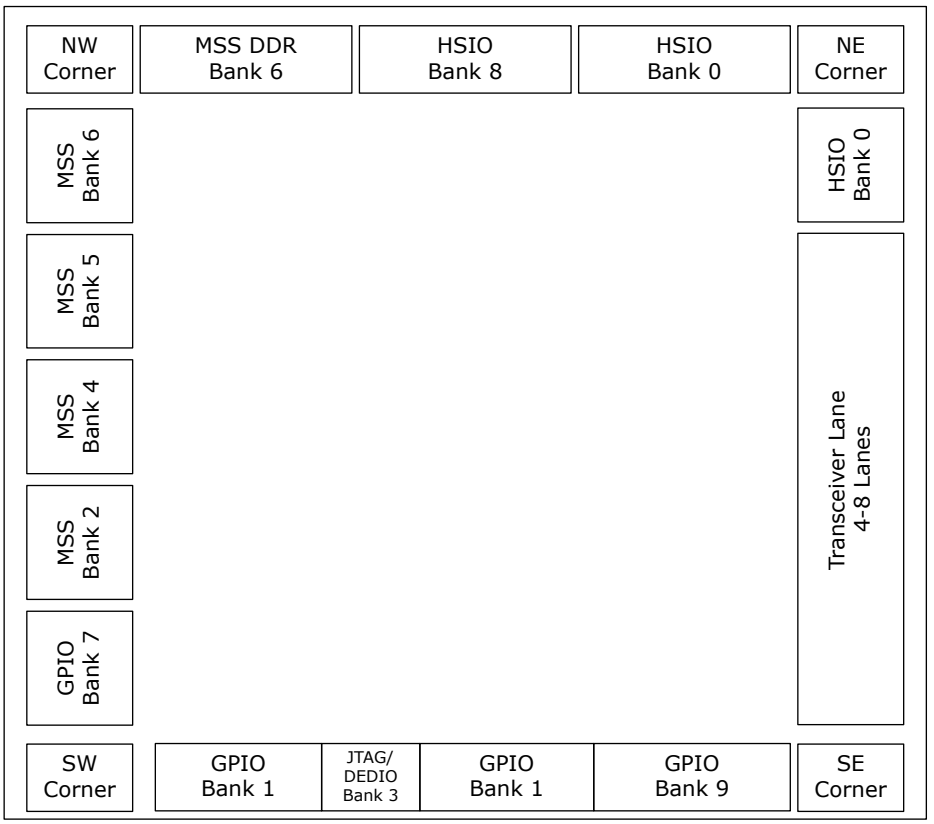


Figure 3-8. PolarFire SoC FPGA MPFS095 I/O Banks

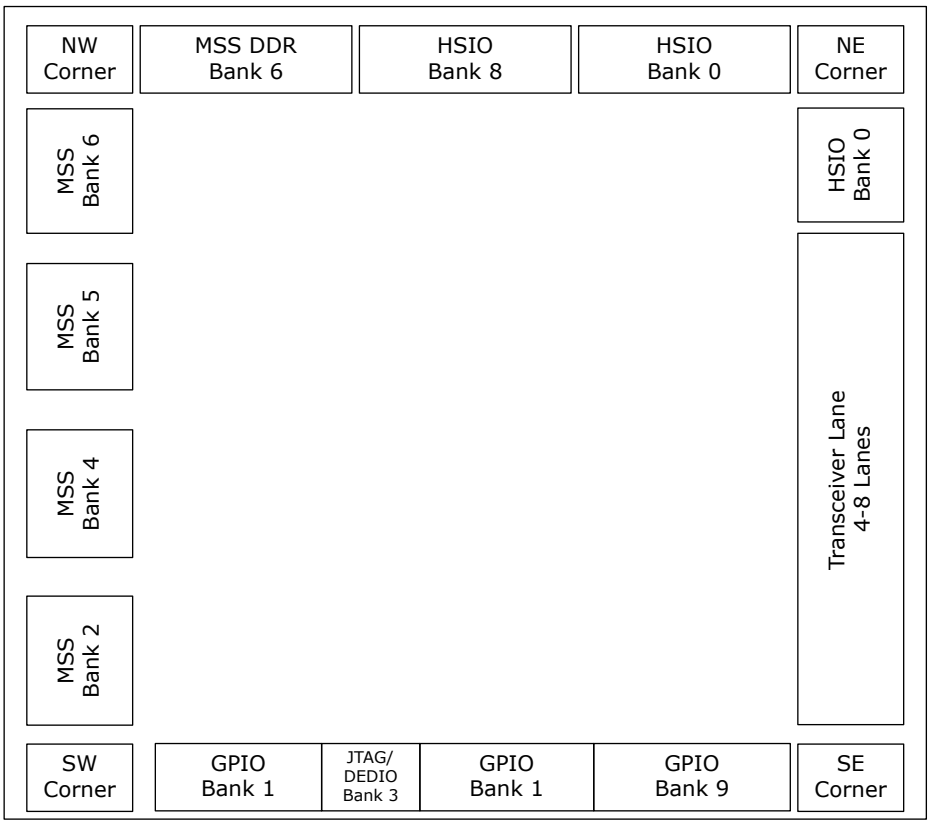
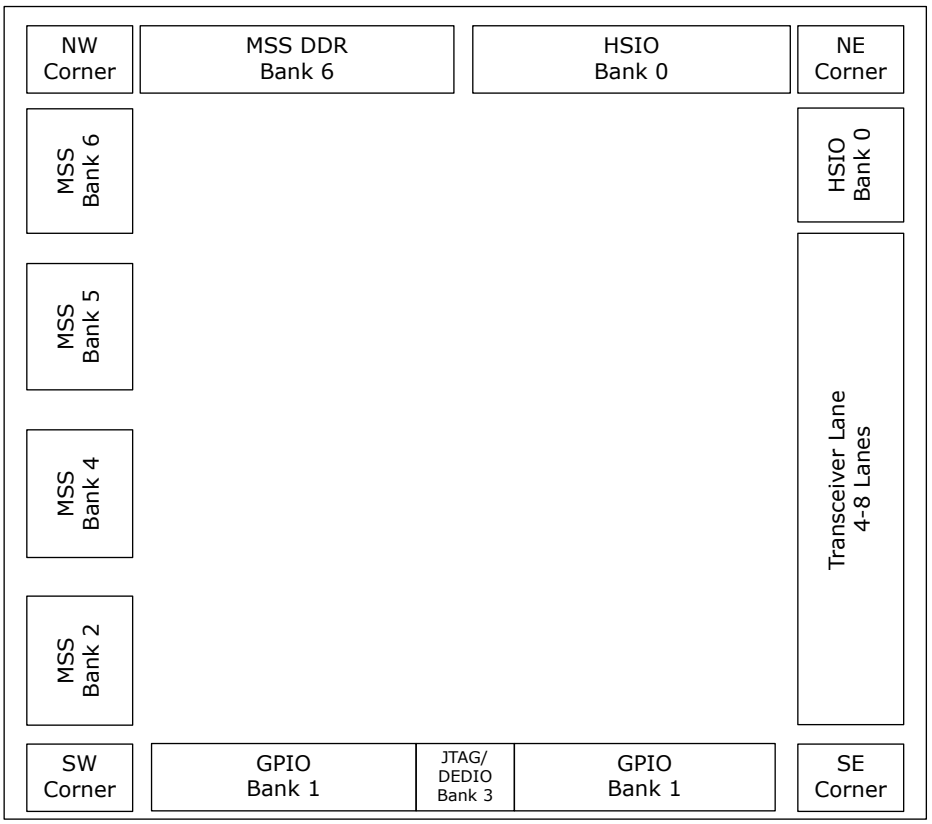


Figure 3-9. PolarFire SoC FPGA MPFS025 I/O Banks



4. Supply Voltages for I/O Banks (Ask a Question)

Multiple I/O banks require the following bank power supplies listed in the table.

Table 4-1. Supply Pin

Name	Description	Nominal Operating Voltage
VDDIx	Supply for I/O circuits in a bank	For JTAG bank—1.8V/2.5V/3.3V For GPIO bank—1.2V/1.5V/ 1.8V/2.5V/3.3V For HSIO bank—1.2V/1.5V/1.8V
VDD25	Power for corner PLLs and PNVM	2.5V
VDD18	Power for programming and HSIO receiver	1.8V
VDDAUXx	Auxiliary supply for I/O circuits. Auxiliary supply voltage must be set to 2.5 V or 3.3 V and must be always equal to or higher than VDDIx. See Table 7-14 GPIO Mixed Reference Receiver Mode for legal VDDI and VDDAUX combinations.	Greater than or equal to VDDI. In cases where VDDI and VDDAUX in a given GPIO bank are both 2.5V or 3.3V, they must be tied together to the same supply.
VREF	VREF is the supply reference voltage for reference receivers. Each bank can have only one VREF value. VREF can be externally supplied or internally generated, see Supply Voltages for I/O Banks for VREF assignment use model for more information.	Depends on the I/O standards
VDDIx MSSIO Banks[1]	Supply for MSS I/O circuits in a bank	1.2V/1.5V/1.8V/2.5V/3.3V
VDDIx MSS-SGMII Banks[1]	Supply for MSS-SGMII circuits in a bank	2.5V/3.3V
VDDIx MSS-DDR Banks[1]	Supply for MSS-DDR circuits in a bank	1.2V/1.5V/1.8V

Note:

1. For PolarFire SoC and RT PolarFire SoC FPGA only.

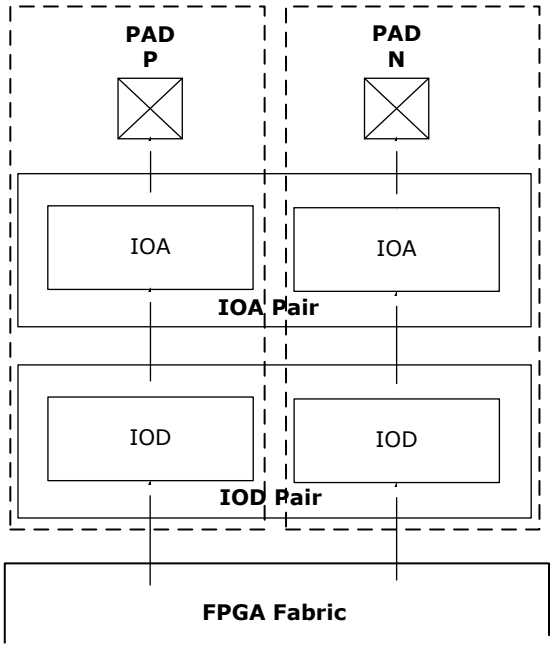
Important: For information about unused condition, see respective PolarFire FPGA Board Design User Guide, PolarFire SoC FPGA Board Design Guidelines User Guide, or RT PolarFire FPGA Board Design User Guide and PPAT.

5. I/O Overview (Ask a Question)

Each I/O is composed of an analog I/O buffer (referred to as IOA) and a digital logic block (referred to as IOD). IOA blocks include analog input and output buffers, while IOD blocks include a logic that enables the IOA buffer to interface with the FPGA fabric. The IOD also includes data bus digital logic to widen the bus to and from the IOA, allowing the external pins to run at a much faster clock rate than the fabric logic.

To support a variety of I/O standards, I/Os are organized into pairs, as shown in the following illustration. The two I/O paths in a pair, labeled as positive (P) and negative (N) respectively, can be configured as two separate single-ended I/Os, as one differential or as a complementary I/O pair.

Figure 5-1. I/O Pair



The IOA buffer includes a transmit and receive buffer, on-die termination (Thévenin, differential, up, and down), a slew-rate control circuit, a bus-keeper circuit, and a programmable weak pull-up or pull-down resistor. The transmit and receive buffers transfer signals between the I/O pad and the IOD. Figure 7-1 shows the overview of IOA buffer.

5.1. Single-Ended Transmitter and Receiver Mode (Ask a Question)

An I/O buffer can be configured as a single-ended transmitter, a single-ended receiver, or both. GPIO and HSIO both support single-ended mode.

5.2. Differential Transmitter Mode (Ask a Question)

The I/O buffer pair allows implementing both true differential output mode and pseudo-differential output mode. The true differential output mode uses an LVDS H-bridge-type driver. The pseudo-differential output mode, also known as complementary-mode, consists of two single-ended drivers where one driver's output is inverted relative to the other. The pseudo-differential output drivers have lower signal integrity and performance, and usually require biasing by external resistors to emulate true differential signal levels. Only GPIO bank supports true differential output modes using a differential current driver. Both GPIO and HSIO banks support complementary output modes.

5.3. Differential Receiver Mode (Ask a Question)

Both GPIO and HSIO receivers support operations in differential receiver mode, where the input data from the differential pair of pads (PAD P and PAD N) is received on both pads and is then driven to the FPGA fabric from the IOD block on the P side.

Libero® SoC controls the enabling and disabling of the transmit and receive buffer based upon the selected standard and I/O mode, whether single-ended or differential. For more information about IOA buffer and its use model, see I/O Features and Implementation, page 15.

5.4. I/O Digital (IOD) (Ask a Question)

The IOD block interfaces with the FPGA fabric on one side and the IOA buffers on the other side. It deserializes and transfers input data to a lower core clock speed, or transfers lower-speed data from the fabric to the high-speed output clock domain, serializing it in the process. The I/O digital block works in conjunction with fast and low-skew clock networks. It also includes special clock dividers and other supported circuits to guarantee clock domain crossings. The I/O digital block deserializes high-speed DDR input data and transfers to FPGA fabric at lower speeds, and also serializes the lower speed FPGA fabric data and transfers to high-speed DDR output. For more information about IOD buffer and its use models, see IOD Features and User Modes, page 38.

6. I/O Primitive (Ask a Question)

The macro library includes a list of I/O primitives to support various I/O standards. Following are the generic I/O primitives, representing most of the available I/O standards.

- INBUF—represents input buffer
- INBUF_DIFF—represents differential input buffer
- OUTBUF—represents output buffer
- OUTBUF_DIFF—represents differential output buffer
- TRIBUFF—represents tristate buffer
- TRIBUFF_DIFF—represents differential tristate buffer

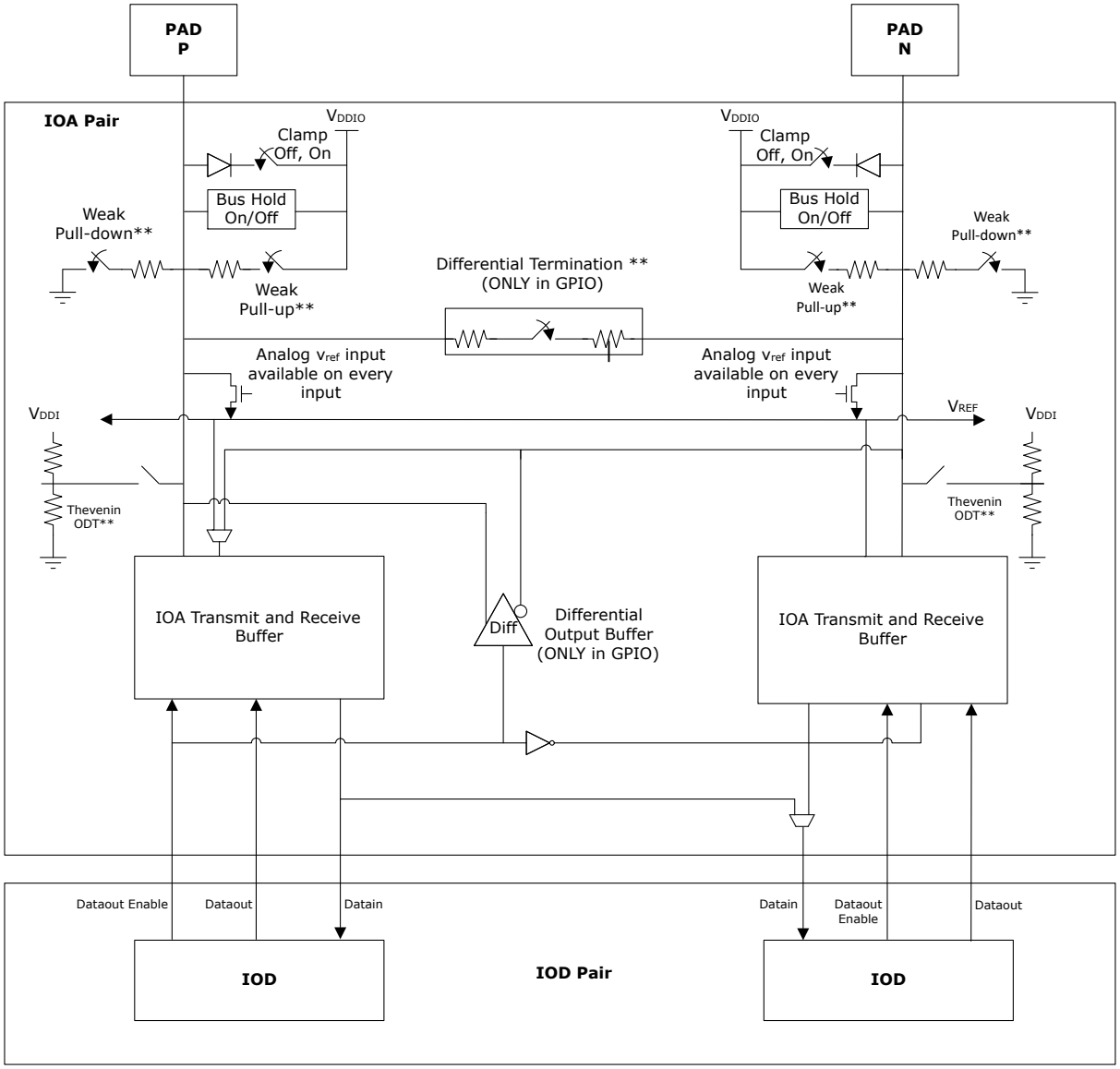
For more information about macro library, see PolarFire FPGA and PolarFire SoC FPGA Macro User Guide.

7. I/O Features and Implementation (Ask a Question)

This chapter describes I/O features and provide details about their use. It also provide guidelines for implementing the various I/O standards using I/Os. The terms receive and input, transmit and output are used interchangeably in this document.

The following illustration shows the I/O pair block diagram.

Figure 7-1. I/O Pair (Detailed View) Block Diagram



Important: The weak pull-up, pull-down, and on-die termination (ODT) ranges are listed in respective PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet.

7.1. I/O Analog (IOA) Buffer Programmable Features (Ask a Question)

GPIO and HSIO provide a number of programmable features. These features are set using the I/O attribute editor in Libero SoC or through PDC commands. The following sections describe these features. For information about PDC constraints, see PDC Commands User Guide for PolarFire FPGA.

7.1.1. Slew Rate Control (Ask a Question)

GPIO supports slew rate control in non-differential output mode. In I/O Editor or the PDC file, when SLEW is set to ON, the device uses a limited slew rate for the I/O standard in the device. In I/O Editor or the PDC file, when SLEW is set to OFF, the device uses the fastest slew rate for the I/O standard. The impacts to Simultaneous Switching Noise (SSN) can be reduced by using the SLEW set to ON. However, this reduces the maximum rate of change of the output signal that can influence switching performance.

The following table lists the I/O standards that support slew rate control.

Table 7-1. Slew Rate Control

I/O Standards	Supported I/O Type	Options
PCI	GPIO (output only)	ON (default), OFF
LVTTL	GPIO (output only)	ON (default), OFF
LVCMOS25 and LVCMOS33	GPIO (output only)	ON (default), OFF

Slew rate settings are controlled using the I/O attribute editor in Libero SoC, or by using the following PDC command:

```
set_io -SLEW <value>
```

The value can be set as ON or OFF.

Slew rate control is not available in HSIO buffers. However, these buffers have built-in PVT-compensated slew rate controllers for optimized signal integrity.

7.1.2. Programmable Weak Pull- Up/Down and Bus-Keeper (Hold) Circuits (Ask a Question)

PolarFire family FPGAs have a programmable weak pull-down (20 KΩ typical), pull-up (20 KΩ typical), and bus-keeper circuit on every I/O pad when in input and output mode. Weak pull-up and pull-down circuits create a default setting for an input when it is not driven. For outputs, the weak pull-up and pull-down can be optionally programmed to set an initial level on the output pad before being actively driven. The bus-keeper circuit is used to weakly hold the signal on an I/O pin at its last driven state, keeping it at a valid level with minimal power dissipation. The bus-keeper circuitry also pulls undriven pins away from the input threshold voltage where noise can cause unintended oscillation. See device/package specific PPAT spreadsheet for default programming of weak pull-up/pull-down for unused pins from Libero SoC.

The following table lists the I/O standards that support weak pull-up/down and bus-keeper control.

Table 7-2. Weak Pull and Bus-Keeper Control

I/O Standards	Supported I/O Types	Options
LVTTL LVCMOS33, LVCMOS25, LVCMOS18, LVCMOS15, and LVCMOS12 PCI	GPIO (input only)	OFF Weak pull-down Weak pull-up Bus-keeper

Table 7-2. Weak Pull and Bus-Keeper Control (continued)

I/O Standards	Supported I/O Types	Options
LVCMOS18, LVCMOS15, and LVCMOS12	HSIO (input only)	OFF Weak pull-down Weak pull-up Bus-keeper

The programmable weak pull-down, pull-up, and bus-keeper settings are controlled by using the I/O attribute editor in Libero SoC or by using the following PDC command:

```
set_io -RES_PULL <value>
```

The value can be set as up, down, hold, or none.

Weak pull-up/pull-down is optionally available with PDC or I/O Editor for differential inputs. Optional pull-up/pull-down resistors might interfere with signal integrity. Users must simulate LVDS communications when using pull-up/pull-down options.

The following table lists the acceptable values for the `-RES_PULL` attribute for the input buffer.

Table 7-3. -RES_PULL Values (Input Buffer)

I/O Standards	Value	Description
Single I/Os: LVTTL, LVCMOS33, LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, and PCI	Up	Includes a weak resistor for pull-up of the input buffer
	Down	Includes a weak resistor for pull-down of the input buffer
	Hold	Holds the last value
	None	Does not include a weak resistor
Differential I/Os: PPDS25, PPDS33, HCSL33, HCSL25, BUSLVDSE25, LVDS33, LVDS25, LVDS18G, MINILVDS33, MINILVDS25, RSDS33, RSDS25, LVPECL33, SUBLVDS33, SUBLVDS25, LCMDS33, and LCMDS25	Up	Includes a weak resistor for pull-up of the input buffer
	Down	Includes a weak resistor for pull-down of the input buffer

Important: The default value of the `-RES_PULL` attribute for input buffer is “Up”.

The following table lists the acceptable values for the `-RES_PULL` attribute for the output buffer.

Table 7-4. -RES_PULL Values (Output Buffer)

I/O Standards	Value	Description
Single I/Os: LVTTL, LVCMOS33, LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, and PCI	Up	Includes a weak resistor for pull-up of the output buffer
	Down	Includes a weak resistor for pull-down of the output buffer
	None	Does not include a weak resistor

Important: The default value of the `-RES_PULL` attribute for output buffer is “None”.

7.1.3. Schmitt Trigger Input Hysteresis (Ask a Question)

GPIO and HSIO can be configured as a Schmitt Trigger input that, when enabled, exhibits a hysteresis that helps to filter out the noise at the receiver and prevents double-glitching caused by noisy input edges.

The following table lists the I/O standards that support the Schmitt Trigger feature. For more information about hysteresis values for different I/O standards when Schmitt Trigger mode is enabled, see respective PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet.

Table 7-5. Schmitt Trigger Control

I/O Standards	Supported I/O Types	Options
• LVTTL • LVCMOS33 • LVCMOS25 • PCI	GPIO (input only)	• ON • OFF The default is “OFF”.
• LVCMOS15 • LVCMOS18	HSIO (input only)	• ON • OFF The default is “OFF”.

Schmitt Trigger mode is enabled by using the I/O attribute editor in Libero SoC or by using the following PDC command:

```
set_io -SCHMITT_TRIGGER <value>
```

The value can be set as ON or OFF.

7.1.4. Programmable Output Drive Strength (Ask a Question)

For LVCMOS, LVTTL, LVDS, and PPDS I/O standards, the I/O output buffer has programmable drive strength control to mitigate the effects of high-signal attenuation caused by long transmission lines.

The following table lists the programmable drive strength support and settings.

Table 7-6. Programmable Drive Strength Control

I/O Standards	Supported I/O Types	Drive Strength Settings (mA)
LVTTL	GPIO (output only)	2, 4, 8, 12, 16, 20 **Note:** Default is 8.
LVCMOS33	GPIO (output only)	2, 4, 8, 12, 16, 20 **Note:** Default is 8.
LVCMOS25	GPIO (output only)	2, 4, 6, 8, 12, 16 **Note:** Default is 8.
LVDS25 and LVDS33	GPIO (output only)	3, 3.5, 4, 6[1] **Note:** Default is 6.
RSDS33 and RSDS25	GPIO (output only)	1.5, 2, 3 **Note:** Default is 3.
MINILVDS33 and MINILVDS25	GPIO (output only)	3, 3.5, 4, 6 **Note:** Default is 6.
SUBLVDS33 and SUBLVDS25	GPIO (output only)	1, 1.5, 2, 3 **Note:** Default is 2.
PPDS33 and PPDS25	GPIO (output only)	1.5, 2, 3 **Note:** Default is 3.

Table 7-6. Programmable Drive Strength Control (continued)

I/O Standards	Supported I/O Types	Drive Strength Settings (mA)
LVCMOS18	GPIO and HSIO (output only)	2, 4, 6, 8, 10, 12 **Note:** Default is 8.
LVCMOS15	GPIO and HSIO (output only)	2, 4, 6, 8, 10 **Note:** Default is 8.
LVCMOS12[2]	GPIO and HSIO (output only)	2, 4, 6, 8, 10 **Note:** Default is 8.

(1) Recommendation to use 100Ω source termination with 6 mA LVDS output drive strength, that is, `SOURCE_TERM` = 100 when `OUT_DRIVE` = 6.
(2) LVCMOS12 output drive strength of 10 mA is supported only for HSIO.

The programmable drive strength is set by using the I/O attribute editor in Libero SoC or by using the following PDC command:

```
set_io -OUT_DRIVE <value>
```

Values can be set as listed in Table 7-6.

7.1.5. Programmable Output Impedance Control (Ask a Question)

For voltage reference I/O standards, I/Os provide the option to control the driver impedance for certain I/O standards such as SSTL, HSUL, HSTL, POD, and LVSTL.

The following table lists the programmable output impedance support and settings.

Table 7-7. Programmable Output Impedance Standards

I/O Standards	Supported I/O Types	Impedance (Ω)
SSTL25I	GPIO	48, 60, 80, 120
SSTL25II	GPIO	34, 40, 48, 60
SSTL18I	GPIO and HSIO	40, 48, 60, 80
SSTL18II	GPIO and HSIO	30, 34, 40, 48
SSTL15I	GPIO and HSIO	40, 48
SSTL15II	GPIO and HSIO	27, 30, 34
SSTL135I	HSIO	40, 48
SSTL135II	HSIO	27, 30, 34
HSUL18I	GPIO and HSIO	34, 40, 55, 60
HSUL18II	GPIO and HSIO	22, 25, 27, 30
HSTL15I	GPIO and HSIO	34, 40, 50, 60
HSTL15II	GPIO and HSIO	22, 25, 27, 30
HSTL135I	HSIO	34, 40, 50, 60
HSTL135II	HSIO	22, 25, 27, 30
HSUL12I	HSIO	34, 40, 48, 60, 80, 120
POD12I	HSIO	40, 48, 60
POD12II	HSIO	27, 30, 34
LVSTLI	HSIO	30, 34, 40, 48, 60, 80, 120, 240
LVSTLII	HSIO	30, 34, 40, 48, 60, 80, 120, 240

The output impedance values can be programmed by using the I/O attribute editor in Libero SoC or by using the following PDC command:

```
set_io -IMPEDANCE <value>
```

Values can be set as listed in Table 7-7.

7.1.6. Differential Near End Termination (Ask a Question)

Programmable output termination is provided for many differential output types. By default, applications with differential signaling is terminated at the receiver (or far-end). However, near-end or source termination can be used to improve signal integrity in lossy connections.

Table 7-8. Source Termination Support

I/O Standard	Values
LVDS18G, LVDS25, LVDS33, MINILVDS25, MINILVDS33, LCMDS33, LCMDS25, PPDS25, PPDS33, RSDS25, RSDS33, SUBLVDS25, SUBLVDS331	OFF, 100. (Default = OFF)

The source termination values can be programmed by using the I/O attribute editor in Libero SoC or by using the following PDC command:

```
set_io -SOURCE_TERM <value>
```

Important: Source termination is required for 1600 Mbps/800 MHz clock.

7.1.7. On-Die Termination (ODT) (Ask a Question)

ODT is used to terminate input signals, helping to maintain signal quality, saving board space, and reducing external component costs. ODT is available in receive mode and also in bidirectional mode when the I/O acts as an input. If ODT is not used or not available, the I/O standards may require an external termination for better signal integrity. For more information, see I/O External Termination.

ODT can be a pull-up, pull-down, differential, or Thévenin termination with both static and dynamic control available, and is set by using the I/O attribute editor in Libero SoC or by using a PDC command.

The following table lists ODT support in GPIO and HSIO.

Table 7-9. ODT Support in GPIO and HSIO

I/O Standards	I/O Types (Input Only)	ODT Control	ODT Type	ODT (Ω)
LVDS18G, LVDS33, LVDS25 RSDS33, RSDS25, MINILVDS33, MINILVDS25, SUBLVDS33, SUBLVDS25, LVPECL33,	GPIO	OFF ON Dynamic	OFF Differential	100
SSTL18I, SSTL18II	GPIO, HSIO	OFF ON Dynamic	OFF Thévenin	50, 75, 150 **Note:** The default is 50.
SSTL15I, SST15II	GPIO, HSIO	OFF ON Dynamic	OFF Thévenin	20, 30, 40, 60, 120 **Note:** The default is 30.
SSTL135I, SSTL135II	HSIO	OFF ON Dynamic	OFF Thévenin	20, 30, 40, 60, 120 **Note:** The default is 40.
POD12I, POD12II	HSIO	OFF ON Dynamic	OFF Up	34, 40, 48, 60, 120, 240 **Note:** The default is 60.

Table 7-9. ODT Support in GPIO and HSIO (continued)

I/O Standards	I/O Types (Input Only)	ODT Control	ODT Type	ODT (Ω)
HSUL12I, HSUL12II	HSIO	OFF ON Dynamic	OFF Up	120, 240 **Note:** The default is 120.
HSTL15I, HSTL15II	GPIO	OFF ON Dynamic	OFF Differential	50
HSUL18I, HSUL18II	GPIO, HSIO	OFF ON Dynamic	OFF Differential	50
LVCMOS25	GPIO, HSIO	OFF ON	OFF Down	120, 240
LVCMOS18, LVCMOS15, LVCMOS12	GPIO, HSIO	OFF ON	OfFF Up Down Thévenin	60, 120, 240 **Note:** The default is 120.

Important: GPIO banks can support 2.5V and 3.3V inputs with VDDI = 1.8V or less.

Select ON in the ODT control to statically set to the `ODT_VALUE`. Select DYNAMIC to enable the `ODT_VALUE` when the `ODT_EN` pin is applied. The static ODT setting and values can be programmed by using the I/O attribute editor in Libero SoC or by using the following PDC command.

```
set_io -ODT <value> -ODT_VALUE <odt_value>
```

Value can be set as ON or OFF and `ODT_VALUE` can be set as listed in Table 7-9.

7.1.8. Common Mode Voltage (Vcm) Settings (Ask a Question)

GPIO and HSIO inputs allow common mode settings for differential receivers. It assists in preventing common-mode mismatches between devices.

The following table lists the programmable differential termination control support and settings. For more information about common mode voltage levels for various I/O standards, see respective PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet.

Table 7-10. Programmable Differential Termination Control

I/O Standards	Supported I/O Types	Differential Termination Type[1]
SSTL18	GPIO, HSIO	Off, Low, Mid
HSUL18	GPIO, HSIO	Off, Low, Mid
SSTL15	GPIO, HSIO	Off, Low, Mid
HSTL15	GPIO, HSIO	Off, Low, Mid
SSTL135	HSIO	Off, Low, Mid
HSTL135	HSIO	Off, Low[1], Mid
HSUL12I	HSIO	Off, Low, Mid
HSTL12	HSIO	Low, Mid
POD12	HSIO	Off, Low, Mid
SSTL25	GPIO	—

Table 7-10. Programmable Differential Termination Control (continued)

I/O Standards	Supported I/O Types	Differential Termination Type[1]
SLVS25	GPIO, HSIO	MID (HSIO) Low, Mid (GPIO) **Note:** The default is MID.
HCSL25	GPIO, HSIO	MID (HSIO) Low, Mid (GPIO) **Note:** The default is MID.
SLVSE	GPIO, HSIO	Off, Mid (HSIO) Off, Low, Mid (GPIO) **Note:** The default is MID.
PPDS25	GPIO, HSIO	Mid (HSIO) Off, Low, Mid (GPIO) **Note:** The default is MID.
MLVDSE	GPIO	Off, Low, Mid **Note:** The default is MID.
BUSLVDS	GPIO	Off, Low, Mid **Note:** The default is MID.
LVPECL	GPIO	Low, Mid **Note:** The default is MID.
LVDS	GPIO, HSIO	Mid (HSIO) Off, Low, Mid (GPIO) **Note:** The default is MID.
RSDS	GPIO, HSIO	Mid (HSIO) Off, Low, Mid (GPIO) **Note:** The default is MID.
MINILVDS	GPIO, HSIO	Mid (HSIO) Off, Low, Mid (GPIO) **Note:** The default is MID.

Note:

1. For more information about low and mid differential termination types, see respective PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet.

Important: For GPIO banks, whenever a mixture of Low and Mid input common mode ranges (VCM_RANGE) are used for true differential inputs within the same GPIO bank, the input pairs that also enable the internal 100Ω On-Die Termination (ODT) resistor have a resistor accuracy tolerance percentage that follows the maximum percentage tolerance of the two ranges, as per the differential termination accuracy specifications in the datasheet. If a smaller termination resistor tolerance is desired on a particular differential input pair, use an external termination resistor on the board and disable the internal differential ODT.

The programmable differential termination control values can be programmed by using the I/O attribute editor in Libero SoC or by using the following PDC command:

```
set_io -VCM_RANGE <value>
```

Value can be set as listed in Table 7-10.

7.1.9. Programmable Clamp Diode (Ask a Question)

Both HSIO and GPIO have internal clamp diodes. Clamp diodes help reduce the voltage level at the input, and are mainly used when the voltage overshoot exceeds the maximum allowable limit. Although, the HSIO clamp is always ON; it is not a PCI clamp. PCI clamp is only on GPIO. If signaling levels of the receiver are greater than the VDDIx of the bank, the clamp diode must be OFF to support hot-swapping insertion. For more information, see Cold Sparing and Hot Swap.

As shown in Figure 7-1, GPIOs have optional clamp diode to VDDIO (not to GND) for complying to PCI standard. For GPIOs, these clamp diodes can be programmed to be ON or OFF by using the I/O attribute editor in Libero SoC or by using a PDC command. For HSIO, the internal clamp diode is always ON.

The following table lists the values for GPIO standards. For HSIO standards, the value is always ON.

Table 7-11. Programmable Clamp Diode

I/O Standards	Values
LVCMOS12, LVCMOS15, LVCMOS18, SSTL18I, SSTL18II, SSTL15I, SSTL15II, HSTL15I, HSTL15II, LVTTL, LVCMOS33, LVCMOS25, SSTL25I, SSTL25II, SLVS25, HCSL25, LVDS25, RSDS25, MINILVDS25, SUBLVDS25, PPDS25, LCMDS25	OFF, ON. The default is ON.
MIPI25, LVDS18G	OFF, ON. The default is OFF.
HSUL18I, HSUL18II, SLVSE15, PCI, SLVS25, SLVS33, HCSL33, HCSL25, MIPIE33, MIPIE25, LVPECL33, LVPECL25, LVPECLE33, LVDS25, LVDS33, RSDS25, RSDS33, MINILVDS25, MINILVDS33, SUBLVDS25, SUBLVDS33, PPDS25, PPDS33, MLVDSE25, BUSLVDSE25, LCMDS25, LCMDS33	ON

The following PDC command is used for programmable clamp diode settings:

```
set_io -CLAMP_DIODE <value>
```

Important: The clamp diode is always on for HSUL18I, HSUL18II, SLVSE15, MIPI25, PCI, SLVS33, HCSL33, MIPIE25, LVPECL33, LVPECLE33, LVDS25, LVDS33, RSDS25, RSDS33, MINILVDS25, MINILVDS33, SUBLVDS25, SUBLVDS33, PPDS25, PPDS33, MLVDSE25, and BUSLVDSE25 I/O standards implemented in GPIO bank.

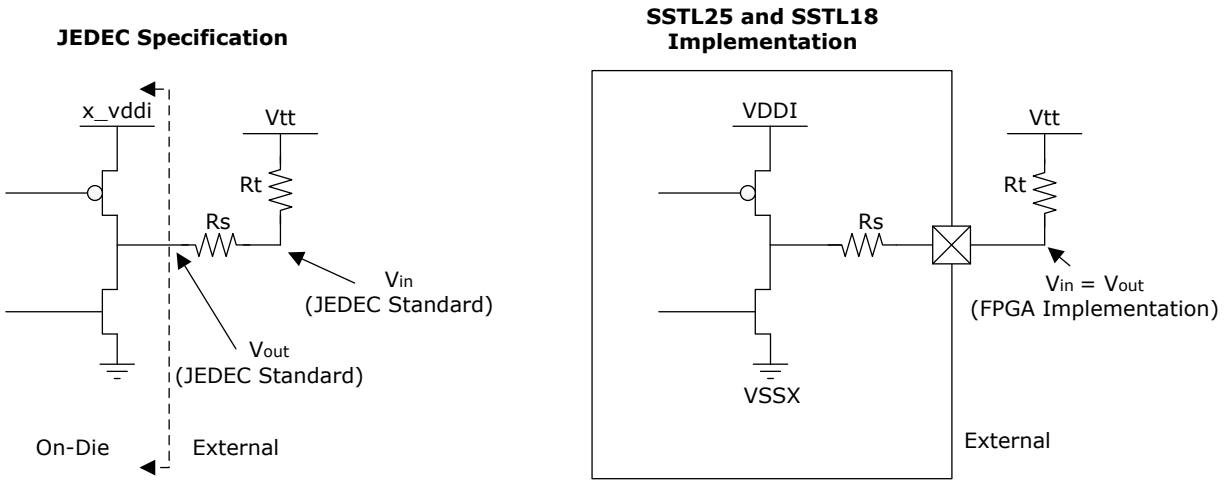
7.1.10. Compensated Drive Impedance and Terminations (Ask a Question)

Resistors are used to match the impedance of the trace. However, adding resistors close to device pins increases the size of the board area and component count, and can in some cases be physically impossible. To address these issues, a reference controller between the VDDI power supply, and pad signal is used to control the source and sink drivers between the pad and the ground. This compensation happens at power up, and on-demand by the user logic. The I/O compensation adjusts the impedances inside the GPIO or HSIO bank by comparing to the internal reference. The impedance change in I/O compensation is due to process variation. The compensation logic adjusts the impedance of the GPIO or HSIO by selectively turning the transistors ON or OFF in the I/Os. The impedance is adjusted to match the internal reference by doing an initial adjustment when the power-on detector for VDDI and VDDAUX gets to a minimal value. The change in impedance also compensates for Temperature variation and Supply Voltage fluctuations. Both input and output compensation are a function performed as part of I/O calibration. See I/O Calibration.

7.1.11. SSTL25 and SSTL18 Stub Resistor (Ask a Question)

For stub-series interface standard SSTL, the output drive also includes the stub resistor. I/Os support this stub resistor for SSTL25 and SSTL18 I/O standards (Figure 7-2). This feature reduces both cost and board complexity.

Figure 7-2. SSTL25 and SSTL18 Stub Resistor



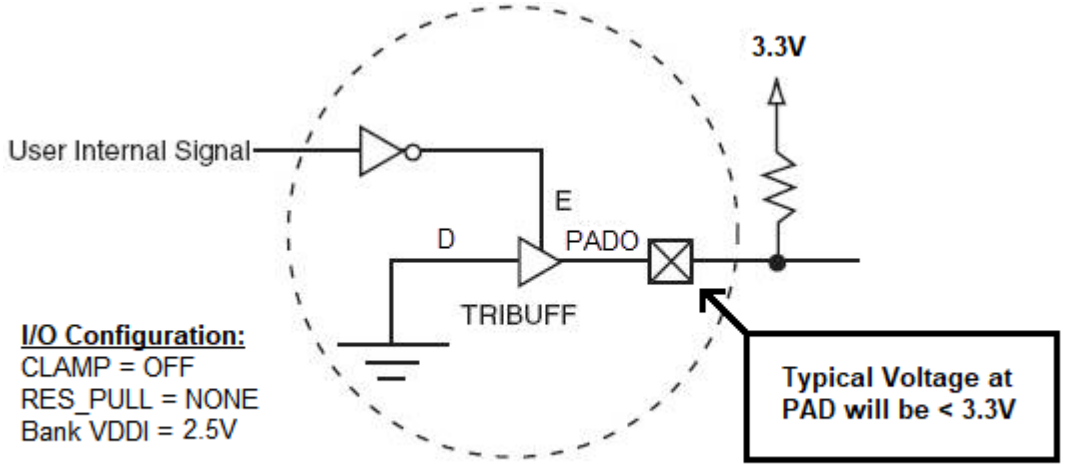
7.1.12. Shield (Ask a Question)

Shield IOTYPE are provided for “soft ground” pins to improve the localized references. These are actual I/O pins that are re-purposed to isolate switching noises around high-speed I/O interfaces. Shields are only implemented on memory interfaces on the unused DQ bits in specific device/package combinations. This rule applies to GPIO and HSIO based DDRx memory interfaces. For maximum shielding benefit, it is recommended to tie these SHIELD signals to VSS on the board for the specific device/packages included in the PPAT description. This pin function is only available when using PolarFire family Memory Controllers. For those combinations that do have Shield pins defined, Libero enforces to use specific pins as Shields. If Shields are identified in the PPAT and Libero SoC software, it is recommended to connect them to GND on the PCB as specified.

7.1.13. Open Drain GPIO (Ask a Question)

GPIO can be used to create an open-drain output when VDDI is below the required high output level. In this case, the GPIO pin only drives a LOW output. When not driving LOW, it is externally pulled up to a maximum of 3.3V through an external 200Ω pull-up resistor. This is accomplished by selecting the CLAMP = OFF and the internal RES_PULL = NONE. The actual voltage at the PAD output will be lower than 3.3V, depending on the value of the pull-up resistor. The user must drive the enable pin of the tristate buffer through an inverter to reflect the user logic High at the PAD Output.

Figure 7-3. Open Drain GPIO Example



Important: External pull-up values of 250Ω are suggested up to 33 MHz operations. 200Ω pull-up is suggested for up to 50 MHz. The external pull-up resistor acts as a voltage divider between 3.3V and the bank VDDI. Based on the pull-up resistor value and bank VDDI chosen, the PAD output voltage will accordingly be lower than 3.3V.

7.1.14. 3.3V Tolerant Input (Ask a Question)

As the demand to consolidate power supplies to lower power-supply voltages, bus translators are often necessary to interface between separately powered components of a logic system. However, the GPIO can operate as an LVCMOS25 input (VDDI = 2.5V) and reliably receive a 3.3V input signal. This is done by adding a 250Ω series resistor and configuring the LVCMOS25 input with the CLAMP = OFF. This configuration is suitable for up to 50 MHz. You must perform IBIS simulations to verify proper performance.

7.2. I/O Implementation Considerations (Ask a Question)

This section provides the generic guidelines when implementing various I/O standards. In addition, it also provides details of I/O states during various device operational modes such as power-up and initialization.

7.2.1. Reference Voltage for I/O Bank (Ask a Question)

Each voltage-referenced I/O standard needs a reference voltage (VREF) for inputs while in operation. Each bank contains a single reference voltage bus, which can either be externally supplied through an I/O in the bank or generated internally by the bank controller.

7.2.1.1. External VREF Input (Ask a Question)

Any GPIO or HSIO pad on the device can be programmed to act as an external VREF input to supply all inputs within a bank. When an I/O pad is configured as a voltage reference, all I/O buffer modes and terminations on that pad are disabled. External VREF is supported for both GPIO and HSIO banks. By default, Libero SoC uses the internal VREF.

Use PDC or the I/O Attribute editor to choose any regular I/O to make it a VREF pin.

This is an example of a PDC command:

```
set_iobank -bank_name Bank0 \
-vcci 1.80 \
```

```
-vref 0.90 \
-vref_pins { U5 } \
-fixed false
set_iobank -bank_name Bank2 \
-vcci 1.80 \
-vref 0.90 \
-vref_pins { A2 } \
-fixed false
```

Important: When external VREF is used, the voltage on VREF pins can be any value between 0 and VDDI. However, the value of the -VREF attribute is specified in PDC as 50% of VDDI value.

Any available package pin can be selected and set it as a VREF. This requires placement of at least one I/O type requiring a VREF in I/O Editor or PDC.

For more information about external reference inputs, see respective PolarFire FPGA Board Design User Guide, PolarFire SoC FPGA Board Design Guidelines User Guide, RT PolarFire FPGA Board Design User Guide, or *RT PolarFire SoC FPGA Board Design Guidelines User Guide* (to be available in a future release).

7.2.1.2. Internally-Generated VREF (Ask a Question)

Every bank also has an internally-generated VREF available. This internally-generated VREF adds more flexibility and dynamic control. This VREF is Libero SoC programmed (to be 50% of VDDI).

7.2.1.3. MSS DDR VREF (PolarFire SoC and RT PolarFire SoC Only) (Ask a Question)

Bank 6 is unique as it has two internal VREF generators. These VREF generators have separate and different VREFs, one for the CA bus and the other for the DQ bus. When both VREFs are used internally, the values are same, which is 50% of VDDI for that bank. Typically, both VREFs are internal and the same. It may be desirable to provide external VREF for DQ depending on terminations, or simply having the ability to monitor that voltage. For more information about MSS_DDR and VREF options, see DDR I/O.

7.2.2. Mixed I/O in VDDI Banks (Ask a Question)

Each bank has a VDDI supply that powers the single-ended output drivers and the ratio input buffers such as LVTTL and LVCMOS. In addition to the bank VDDI supply, the GPIO banks include an auxiliary supply (VDDAUX) that powers the differential and referenced input buffers. Similarly, in HSIO banks, there are VDDI power pins, however, there are no dedicated VDDAUX pins as the VDD18 supply is used to power the differential and referenced input buffers. This flexibility of power supplies to the I/O provides independence for mixing I/O standards in the same bank.

PolarFire Family FPGA inputs are designed to support mixing assignment for certain I/O standards, allowing I/O using compatible standards to be placed in the same I/O bank. The GPIO are self-protecting, which supports mixed input voltage combinations. It also supports over-voltage conditions because of its hot-swap design. For example, when VDDI is set to 3.3V, a input receiver of 3.3V, 2.5V, 1.8V, and 1.2V. LVCMOS can be placed in the same I/O bank.

The mixing of different I/O within a bank is supported by the Libero SoC software. Before placing any mixed I/O voltage, you must first set the bank to the desired VDDI voltage followed by setting the attributes of the I/O that allows for mixed mode. Placing the I/O must be the last step. When implementing mixed I/O mode restrictions on ODT, CLAMP and RES_PULL must be followed. The HSIO receivers have a reduced set of compatible I/O standards because the I/O clamp-diode is set to ON. For GPIO, if the signaling levels of the receiver are greater than the VDDI of the bank, the clamp must be set to OFF. See the following tables for details on valid attributes.

Important: When using mixed-mode receivers, the DC Input Levels (VIH and VIL) meet the desired I/O Standard level as referenced in PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet. In this case, VIH and VIL are not derived from the actual VDDI supplied to the targeted mixed-mode I/O bank. For example, if 3.3V is connected to the VDDI bank supply, and a mixed I/O Standard is configured to LVCMOS18, the VIL and VIH relates to the LVCMOS18 I/O standard referenced to VDDI = 1.8V in the DC Input Levels specifications. The VIL and VIH do not relate to the actual 3.3V VDDI.

The following tables list VDDI and mixed receiver compatibility for GPIO, HSIO for single-ended, reference and differential inputs. The tables list that inputs can be mixed within specific banks and still meet the I/O standards VIH/VIL requirements independent of the VDDI applied to the banks.

Table 7-12. GPIO LVTTL/LVCMOS I/O Compatibility in Receive Mode[1]

VDDI	LVTTL/LVCMOS33	LVCMOS25	LVCMOS18	LVCMOS15	LVCMOS12
3.3V	Yes	Yes[2]	Yes[2]	No	Yes[2]
2.5V	Yes	Yes	Yes	Yes	Yes
1.8V	Yes	Yes	Yes	Yes	Yes
1.5V	Yes	Yes	Yes	Yes	Yes
1.2V	Yes	Yes	No	Yes	Yes

(1) RES_PULL must be DOWN or NONE. All mixed modes in the preceding table require CLAMP = OFF.
(2) ODT must be OFF.

Table 7-12 lists the compatible I/O types when mixing within the VDDI banks. Using the table for example, a VDDI low voltage of 1.2V in GPIO can include LVCMOS33 inputs. Similarly, a VDDI low voltage of 1.2V cannot include LVCMOS18 inputs.

Table 7-13. HSIO LVCMOS I/O Compatibility in Receive Mode[1]

VDDI	LVCMOS18	LVCMOS15	LVCMOS12
1.8V	Yes	Yes	Yes
1.5V	No	Yes	Yes
1.35V	No	No	Yes
1.2V	No	No	Yes

(1) RES_PULL must be DOWN or NONE. All mixed modes in the preceding table require CLAMP = ON.

The following table lists GPIO mixed reference receiver mode data.

Table 7-14. GPIO Mixed Reference Receiver Mode[1]

VDDI	VDDAUX	SSTL25	SSTL18, HSUL18	SSTL15, HSTL15
3.3V	3.3V	No	No	No
2.5V	2.5V	Yes (mid-range Vcm)	Yes (mid-range Vcm)	Yes (Low-range Vcm)
1.8V	2.5V	Yes (mid-range Vcm and clamp diode off)	Yes (mid-range Vcm)	Yes (Low-range Vcm)
1.5V	2.5V	Yes (mid-range Vcm and clamp diode off)	Yes (mid-range Vcm and clamp diode off)	Yes (Low-range Vcm)
1.2V	2.5V	No	No	No

(1) ODT must be OFF for all cases.

Table 7-15. HSIO HSUL12/HSTL12/POD I/O Compatibility in Receive Mode[1,2]

VDDI	SSTL15 HSUL15	SSTL18 HSTL18	SSTL135 HSTL135	HSUL12 HSTL12 POD
1.8V	No[2] (mid-range Vcm)	Yes (mid-range Vcm)	No[2] (mid-range Vcm)	No[2] (mid-range Vcm)
1.5V	Yes (mid-range Vcm)	No[2]	No[2] (mid-range Vcm)	No[2] (mid-range Vcm)
1.35V	No[2]	No[2]	Yes (mid-range Vcm)	No[2] (mid-range Vcm)
1.2V	No[2]	No[2]	No[2]	Yes (mid-range Vcm)

Notes:

1. ODT must be OFF for all cases.
2. Single-ended I/O with Vref cannot be used in mixed voltage. Single-ended I/O with Vref can only be supported when I/O standard is set to correct VDDI.

Table 7-16. GPIO Differential I/O Compatibility in Receive Mode

VDDI	LVDS25, RSDS25, SUBLVDS25, MINILVDS25, PPDS25, LCMDS25, SLVS25, HCSL25	MIPI25
3.3V	No	Yes (Clamp diode ON or OFF)
2.5V	Yes	Yes
1.8V	Yes	Yes
1.5V	Yes	Yes
1.2V	Yes	Yes

Note: Clamp diode OFF is used for all except where noted.

HSIO differential receivers do not support mixed I/O voltage combinations.

7.2.2.1. LVDS (Ask a Question)

GPIO and HSIO banks can receive LVDS input signals. For GPIO, these inputs have an internal 100Ω differential termination resistor that can be enabled by the Libero SoC software. HSIO does not have this internal resistor capability. HSIO requires a 100Ω resistor across the P and N pair of the LVDS inputs. This requires careful PCB layout to provide this termination close to the device pins. The LVDS25 IOSTD is supported only for input and output I/Os (TRIBUF and INOUT is not supported).

HSIO banks only support LVDS18 inputs. LVDS18 outputs are not available. Only emulated LVDS-like outputs with lower performance are available in HSIO banks. True LVDS outputs are natively available in GPIO banks. Use either VDDI = 2.5V or 3.3V (LVDS25 or LVDS33) or LVDS18G with VDDI = 1.8V and VDDAUX = 2.5V. See LVDS in GPIO Banks with VDDI = 1.8V for more information about GPIO LVDS18G inputs and outputs. For more information about DC specification, see respective PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet. LVDS outputs are not available in HSIO banks.

7.2.2.2. LVDS in GPIO Banks with VDDI = 1.8V (Ask a Question)

LVDS inputs and outputs in the GPIO banks are supported from Libero SoC with VDDI = 1.8V and VDDAUX= 2.5V or 3.3V. Both LVDS18G inputs and outputs operate from the VDDAUX power supply, which allows operation independent of the VDDI power supply. LVDS18G inputs include on-chip differential termination, and true high-speed differential outputs are used in LVDS18G. The LVDS18G IOSTD is supported only for input and output I/Os (TRIBUF and INOUT is not supported).

LVDS18G I/O standard allows Libero SoC to set VDDI = 1.8V, thereby placing other 1.8V I/O on the bank as mixed voltage support, and simultaneously supporting the LVDS modes (see respective PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet), RT PolarFire FPGA Datasheet, or RT

PolarFire SoC Datasheet. If setting the I/O standards from I/O Editor or PDC with VDDI = 1.8V, it requires the selection of LVDS18G as I/O TYPE and VDDAUX = 2.5V (or 3.3V) circuit board voltage supply. LVDS18G input requires Clamp Diode to be OFF when placed on a bank with VDDI = 1.8V.

7.2.3. I/O External Termination (Ask a Question)

If ODT is not used or not available, I/Os require an external termination for better signal integrity. Voltage-referenced standards generally have serial (driver) and parallel (receiver) termination schemes while differential standards only require parallel (receiver) termination.

The following table lists the external termination schemes for the supported I/O standards when the ODT/driver impedance calibration feature is not used.

Table 7-17. I/O External Termination with ODT Off

I/O Standards	External Termination Schemes
SSTL15, SSTL18, SSTL2 single-ended	Single-ended SSTL I/O standard termination
HSTL15	Single-ended HSTL I/O standard termination
SSTL15, SSTL18, SSTL2 differential	Differential SSTL I/O standard termination
HSTL15	Differential HSTL I/O standard termination
LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25, and LVCMOS33	No external termination required
LVDS	100Ω, differential termination (HSIO only)
SLVS	100Ω, differential termination (HSIO only)
MLVDS	100Ω, differential termination (HSIO only)
BLVDS	100Ω, differential termination (HSIO only)
RLVDS	100Ω, differential termination (HSIO only)
Mini-LVDS	100Ω, differential termination (HSIO only)
LVPECL	100Ω, differential termination (HSIO only)

7.2.4. Implementing Emulated Standards for Outputs (Ask a Question)

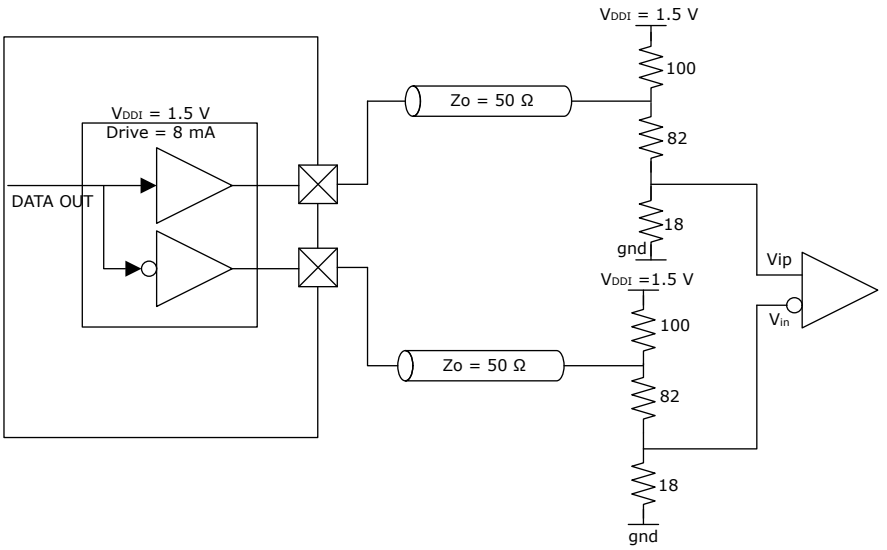
External terminations are required to implement SLVSE, BLVDSE, MLVDSE, and LVPECLE output modes. These outputs, referred to as emulated differential outputs, are noted in Table 7-5.

Emulated differential standards use compensated push-pull drivers in complementary output mode and require external terminations on the board to match the common-mode and voltage swing to meet the I/O signal standards. PCB design practices must account for the effective impedance of the entire connection including the bus trace characteristic impedance Zo and the capacitive loading. This section provides example implementations for the emulated standards.

7.2.4.1. Scalable Low-Voltage Signaling Emulated (SLVSE15) Output Mode (Ask a Question)

GPIO and HSIO support SLVS transmitter with external terminations. The following illustration shows an example of SLVSE implementation. This implementation requires 100Ω, 82Ω, and 18Ω external termination. Additionally, all driver output levels in the implementation are level-shifted by approximately 18%.

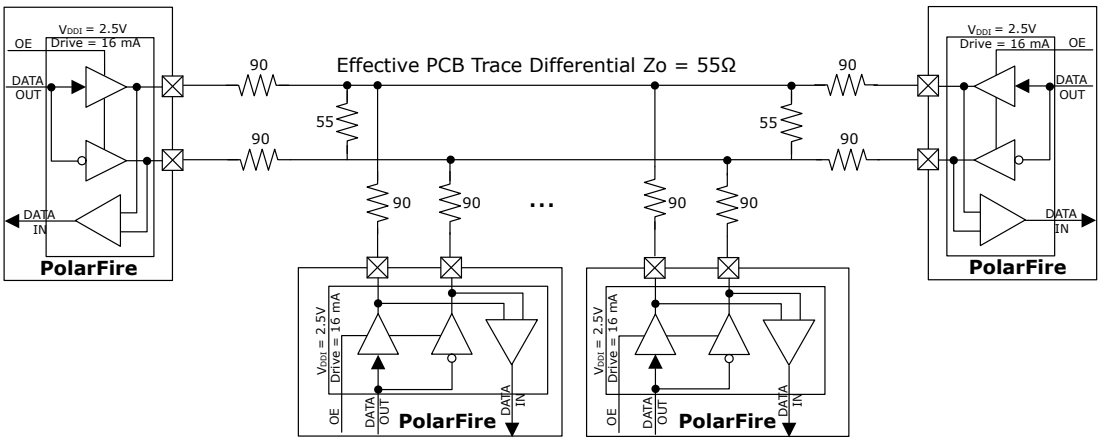
Figure 7-4. SLVSE System Diagram



7.2.4.2. Bus-LVDS Emulated (BLVDSE25) Output Mode (Ask a Question)

BLVDS is used in multipoint, bidirectional, and heavily-loaded backplane applications. The effective impedance of these systems is lower than a typical pair of PCB traces due to the backplane capacitance, the connectors on the backplane, and the line stubs. The following illustration shows an example of BLVDS implementation using 90Ω stub resistors at every drop and 55Ω stub resistors on either side of the bus. The termination values at the end of the bus, which can range anywhere between 45Ω and 90Ω, must be optimized through PCB design simulations to match the effective differential impedance of the bus. In this example, the two parallel 55Ω stub discrete termination resistors yield an effective 27Ω differential termination.

Figure 7-5. Bus-LVDSE System Diagram

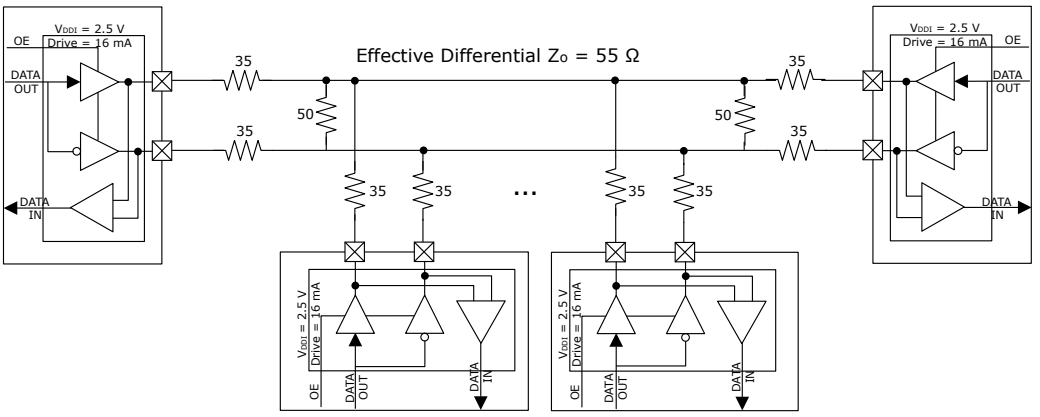


7.2.4.3. Multipoint Low-Voltage Emulated (MLVDSE25) Output Mode (Ask a Question)

MLVDS has larger signaling amplitude when compared to BLVDS, and therefore, it requires more drive current. Similar to BLVDS, the effective impedance of these systems is lower than a typical pair of PCB traces due to backplane capacitance, the connectors on the backplane, and the line stubs. The following illustration shows an example implementation using 35Ω stub resistors at every drop and 50Ω stub resistors on either side of the bus. The termination values at the ends of the

bus, which can range anywhere between 50Ω and 70Ω, must be optimized to match the effective differential impedance of the bus.

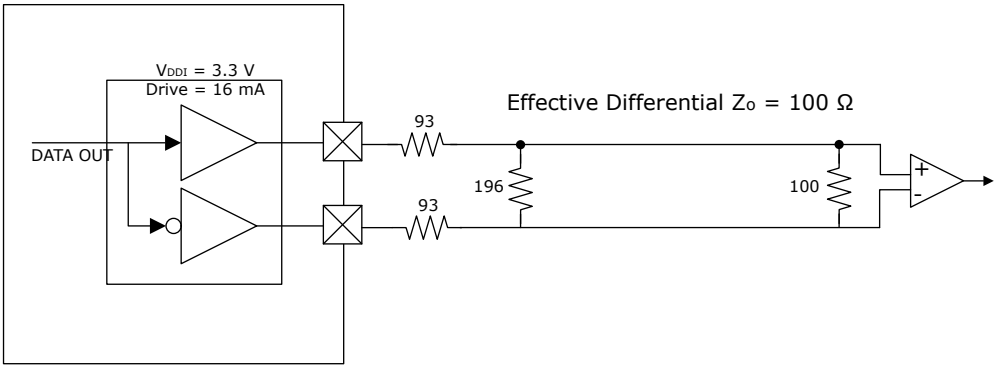
Figure 7-6. MLVDSE System Diagram



7.2.4.4. LVPECL Emulated (LVPECLE33) Output Mode (Ask a Question)

LVPECL is derived from ECL and PECL and uses 3.3V supply voltage. The following illustration shows an example of implementation using 93Ω stub resistors with a 196Ω parallel/differential termination at the driver and a 100Ω differential termination at the receiver. The termination values at the driver must be optimized to match the effective differential impedance of the bus. In this example, the effective parallel differential termination at the receiver is around 66Ω. However, the series 93Ω resistors are always seen by the driver yielding an effective differential impedance of 252Ω. The receivers see an attenuated signal.

Figure 7-7. LVPECL System Diagram



7.2.5. Implementing MIPI D-PHY (Ask a Question)

PolarFire family supports implementation of the MIPI D-PHY standard used in camera and display applications. A minimum D-PHY configuration consists of a clock and one or more data signals. The MIPI D-PHY uses two-conductor connections for both data and clock. Both the device families support MIPI D-PHY with MIPI25 and MIPIE25 I/O types dependent on the interface. See the MIPI25 input and MIPIE25 output in the ac-performance section of respective PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet for MIPI D-PHY performance.

7.2.5.1. MIPI D-PHY Receive Interface (Ask a Question)

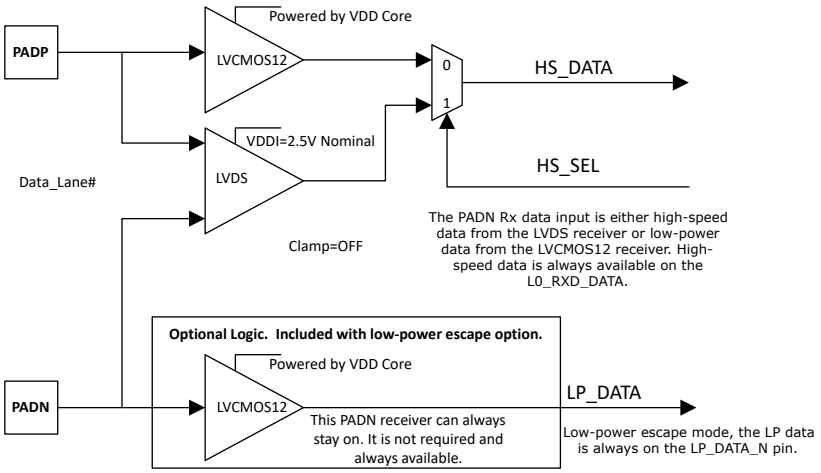
GPIO supports unidirectional MIPI D-PHY I/O in the receive direction, as shown in the following illustration. The MIPI D-PHY receiver supports high-speed (HS) signaling mode for data traffic and low-power (LP) signaling mode used for control. Each HS lane using MIPI25 is terminated and driven by a low-swing, differential signal. LP lanes operate single-ended and not terminated using two MIPI25 outputs driving each connection of the lane independently.

The MIPI receiver supports both the high-speed (HS) and a low-power (LP) receiver modes. These modes are selectable through an enable (HS_SEL) from the IOD component when MIPI low-power escape support is selected in the IOD Generic Receive Interfaces configurator (See Figure 9-2).

When the MIPI25 low-power escape support is used, the I/O is generated with a differential receiver between PADP and PADN. An additional single-ended receiver is connected to the PADP, allowing the HS_SEL signal to select between receivers. It also enables the 100Ω differential termination resistor when HS_SEL = 1. This is generated by Libero SoC when selected in the IOD configurator.

When HS_SEL is selected, the HS_SEL pin serves as the output enable. When HS_SEL = 1, then the HS differential receiver and differential 100Ω termination is turned ON and a single-ended receiver connected to the compliment PADN pin. When HS_SEL = 0, the differential termination is disabled and the single-ended receiver is enabled on the PADN pins. This MIPI interface is implemented by configuring PADP as a MIPI receiver, PADN pin and LVCMOS12 receiver. FPGA hosted logic is required to control this feature.

Figure 7-8. MIPI D-PHY Receiver



FPGA fabric synchronization registers are required when clocking LP_DATA with RX_CLK_R of the IOD to ensure clean capture of the data.

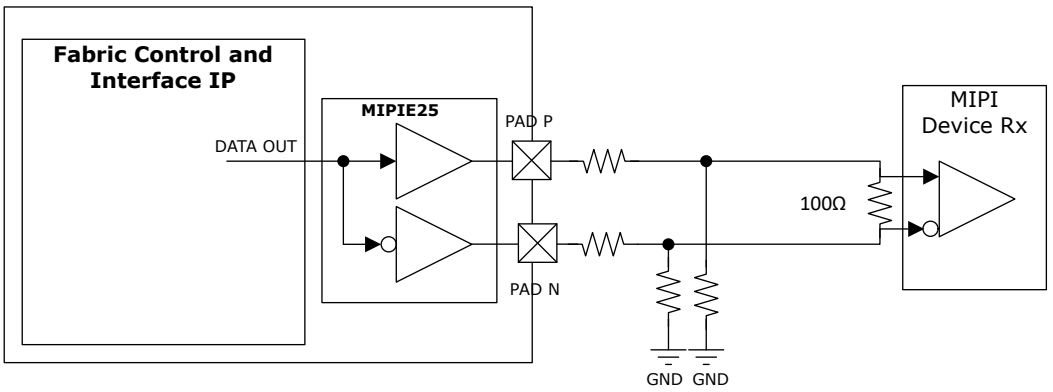
Note: Low-power LVMOS12 inputs are powered by internal VDD core. VDDI is not used with Low-power LVCMOS12 inputs.

Note: For Low-power detection of MIPI, HS_SEL can be tied to Logic one.

7.2.5.2. MIPI D-PHY Transmitting Interface (High-speed Only) (Ask a Question)

GPIO supports unidirectional MIPI D-PHY transmit interface with the external resistors, as shown in the following illustration. Every GPIO P and N pair (MIPIE25) can be configured as a MIPI D-PHY transmit interface. MIPIE25 is a specific output mode that uses complimentary LVCMOS output drivers and external termination resistors. This combination matches the signaling requirements of a MIPI standard receiver as well as meets the performance requirements based on MIPI loading and switching levels.

Figure 7-9. MIPI D-PHY Transmit Interface

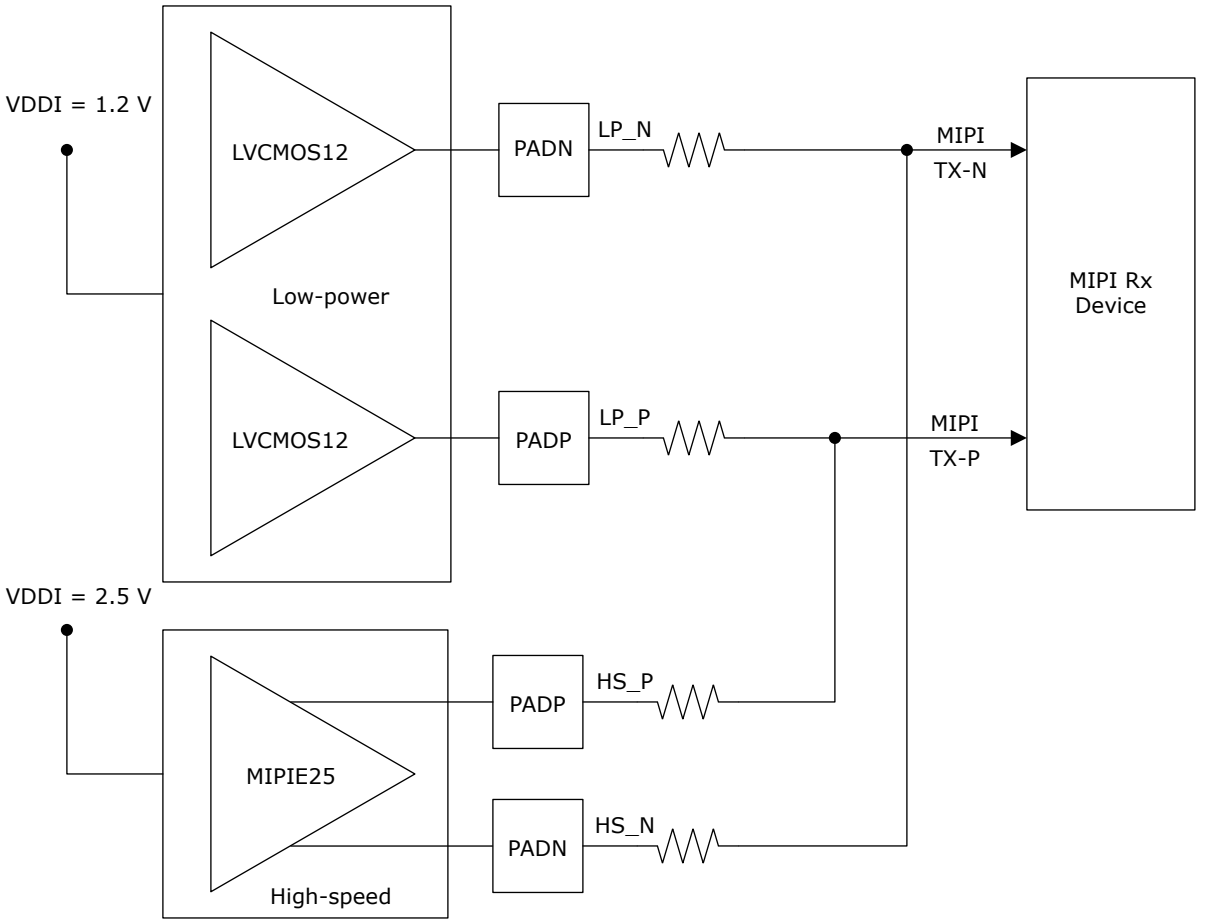


Important: Resistor value vary based on optimal performance. For resistor specifications, see respective PolarFire FPGA Board Design User Guide, PolarFire SoC FPGA Board Design Guidelines User Guide, RT PolarFire FPGA Board Design User Guide, or *RT PolarFire SoC FPGA Board Design Guidelines User Guide* (to be available in a future release).

7.2.5.3. MIPI D-PHY Transmit Only (High-Speed and Low-Power) (Ask a Question)

The MIPI Low-power (LP) transmit signaling uses pins located in either GPIO or HSIO banks using 1.2V VDDI I/O bank supply using LVCMOS12 outputs. High-speed MIPI transmit signals must be in GPIO bank using a 2.5V VDDI I/O bank supply using MIPIE25 emulated differential output drivers. The MIPI TX standards are implemented by using the resistor divider network for LP and High-speed (HS) signals, as shown in the following figure. For required pin-out planning of the HS and LP Tx pins. See respective PolarFire FPGA Board Design User Guide, PolarFire SoC FPGA Board Design Guidelines User Guide, RT PolarFire FPGA Board Design User Guide, or *RT PolarFire SoC FPGA Board Design Guidelines User Guide* (to be available in a future release).

Figure 7-10. MIPI D-PHY Transmit Interface (High-Speed and Low-Power)



Important: Resistor value vary based on optimal performance. For resistor specifications, see respective PolarFire FPGA Board Design User Guide, PolarFire SoC FPGA Board Design Guidelines User Guide, RT PolarFire FPGA Board Design User Guide, or *RT PolarFire SoC FPGA Board Design Guidelines User Guide* (to be available in a future release).

MIPI D_PHY transmits the TXD_DATA out on to the TXD/TX_CLK MIPI pins when the HS_DATA_SEL/HS_CLK_SEL port is asserted controlling the OE of the HS differential driver. The HS_DATA_SEL/HS_CLK_SEL are optionally configured using the Libero SoC IOD configurator.

When HS_DATA_SEL/HS_CLK_SEL is asserted, both single-ended LVCMOS12 8 mA drivers are driven low by the IOD to ensure proper level shifting occurs for high-speed operation.

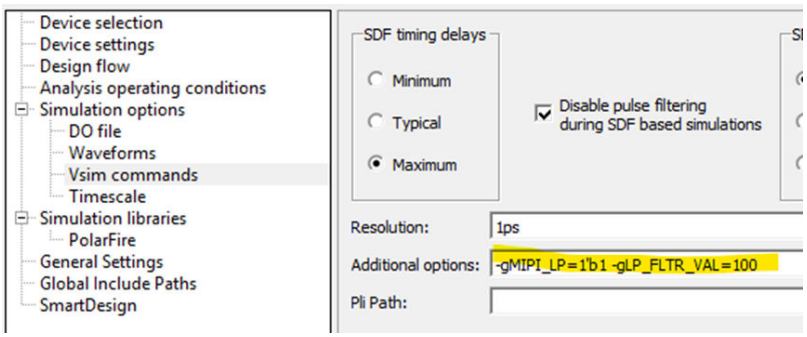
In LP operation, HS_DATA_SEL/HS_CLK_SEL de-assertion sends the LP_DATA and LP_CLK out to the TXLP/TX_CLK_LP MIPI pins while the HS_TX pair is disabled in a High-Z state.

The D-PHY transmit must be interfaced to the MIPI receiver using the terminated interface shown in Figure 7-10.

Important:

- During the simulation only, a false scenario is observed, resulting in the low-power (LP_DATA_N) signal toggling. This impacts the observed functionality of the MIPI RX decoder IP linked to the output of the IOD, during high-speed data. This does not occur in real device operation but is only observed in simulation. On silicon, in low-power mode, the common mode voltage of the differential I/O is lower than the trigger point of LVCMOS12. Therefore, the LP_DATA_N signal operates correctly and does not toggle in high-speed mode.
- During high-speed data transition, if the user does not want the toggling of L0_LP_DATA and L0_LP_DATA_N signals of the PF_IOD_GENERIC_RX IP, the user can add the `-gMIPI_LP=1'b1 -gLP_FLTR_VAL=100` Vsim command/arguments in **Libero SoC** > **Project Settings** > **Simulation options** > **Vsim commands** (see Figure 7-11) as shown in the following figure.

Figure 7-11. Vsim Command



Where,

- `MIPI_LP` is the 1'b1 parameter value to activate the filtering logic. The 1'b0 value retains the old functionality as it is. The default is 1'b0.
- `LP_FLTR_VAL` is the glitch pulse in nanoseconds to be filtered. The user can change it to any value as required. The default is 100 ns.

- In a design, if all IODs are not used for MIPI, individual IOD must be controlled by using `defparam` in the user testbench as shown in the following example:

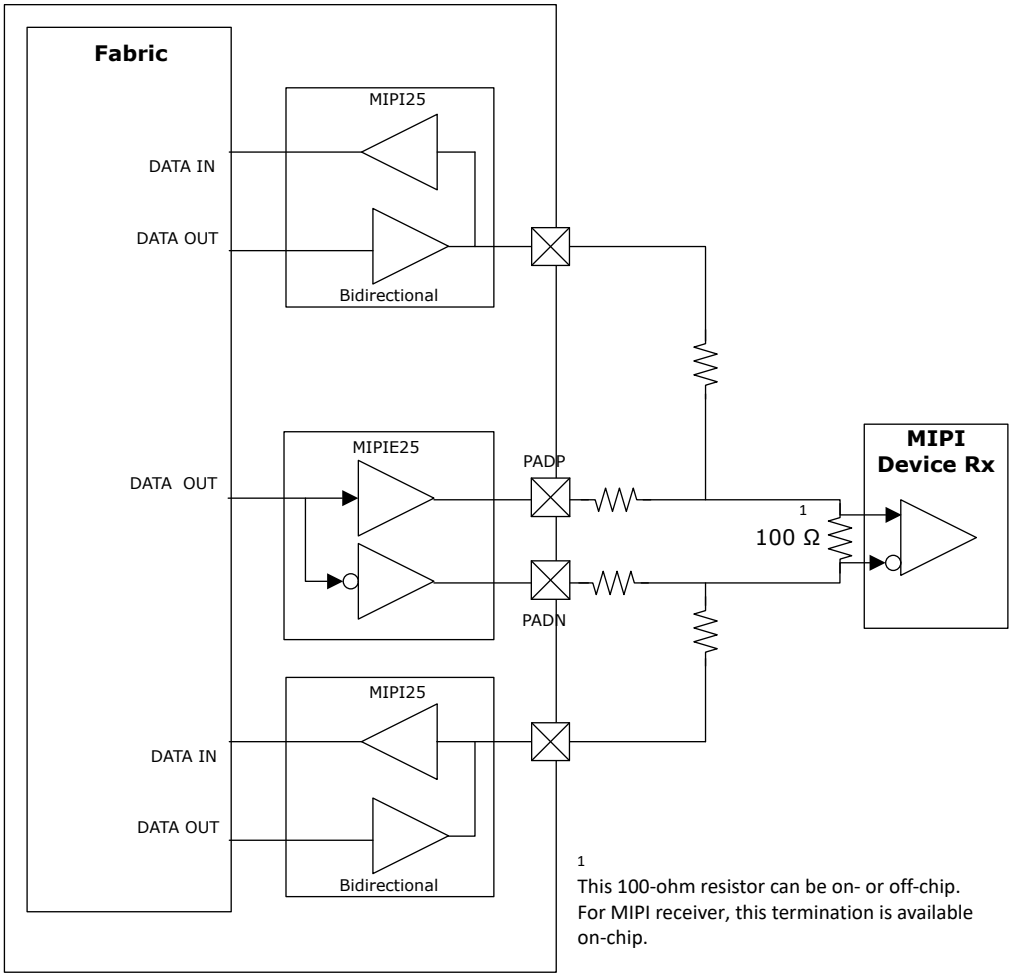
```
parameter MIPI_LP = 1'b1;
parameter LP_FLTR_VAL = 100;

// MIPI_LP for INBUF_DIFF_MIPI
defparam test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_INBUF_DIFF_MIPI_3.MIPI_LP
= MIPI_LP;
defparam test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_INBUF_DIFF_MIPI_2.MIPI_LP
= MIPI_LP;
defparam test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_INBUF_DIFF_MIPI_1.MIPI_LP
= MIPI_LP;
defparam test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_INBUF_DIFF_MIPI_0.MIPI_LP
= MIPI_LP;
// LP_FLTR_VAL for INBUF_DIFF_MIPI
defparam
test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_INBUF_DIFF_MIPI_3.LP_FLTR_VAL =
LP_FLTR_VAL;
defparam
test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_INBUF_DIFF_MIPI_2.LP_FLTR_VAL =
LP_FLTR_VAL;
defparam
test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_INBUF_DIFF_MIPI_1.LP_FLTR_VAL =
LP_FLTR_VAL;
defparam
test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_INBUF_DIFF_MIPI_0.LP_FLTR_VAL =
```

```
LP_FLTR_VAL;

// MIPI_LP for IOD
defparam test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_IOD_3.MIPI_LP = MIPI_LP;
defparam test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_IOD_2.MIPI_LP = MIPI_LP;
defparam test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_IOD_1.MIPI_LP = MIPI_LP;
defparam test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_IOD_0.MIPI_LP = MIPI_LP;

// LP_FLTR_VAL for IOD
defparam test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_IOD_3.LP_FLTR_VAL =
LP_FLTR_VAL;
defparam test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_IOD_2.LP_FLTR_VAL =
LP_FLTR_VAL;
defparam test_tb.top_0.rx_top_0.PF_IOD_GENERIC_RX_C0_0.PF_IOD_RX.I_IOD_1.LP_FLTR_VAL =
LP_FLTR_VAL;
```

7.2.5.4. MIPI D-PHY Transmit Interface (High-Speed Only) with Bidirectional Low-Power Mode (Ask a Question)

GPIO also supports a bidirectional MIPI D-PHY lane with external resistors, as shown in the following illustration. Microchip provides a macro that can be instantiated in the user design to implement the MIPI transmit interface (high-speed only) with bidirectional low-power mode, see Generic I/O Interfaces for more information.

Figure 7-12. High-Speed Transmit with Bidirectional



Important: For resistor specifications, see respective PolarFire FPGA Board Design User Guide, PolarFire SoC FPGA Board Design Guidelines User Guide, RT PolarFire FPGA Board Design User Guide, *RT PolarFire SoC FPGA Board Design Guidelines User Guide* (to be available in a future release).

Important: For information about implementation, see DG0807: PolarFire Imaging and Video Kit Demo Guide (MIPI CSI-2 Camera Sensor).

7.2.6. I/O States During Various Operational Modes (Ask a Question)

The state of an I/O at any given point in time depends on the operational mode of the device at that point. This section describes the I/O state during various operational modes so that you can design your boards accordingly.

7.2.6.1. Power-Up and Initialization (Ask a Question)

The following table lists the I/O states during power-up and initialization modes.

Table 7-18. I/O States during Power-Up and Initialization

Device State	I/O State
Power-up start/powering up	Tri-state I/O buffers are disabled. Output drivers are disabled (tri-stated). Receivers are disabled (input signals are not passed to the FPGA fabric). All terminations, PCI clamp diodes, and weak pull-up/down modes are off. All I/O bank power detectors and PVT controllers are disabled.
User mode	The buffer is programmed based on Libero SoC I/O settings. Data and output enable signals are based on user settings.

See the following references:

- For more information about I/O states, see respective PolarFire FPGA and PolarFire SoC FPGA Programming User Guide or RT PolarFire FPGA Programming User Guide.
- For more information about I/O settings for unused I/O pins, see PPAT spreadsheets.

7.2.6.2. Device Programming Modes (Ask a Question)

The following table lists the user I/O states during various programming modes. For more information about programming modes, see PolarFire FPGA and PolarFire SoC FPGA Programming User Guide or RT PolarFire FPGA Programming User Guide.

Table 7-19. GPIO and HSIO States During Programming Modes

Programming Modes	I/O States
JTAG	Set during JTAG programming in Libero SoC
SPI slave programming	Tri-state with weak pull-up/pull-down
IAP	Tri-state with weak pull-up/pull-down
Auto-programming	Tri-state
IAP recovery	Tri-state with weak pull-up/pull-down

7.2.7. Cold Sparing and Hot Swap (Ask a Question)

This section describes cold sparing and hot swapping capabilities of the user I/Os. For more information about cold sparing and hot socketing, see respective PolarFire FPGA Board Design User Guide, PolarFire SoC FPGA Board Design Guidelines User Guide, RT PolarFire FPGA Board

Design User Guide, or *RT PolarFire SoC Board Design Guidelines User Guide* (to be available in a future release).

7.2.7.1. Cold Sparing (Ask a Question)

In cold-sparing applications, voltage can be applied to device I/Os before and during power-up. For cold-sparing applications, the device must support the following characteristics:

- I/Os must be tri-stated before and during power-up
- Voltage applied to an I/O must not power up any part of the device
- Device reliability must not be compromised if voltage is applied to I/Os before or during power-up

Cold Sparing is supported with GPIO—any GPIO of an unpowered device can be safely driven with very minimal leakage current. When the device is powered OFF, both VDD and the VDDI are clamped to ground, preventing these supplies from powering up when a voltage is applied to the inputs. It is a good design practice not to rely on the outputs of an unpowered or partially powered device to drive other components in the system.

HSIO are pseudo-cold spare. It requires the spare device to have its HSIO VDDI banks powered-up to prevent I/O leakage through the ESD diodes. This is required to maintain low power and a protected state.

7.2.7.1.1. Hot Swap (Ask a Question)

Hot Swap allows a voltage to be applied to the inputs of devices before power is present on the VDDI pins. A pull-up clamp diode must not be present in the I/O circuitry to be hot swap. GPIO supports hot swap, but HSIO does not support hot swap.

When FPGA is not powered, GPIO is in a high-impedance state (hi-Z), also known as disabled state. For GPIO configured for I/O standards requiring a VREF, the amount of current flowing into or out must be minimized for the GPIO pin so that the external VREF signal is not affected.

7.2.8. I/O Glitches (Ask a Question)

I/O glitches can occur at power up or power down. The conditions that cause the glitches depend on the use of GPIO or HSIO in the system. The dependencies of VDD, VDDI, and VDDAUX to mitigate any glitches on the I/O interfaces are discussed in respective PolarFire FPGA Board Design User Guide or PolarFire SoC FPGA Board Design Guidelines User Guide.

7.2.9. I/O Calibration (Ask a Question)

HSIO and GPIO have a built-in I/O calibration feature per bank excluding bank 3. The I/O calibration circuitry is completely self-contained requiring no external reference resistors. The basis for calibration is to optimize the device performance to compensate for process, voltage and temperature (PVT) variations. The calibration controller is used to achieve impedance control for the GPIO and HSIO output buffer drive, termination and HSIO slew rate control by calibrating the I/O drivers. The calibration is initially completed at power-up. It is initiated by power-on detectors on VDDI and VDDAUX power supplies. The internal calibration engine initializes the I/O with internal approximation register settings at power-up. I/O calibration occurs automatically at power-up or by toggling DEVRST_N. On-demand calibration can be invoked by you after the initial I/O calibration. The PF_INIT_MONITOR FPGA IP is used to control the I/O recalibration or to monitor the initial I/O calibration. For more information about calibration requirements for proper start-up, initialization, re-calibration, and usage of the PF_INIT_MONITOR module, see PolarFire Family Power-Up and Resets User Guide.

The ODT and output drive features of HSIO and GPIO are calibrated depending on the I/O standard used in a Libero SoC design. The calibration logic is initially in a reset state at power-on. This initial pre-calibration state of the device sets the default to maximum calibration settings. This is done to the I/Os to ensure that the buffers are functionally operational after the power-on is complete.

The maximum settings are temporarily used by the buffers until the initial startup is completed. When this is completed, the optimized calibration values are then distributed to the associated I/Os within the bank. The calibration values are used for PVT compensation. GPIO and HSIO use the calibrated values for both drive strength and termination strength. The GPIO differential termination are also calibrated, and HSIO buffers are calibrated for output slew rate control.

HSIO and GPIO initially power-on with default maximum settings. Maximum pre-calibrated settings are defined as strong drive strength (low output impedance) and low termination values. Due to the nature of these initial pre-calibration settings, a transient current on the VDDI of the associated bank occurs during this pre-calibration phase. The transient current does not have long-term reliability concerns. The transient current diminishes when exiting the pre-calibration phase.

The initial transient current caused by pre-calibration can be mitigated if it is undesirable to the system. Transient current that is caused due to ODT termination can be managed by using the ODT control capabilities in the I/O (see On-Die Termination (ODT)). Training IP (TIP) normally associated with high-speed DDR interfaces can be used to disable the I/O termination until calibration is complete. For untrained termination interfaces, the ODT_DYN interface can be used to disable this pre-calibrated termination. I/O calibration/recalibration is not related to any IOD operations for source-synchronous interfaces or I/O CDR functionality. I/O calibration is limited to the analog optimization of the I/O buffer circuitry. IOD circuitry does not impact I/O calibration nor does I/O calibration impact IOD operation.

7.2.10. Dynamic ODT or Fail-Safe LVDS (Ask a Question)

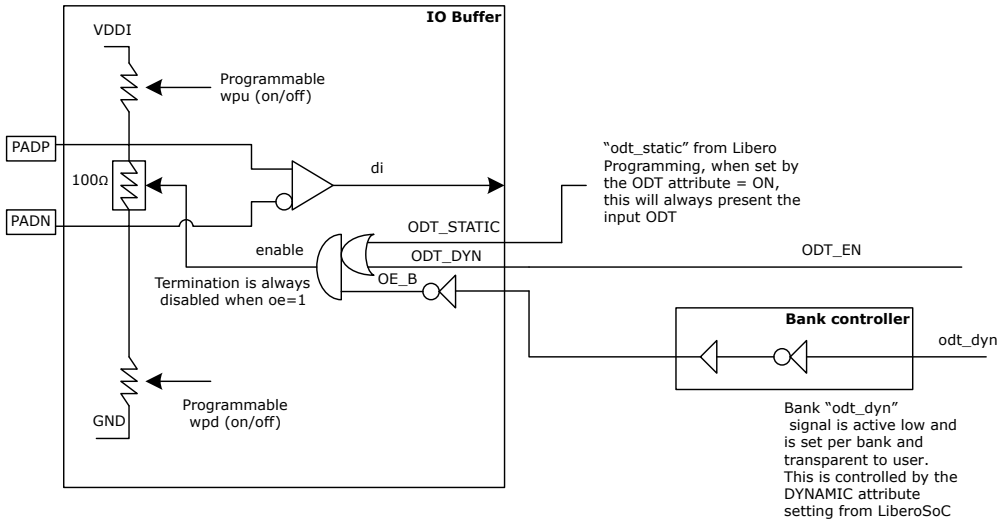
PolarFire family FPGAs support an internal LVDS fail-safe solution. This configuration uses a combination of the following device features:

- Dynamic on-die-termination (ODT) access per I/O
- Weak pull-up/pull-down resistor for differential inputs

When the LVDS input temporarily floats during operation, a bank-level input signal can dynamically turn-off the on-die termination resistor so that each leg of the LVDS pair can only see the weak pull-up and pull-down resistor enabled, creating an LVDS fail-safe input.

The ODT_EN pin can be exposed for any I/O that subscribes for DYNAMIC ODT required to be LVDS fail-safe. The user design uses ODT_DYN_PIN to switch in or switch out the differential termination while the weak pull-up resistor I/O attribute is added on PADP of the LVDS I/O and the PADN is weakly pulled down, automatically. The fail-safe condition has the ODT disabled leaving the pull resistors to differentially bias the PADP and PADN, preventing unwanted behavior when not being driven. During normal operation, the internal ODT must be present for the LVDS receiver.

Figure 7-13. Dynamic ODT used for Fail-safe LVDS



Important: During fail-safe, the weak pull-up and weak pull-down are on all the time. ODT_DYN_PIN is used to disable the differential termination when the pads are not driven externally.

I/O configurators that use LVDS input have an option to expose the **Enable ODT_EN pin for LVDS Failsafe** option. This is an active high input that must be driven from the fabric by the user.

Only GPIO has internal 100Ω ODT termination that can be dynamically controlled by a fabric pin. HSIO requires fixed, external termination resistor on PCB.

If the dynamic ODT option is enabled in the IO editor or with a PDC command, the following warning occurs during place and route:

```
Warning: Port 'user port name' has the 'On Die Termination' value of DYNAMIC
but the ODT_EN pin is connected to a power net on the IOD.
```

Important:

- Differential transceiver reference clock inputs include optional 100Ω differential termination. However, dynamic failsafe is not included. Designs must not allow for XCVR REFCLK P/N pads to float. Unused REFCLK pads must follow the recommendations in the public pin-out assignment tables (PPAT).
- There is a known issue in the Libero SoC I/O Editor and the pin report. A software limitation exists where a design cannot have different values for the P and N sides. Currently, both must have the same value. Libero SoC programs the P and N side correctly for programming. In IOEditor or pin report, if RES_PULL is up on both, it means the N side is programmed as down or visa-versa.

7.2.11. Dedicated I/O Pins (Ask a Question)

PolarFire family of devices have a dedicated bank of I/Os. These I/Os are used for JTAG, SPI, and dedicated functions and pins supporting from 3.3V to 1.8V (Nominal) operation using the dedicated VDDI3 bank. Single-ended CMOS/TTL receiver input and output drivers are fixed in the device as listed in the following table.

Table 7-20. Dedicated I/Os[1]—Fixed Settings

Dedicated I/O Signals[2]	Direction	Hot Swap	Pull Mode	Clamp	Hysteresis	Drive Strength
SCK	BIDI	Yes	OFF	OFF	ON	NA
SS	BIDI	Yes	OFF	OFF	ON	NA
SDO	OUT	Yes	OFF	OFF	OFF	8 mA
TDO	OUT	Yes	OFF	OFF	OFF	12 mA
TMS	IN	No	Pull-up	OFF	ON	NA
TDI	IN	No	Pull-up	OFF	ON	NA
TCK	IN	Yes	OFF	OFF	ON	NA
TRSTB	IN	No	Pull-up	OFF	ON	NA
DEVRST_N	IN	No	Pull-up	OFF	ON	NA
RESERVED	IN	No	Pull-up	OFF	ON	NA
SDI	IN	Yes	OFF	OFF	ON	NA
IO_CFG_INTF	IN	Yes	OFF	OFF	ON	NA
SPI_EN	IN	Yes	OFF	OFF	ON	NA

Notes:

1. Dedicated I/O pins are fixed function GPIO using the fixed settings as listed in the preceding table. For more information, see the "DC Input and Output" specifications in PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet.
2. These signals cannot be altered or programmed with Libero SoC.

The five dedicated inputs—TDI, TMS, TRSTB, DEVRSTB, and RESERVED—do not support hot swap. The SDI pin does not have an on-chip weak pull-up due to the following reasons:

- There may be multiple SPI interfaces connected on the board yielding too many parallel weak pull ups.
- If SPI is not used in the system, Microchip has defined how the pin must be connected using the PPAT file.

For more information, see PolarFire FPGA and PolarFire SoC FPGA Programming User Guide or RT PolarFire FPGA Programming User Guide.

7.2.12. Transceiver Receivers, Transmitters, and Reference Clock Inputs (Ask a Question)

For information about Transceiver (XCVR) input receivers and output transmitters, see PolarFire Family Transceiver User Guide.

Reference Clock (REF_CLK) Inputs dedicated to transceivers are similar to FPGA GPIO. However, some features and capabilities do differ. For information, see PolarFire Family Transceiver User Guide.

7.2.13. MSS Pins (For PolarFire SoC and RT PolarFire SoC FPGA Only) (Ask a Question)

PolarFire SoC and RT PolarFire SoC FPGA MSS has dedicated interfaces available. These pinouts are determined with the MSS Configurator. The interfaces have optional features that the user must determine to match the application. These interfaces include MSSIO, MSS_SGMII, and MSS_DDR.

7.2.13.1. MSSIO (Ask a Question)

There are 38 general purpose I/O pads—split over two banks within the MSS block referred to as MSSIO—to support the peripheral devices/standards. MSSIO supports different peripherals, such as SD, SDIO, eMMC, USB 2.0, I2C, MMUART, SPI and CAN standards and support for USB 2.0 OTG protocol. MSSIO supports the following features:

- 3.3V – 1.2V (Nominal) Operation
  - PCI/LVTTL/LVCMOS (3.3V)
  - LVCMOS (2.5V – 1.8V)
  - LVCMOS (1.5V and 1.2V)
- Single-ended CMOS/TTL output driver modes and receiver input modes
- Ratio Receiver with dynamically configurable ON/OFF hysteresis
- Dynamically configurable ON/OFF clamp
- Supports hot socket and cold spare
- Push-pull output driver
- Support for standard and fast operation for I2C only

The MSSIO are configured through the PolarFire SoC MSS configurator and are programmed when the MSS module is included in the Libero SoC project. The pin out information is found in the PolarFire SoC FPGA Package Pin Assignment Tables. In the MSS, MSSIOs are either in Bank 2 or Bank 4 and are used when the MSSIO peripherals are enabled. The associated VDDI for these banks must be connected to support the assigned peripheral.

For AC and DC characteristics of MSSIO, see the associated GPIO buffer specifications, that is, LVCMOS25 in the PolarFire SoC FPGA Datasheet.

The following tables lists the protocols supported by MSSIO bank.

Table 7-21. Protocols Supported by MSSIO Bank

Protocol	Mode	Data Rate (per IO)	Line-side Max Speed per IO in Kilo/Mega BITS per Sec - kbps/Mbps	Interface speed (width) in kilo/Mega BYTES per Sec - KBps/MBps	I/O Standard
USB 2.0	High	Single	60 Mbps	480 MBps (8-bit)	LVCMOS 3.3V
					LVCMOS 2.5V
					LVCMOS 1.8V
eMMC	Default speed	Single	26 Mbps	3.25 MBps (1-bit)	LVCMOS 3.3V
				13 MBps (4-bit)	LVCMOS 1.8V
				26 MBps (8-bit)	LVCMOS 1.2V
	High speed	Single	52 Mbps	6.5 MBps (1-bit)	LVCMOS 3.3V
				26 MBps (4-bit)	LVCMOS 1.8V
				52 MBps (8-bit)	LVCMOS 1.2V
	High speed DDR	Dual	104 Mbps	52 MBps (4-bit)	LVCMOS 3.3V
				104 MBps (8-bit)	LVCMOS 1.8V
					LVCMOS 1.2V
	HS200	Single	200 Mbps	100 MBps (4-bit)	LVCMOS 1.8V
				200 MBps (8-bit)	LVCMOS 1.2V
	HS400	Dual	400 Mbps	400 MBps (8-bit)	LVCMOS 1.8V
					LVCMOS 1.2V
	HS400-ES	Dual	400 Mbps	400 MBps (8-bit)	LVCMOS 1.8V
					LVCMOS 1.2V
SDIO	Low Speed	Single	400 kbps	200 KBps (4-bit)	LVCMOS 3.3V
	Full Speed	Single	25 Mbps	12.5 MBps (4-bit)	LVCMOS 3.3V

Table 7-21. Protocols Supported by MSSIO Bank (continued)

Protocol	Mode	Data Rate (per IO)	Line-side Max Speed per IO in Kilo/Mega BITS per Sec - kbps/Mbps	Interface speed (width) in kilo/Mega BYTES per Sec - KBps/MBps	I/O Standard
SD	Default speed	Single	25 Mbps	12.5 MBps (4-bit)	LVCMOS 3.3V
	High speed	Single	50 Mbps	25 MBps (4-bit)	LVCMOS 3.3V
	SDR12	Single	25 Mbps	12.5 MBps (4-bit)	LVCMOS 1.8V
	SDR25	Single	50 Mbps	25 MBps (4-bit)	LVCMOS 1.8V
	SDR50	Single	100 Mbps	50 MBps (4-bit)	LVCMOS 1.8V
	DDR50	Dual	100 Mbps	50 MBps (4-bit)	LVCMOS 1.8V
	SDR104	Single	208 Mbps	104 MBps (4-bit)	LVCMOS 1.8V
CAN	—	Single	125 kbs - 1Mbps	15.6 KBps - 125 KBps (1-bit)	LVCMOS 3.3V
QSPI	—	Single	41.6 Mbps	10.4 MBps (2-bit)	LVCMOS 3.3V
				20.8 MBps (4-bit)	LVCMOS 2.5V
					LVCMOS 1.8V
					LVCMOS 1.5V
					LVCMOS 1.2V
SPI	Master	Single	41.6 Mbps	5.2 MBps (1-bit)	LVCMOS 3.3V
					LVCMOS 2.5V
					LVCMOS 1.8V
					LVCMOS 1.5V
					LVCMOS 1.2V
	Slave	Single	150 Mbps	18.75 MBps (1-bit)	LVCMOS 3.3V
					LVCMOS 2.5V
					LVCMOS 1.8V
					LVCMOS 1.5V
					LVCMOS 1.2V
MMUART	—	Single	1 Mbps	125 KBps (1-bit)	LVCMOS 3.3V
					LVCMOS 2.5V
					LVCMOS 1.8V
I^2C	Standard	Single	100 kbps	12.5 KBps (1-bit)	LVCMOS 3.3V
					LVCMOS 1.8V
	Fast	Single	400 kbps	50 KBps (1-bit)	LVCMOS 3.3V
					LVCMOS 1.8V

Important: Output drive support based on the voltage and not any specific peripheral:

- LVCMOS33 supports 8, 12, 16, 20 mA
- LVCMOS25 supports 6, 8, 12, 16 mA
- LVCMOS18 supports 6, 8, 10, 12 mA
- LVCMOS12 supports 4, 6, 8 mA

7.2.13.2. SGMII I/O (Ask a Question)

SGMII I/O are provided for a PHY interface in Bank5 (MSS_SGMII). These pins are configured to the proper differential input and output standards for the dedicated interconnection to the Ethernet

MACs hosted by the MSS block. There are also two I/Os for an external reference clock source inputs (MSS_REFCLK).

The following table lists the MSS_SGMII Tx options.

Table 7-22. MSS_SGMII Tx Options

Tx I/O Standard	Drive (mA)	Resistor Pull Mode [1]	SOURCE_TERM	VDDI
LVDS25/33	6, 4, 3.5, 3	None, Up, Down	OFF, 100	2.5V, 3.3V
RSDS25/33	4, 3, 2, 1.5	None, Up, Down	OFF, 100	2.5V, 3.3V
MINILVDS25/33	6, 4, 3.5, 3	None, Up, Down	OFF, 100	2.5V, 3.3V
SUBLVDS25/33	3, 2, 1.5	None, Up, Down	OFF, 100	2.5V, 3.3V
PPDS25/33	4, 3, 2, 1.5	None, Up, Down	OFF, 100	2.5V, 3.3V
LCMDS25/33	6, 4, 3.5, 3	None, Up, Down	OFF, 100	2.5V, 3.3V

[1] Optional pull-up/pull-down resistors may interfere with signal integrity. Users must simulate LVDS communications if using pull-up/pull-down options.

The following table lists the MSS_SGMII Rx Options available in the MSS Configurator.

Table 7-23. MSS_SGMII Rx Options

Rx I/O Standard	Pull Mode	VCM_RANGE	ODT	VDDI
LVDS25/33	None, Up, Down	MID, LOW	OFF, 100	2.5V, 3.3V
RSDS25/33	None, Up, Down	MID, LOW	OFF, 100	2.5V, 3.3V
MINILVDS25/33	None, Up, Down	MID, LOW	OFF, 100	2.5V, 3.3V
SUBLVDS25/33	None, Up, Down	MID, LOW	OFF, 100	2.5V, 3.3V
PPDS25/33	None, Up, Down	MID, LOW	OFF, 100	2.5V, 3.3V
LCMDS25/33	None, Up, Down	MID, LOW	OFF, 100	2.5V, 3.3V

For AC and DC characteristics of MSS_SGMII, see the associated GPIO I/O Standard, for example, LVDS25 buffer specifications in the PolarFire SoC FPGA Datasheet.

7.2.13.3. DDR I/O (Ask a Question)

There are dedicated DDR I/Os used when the MSS DDR subsystem is configured in Bank 6. These pins are used when the MSS configurator selects support for DDR3(L), DDR4, LPDDR3, and LPDDR4 memory devices. The I/O standard is configured automatically dependent on the memory interface type selected by the MSS configurator. Other I/O features are also available to match key system characteristics. This bank also includes an MSS_DDR_VREF pin that is not used because an internally generated reference voltage is provided. See MSS DDR VREF (PolarFire SoC and RT PolarFire SoC Only) for more information. This MSS_DDR_VREF pin must be treated as UNUSED per the recommendation in the PPAT.

Table 7-24. MSS_DDR I/O Options

IO_TYPE	Drive Impedance (Ω)	ODT Termination (Ω)	Memory Standard	VDDI
Address/Command				
SSTL15I	34, 40, 24, 48	OFF	DDR3	1.5
SSTL135I	34, 40, 24, 48	OFF	DDR3L	1.35
HSTL12I	48, 34, 27, 60	OFF	LPDDR3/DDR4	1.2
HSTL12I	48, 40, 34, 60	OFF	LPDDR3/DDR4	1.2
POD12I	48, 40, 34, 60	OFF	DDR4	1.1
LVSTLII	48, 40, 34, 60	OFF	LPDDR4	1.1
DQ				
SSTL15I	34, 40, 24, 48	120, 60, 40, 30	DDR3	1.5
SSTL135I	34, 40, 24, 48	120, 60, 40, 30	DDR3L	1.35

Table 7-24. MSS_DDR I/O Options (continued)

IO_TYPE	Drive Impedance (Ω)	ODT Termination (Ω)	Memory Standard	VDDI
HSUL12I	48, 34, 27, 60	120, 60, 40, 30	LPDDR3	1.2
POD12I	48, 40, 34, 60	120, 60, 40, 30	DDR4	1.2
POD12I	48, 40, 34, 60	120, 60, 40, 30	DDR4	1.1
LVSTLII	48, 40, 34, 60	120, 60, 40, 30	LPDDR4	1.1
DDR_DQS				
SSTL15I	34, 40, 24, 48	120, 60, 40, 30	DDR3	1.5
SSTL135I	34, 40, 24, 48	120, 60, 40, 30	DDR3L	1.35
HSUL12I	48, 34, 27, 60	120, 60, 40, 30	LPDDR3	1.2
POD12I	48, 40, 34, 60	120, 60, 40, 30	DDR4	1.2
POD12I	48, 40, 34, 60	120, 60, 40, 30	DDR4	1.1
LVSTLII	48, 40, 34, 60	120, 60, 40, 30	LPDDR4	1.1

For AC and DC characteristics of MSS_DDR, see the associated HSIO I/O Standard, for example, SSTL15I specifications in the PolarFire SoC FPGA Datasheet.

All MSS annotated I/Os are bonded out to pins in all the PolarFire SoC device and package offerings. See the PolarFire SoC FPGA MSS Technical Reference Manual for in-depth information.

7.2.14. Unused I/O Pins (Ask a Question)

In application designs not using specific I/O pins, the unused programming of these I/O pins are managed by Libero SoC. Designers can refer to the PPAT spreadsheets to determine the default programming of unused I/O by Libero SoC. Other recommendations for unused pins such as power pins and special function pins are defined in the respective PolarFire FPGA Board Design User Guide or PolarFire SoC FPGA Board Design Guidelines User Guide.

7.3. I/O Initialization (Ask a Question)

High-speed I/O have many stages of initialization that are involved in optimizing I/O performance. These stages include the following:

- I/O calibration
- I/O training
- Memory controller calibration or training

These phases of I/O initialization and the underlying timing paths are very critical in guaranteeing optimized performance of the high-speed I/O interfaces. A general understanding is required regarding the methodologies used for analyzing system timing.

7.3.1. I/O Calibration (Ask a Question)

HSIO, GPIO, MSSIO and MSSDDR have a built-in I/O calibration feature per bank excluding Bank 3. The I/O calibration circuitry is completely self-contained requiring no external reference resistors. The basis for I/O calibration is to optimize the device performance to compensate for Process, Voltage, and Temperature (PVT) variations. An embedded I/O calibration controller is used to achieve the impedance control for the I/O output buffer drive, termination, and slew rate control by calibrating the I/O drivers. The calibration is initially completed at power-up. Each bank is calibrated at power-up after the bank VDDI/VDDAUX voltages reach minimum thresholds and the bank "Auto calibration ramp time" has expired.

The internal calibration engine initializes the I/O with internal approximation register settings at power-up. I/O calibration occurs automatically at power-up or by toggling DEVRST_N. On-demand calibration can be invoked by user control after the initial I/O calibration. The PF_INIT_MONITOR FPGA IP is used to control the I/O recalibration or to monitor the initial I/O calibration and trigger on-demand recalibration.

Although the I/O is functional early in the power-up sequence, the high-speed I/O performance cannot be expected until it is completely calibrated. For more information about the calibration requirements for proper start-up, initialization, re-calibration, and usage of the PF_INIT_MONITOR module, see PolarFire Family Power-Up and Resets User Guide.

The ODT and the output drive features of HSIO, MSS DDR, MSSIO, and GPIO are calibrated specifically to the I/O standard targeted in the project. The calibration logic is initially in a reset state at power-on. This initial pre-calibration state of the device sets the default to maximum calibration settings. This is done to the I/Os in order to ensure that the buffers are operational after the power-on is complete.

The maximum settings are temporarily used by the buffers until the initial start-up is completed. When this is completed, the optimized calibration values are then distributed to the associated I/Os within the bank. The initial pre-calibrated settings are defined as strong drive strength (low output impedance) and low termination values. Due to the nature of these initial pre-calibration settings, a transient current on the VDDI of the associated bank occurs during this pre-calibration phase. The transient current does not have long-term reliability concerns. The transient current diminishes when exiting the pre-calibration phase. The calibration values are used for PVT compensation for both drive strength and termination strength. The GPIO differential termination is also calibrated, and HSIO buffers are calibrated for output slew rate control.

The initial transient current caused by pre-calibration can be mitigated if it is undesirable to the system. Transient current that is caused due to ODT termination is managed by utilizing the ODT control capabilities in the I/O (see On-Die Termination (ODT)). Training IP (TIP) associated with the high-speed DDR interfaces is used to disable the I/O termination until calibration is complete. For untrained termination interfaces, the ODT_DYN interface is used to disable this pre-calibrated termination. I/O calibration or recalibration is not related to any IOD operations for the source-synchronous interfaces or I/O CDR functionality. I/O calibration is limited to the analog optimization of the I/O buffer circuitry. IOD circuitry does not impact I/O calibration nor does I/O calibration impact IOD operation.

7.3.2. I/O Training (Ask a Question)

The training logic manages the timing exchanges between the IOD and bit alignment IP (or DDR controller modules). I/O training is related to the optimization of source-synchronous timing paths into and out of the device. This stage of training uses elements in the IOD block to perform DYNAMIC or real-time optimization, which analyzes and tunes the required changes in device performance based on device process, temperature, and voltages variations.

IO training utilizes the FPGA hosted logic to handshake to the IOD block (see Dynamic Delay Control). The handshaking optimizes the delay elements within the I/O data and clock paths to appropriately adjust the timing relationships to optimize performance.

For non-memory controller interfaces, see CoreRxIODBitAlign for information about the usage of fabric logic to train the IOD interface. For more information, see Dynamic IOD Interface Training or see the Training Logic chapter of PolarFire Family Memory Controller User Guide.

7.3.3. Memory Controller Training (Ask a Question)

This training is specific to the FPGA memory controller interface synchronization to the external memory devices. This phase of calibration is specific to optimizing the bandwidth performance between an FPA hosted memory controller and an external DRAM memory. This is separate from the FPGA device I/O calibration. The memory calibration of the memory interface paths is controlled and managed by memory controller IP, which uses the DYNAMIC I/O training elements and the device I/O calibration. The FPGA device completes the device I/O calibration on power-up before moving into the memory calibration phase.

The memory controller IP is responsible for calibration of write-leveling chains and programmable output delay chain to align the DQS edge with the CK edge at memory to meet the tDQSS, tDSS, and

tDSH specifications. For more information about memory calibration operations, see the Training Logic chapter of PolarFire Family Memory Controller User Guide.

8. IOD Features and User Modes (Ask a Question)

Each I/O (both GPIO and HSIO) has a digital block, called IOD, that interfaces with the FPGA fabric on one side and the IOA buffers on the other side (Figure 8-1). The IOD block includes several digital features, including I/O digital. The I/O digital allows for easy data transfer between the high-speed IOA buffers and the lower-speed FPGA core.

The IOD block can be configured for both input and output SDR and DDR modes. It also allows the gearing-up of the output data rate and gearing-down of the input data rate. These options are configured in Libero SoC and are used to build source synchronous I/O interfaces such as DDR and QDR memory controllers, common interfaces such as RGMII, MIPI D-PHY, 7:1 Video LVDS, and several other non-memory user interfaces.

This chapter provides information about the IOD block and the various I/O user modes, including various SDR, DDR, and digital modes.

8.1. IOD Block Features (Ask a Question)

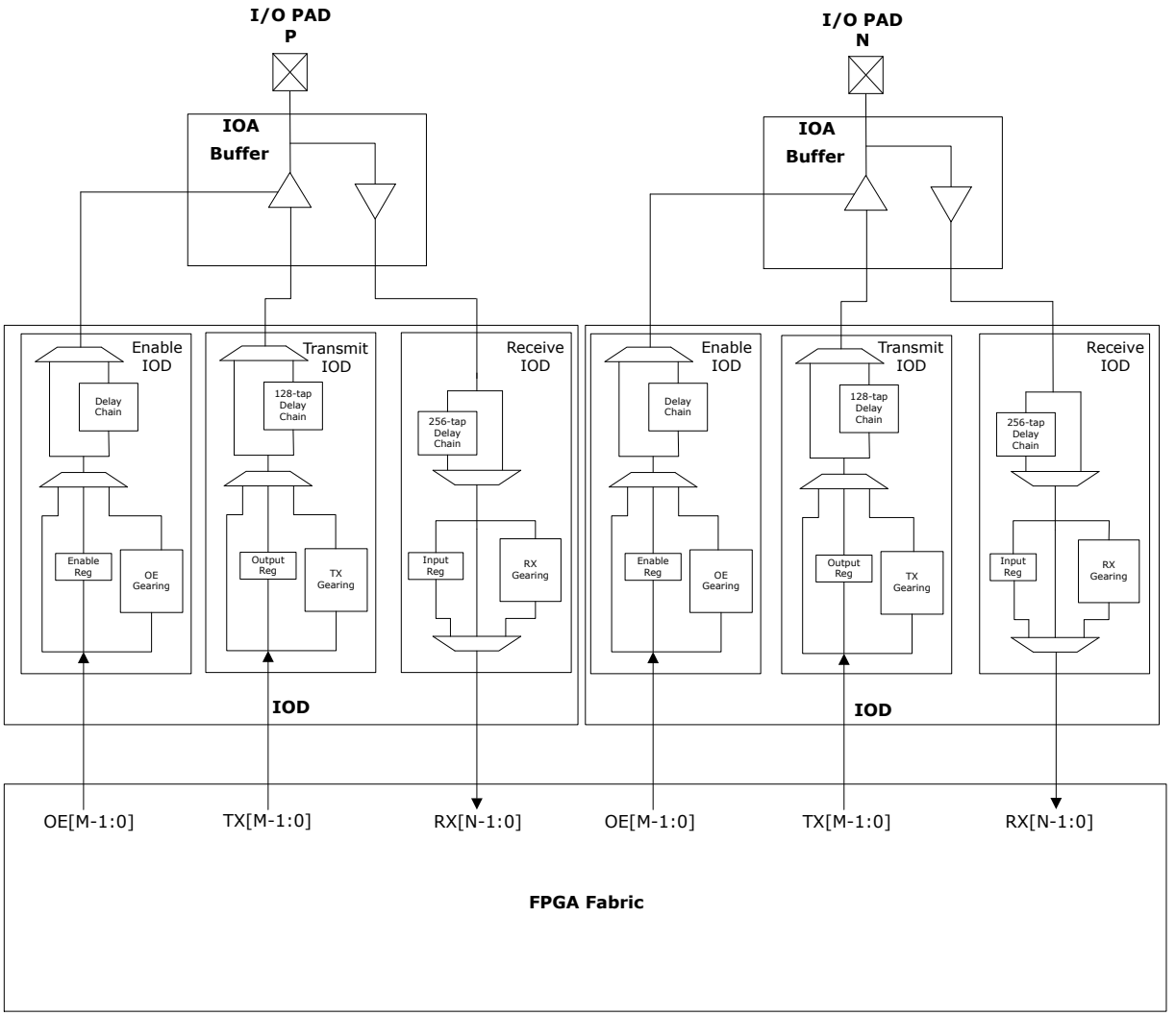
- Programmable input and/or output delay chain
- I/O register for data-in, data-out, and output enable signals
- Up to 1:10 input deserialization (input digital)
- Up to 10:1 output serialization (output digital)
- Support for DDR and SDR interfaces
- Word alignment with a slip control
- High-speed and low-skew I/O clock networks
- Clock recovery for serial protocols and other similar interfaces
- Low-power mode support to latch state of input or output data

8.2. IOD Block Overview (Ask a Question)

The IOD block includes the input and output delay functions, I/O registers, and digital logic blocks. The digital logic blocks are receive digital (Rx digital) for input, transmit digital (Tx digital) for output, and enable digital (OE digital) for the enable signals. The IOD block also includes several high-speed, low-skew clock networks. Figure 8-1 shows an overview of the IOD block. Various I/O features are set mainly by the protocol configurator or the Libero SoC configurator within Libero SoC. However, some of the I/O features such as I/O register and programmable delay can be controlled automatically or manually by Libero SoC.

The following illustration shows an overview of the IOD block.

Figure 8-1. IOD Configured for I/O Registers



Note: The values of M and N depend on the digital ratio. Enable path delay is not customizable by the user design.

8.2.1. Programmable I/O Delay (Ask a Question)

The IOD block includes process, voltage, programmable delay chains for both input and output data paths. The input delay path has an intrinsic delay when the delay chain is enabled. This added delay is above the value of the incremental tap delay and is reported by the Libero SoC software when used. Consequently, there is a fast path to the fabric when the input delay chain is not present. The programmable delay chains on the output data path allow 128 tap delay. The enable path also includes a 8-tap programmable delay chain that is programmed statically by software and is not user programmed. The programmable delay chain can be set statically by using the I/O attribute editor or by using a PDC command in Libero SoC. The value per tap delay is not process, voltage, temperature(PVT) compensated and can have variation. For information about delays, see respective PolarFire FPGA Datasheet or PolarFire SoC FPGA Datasheet.

The programmable delay chain is used to:

- Ensure zero hold time for the input registers

- Cancel the skew between the input data path and clock injection path
- Spread out I/O buffer timing along with an edge of the device for SSO noise control

The programmable delay chain can also be controlled through dynamic control signals from the FPGA fabric. Dynamic delay control is useful for high-speed interfaces that require per-bit alignment. The dynamic control is only available for certain I/O interfaces, see Generic I/O Interfaces for more information. Static delay values can be controlled by PDC command constraint through IO Editor or manual constraint file input. In the PDC constraint file, IN_DELAY allows settings from OFF, 0-127, 128-254 (even numbers only).

example:

```
set_io -port_name PAD \
-IN_DELAY 2 \
-DIRECTION INPUT
```

The output delay values can be controlled by PDC command constraint through IO Editor or manual constraint file input. In the PDC constraint file, OUT_DELAY allows settings from OFF, 1 - 128.

```
set_io -port_name PAD_0 \
-OUT_DELAY 2 \
-DIRECTION OUTPUT
```

8.2.1.1. Static Timing Analysis (Ask a Question)

Static delays are automatically prescribed by the IOD configurator. The values that are added based on the IOD configuration can be found in the `boardlayout.xml` report shown in the following figure. These are the initial values set by the software based on initial IOD setup information. The settings can be modified as mentioned in the preceding section with pdc or IOEditor. You can adjust the delay values by adding or subtracting from the initial value applied in the Libero SoC configurations. The per tap incremental delay value is found in the respective PolarFire FPGA Datasheet or PolarFire SoC Datasheet.

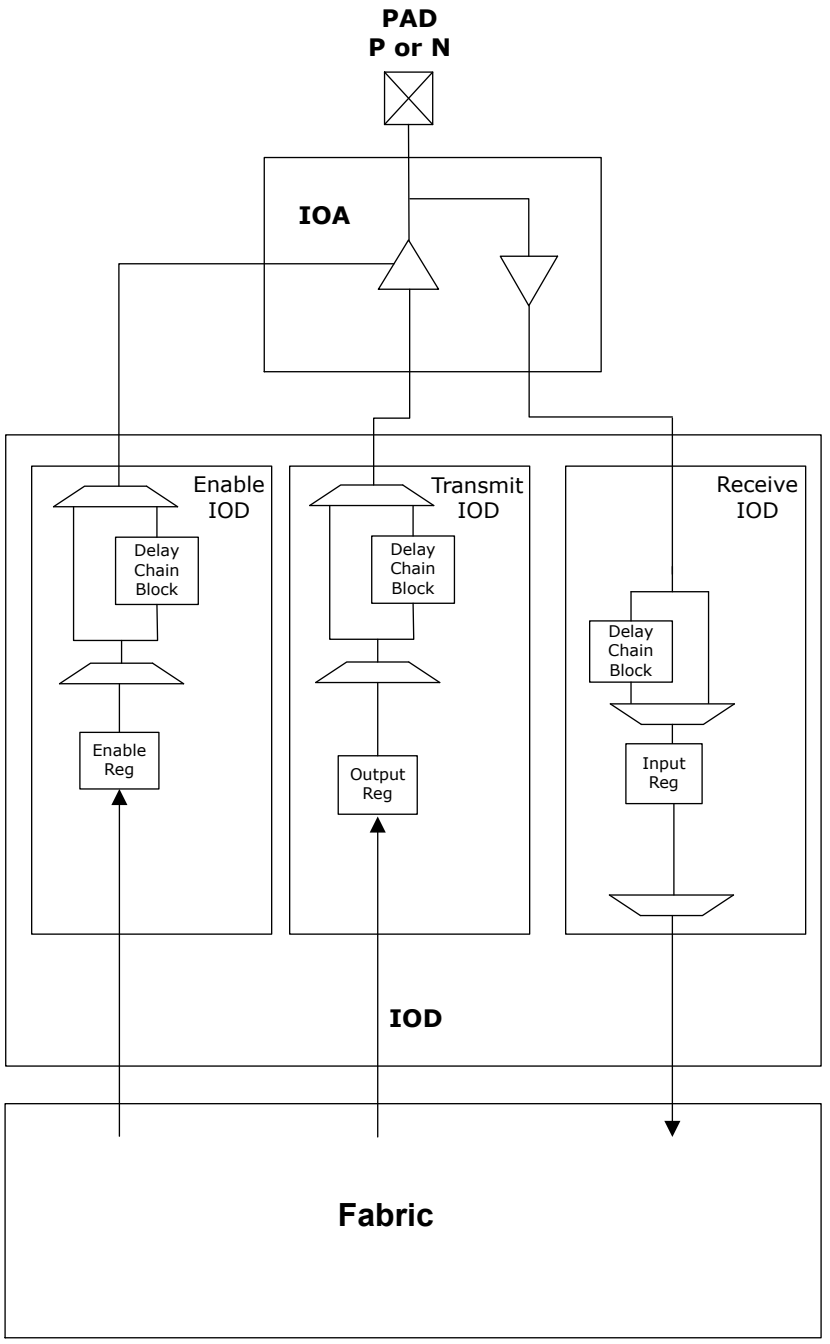
Figure 8-2. IOD Input Delay Example

Pin	Port	Function	Bank	State	Use I/O Reg	I/O Std	Direction	User I/O Lock Down	Clamp Diode	Resistor Pull	Schmitt Trigger	Vcm Input Range	On Die Termination	Odt Value (Ohm)	Input Delay	Slew	Output Drive (mA)	Impedance (ohm)	Output Load (pF)	Source Termination (Ohm)	Output Delay	Board Layout
AA2	adc_rxd_n_i[10]	HSIO91NB0	Bank0	Fixed	No	LVDS18	Input	No	ON	None	OFF	MID	---	---	49	---	---	---	---	---	---	adc_rxd_n_i[10]
AA3	adc_rxd_p_i[11]	HSIO90PB0	Bank0	Fixed	No	LVDS18	Input	No	ON	None	OFF	MID	---	---	49	---	---	---	---	---	---	adc_rxd_p_i[11]
AA4	adc_rxd_p_i[12]	HSIO95PB0	Bank0	Fixed	No	LVDS18	Input	No	ON	None	OFF	MID	---	---	49	---	---	---	---	---	---	adc_rxd_p_i[12]
AA5	adc_rxd_n_i[12]	HSIO95NB0	Bank0	Fixed	No	LVDS18	Input	No	ON	None	OFF	MID	---	---	49	---	---	---	---	---	---	adc_rxd_n_i[12]

8.2.2. I/O Registers (Ask a Question)

The IOD block includes registers for data-in, data-out, and output enable signals. The input registers (IOINFF) provide the registered version of the input signals from the IOA to the FPGA fabric. The output registers (IOUTFF) provide the registered version of the output signals from the FPGA fabric to the IOA. The output enable register (IOENFF) acts as a control signal for the output if the I/O is configured as tri-stated or bidirectional. The following figure shows the I/O registers. These registers in IOD blocks are similar to the D-type flip-flops available in fabric logic elements. The IOD blocks contain several macros that cannot be instantiated. The macros are included in the place and route software.

Figure 8-3. I/O Registers in IOD



The I/O register is used for:

- Better I/O interface performance, as the registers are placed close to the I/O pads.
- Synchronizing the transmit and receive bus signals. For example, the I/O registers ensure that all the bits of the bus are synchronized to the clock signal when they are transmitted or received.

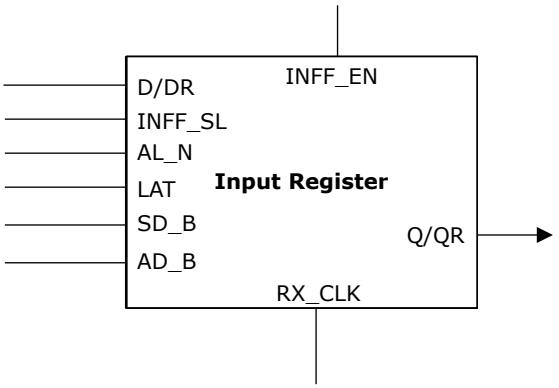
The I/O registers are used by default during place and route if the register can be mapped to the I/O register.

The PF_IO macro must be used to configure I/O registers. The configurator constructs registered I/O blocks that can be instantiated. See Basic I/O Configurator - PF_IO.

8.2.2.1. Input Register (Ask a Question)

The following illustration shows the input register.

Figure 8-4. Input Register



The following table lists the input register pins and descriptions.

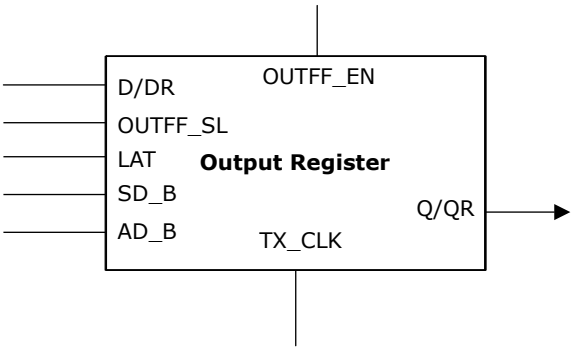
Table 8-1. I/O Input Register Ports

Ports	Types	Descriptions
RX_CLK	Input	Receive Clock input
D/DR	Input	Data input
Q/QR	Output	Data output
INFF_EN	Input	Clock enable (active high)
INFF_SL	Input	Synchronous load (active high)
AL_N	Input	Active low asynchronous load (active low)
LAT	Input	Latch enable (active high)
SD_B	Input	Synchronous data
AD_B	Input	Asynchronous data (active low)

8.2.2.2. Output Register (Ask a Question)

The following illustration shows the output register.

Figure 8-5. Output Register



The following table lists the output register pins and their descriptions.

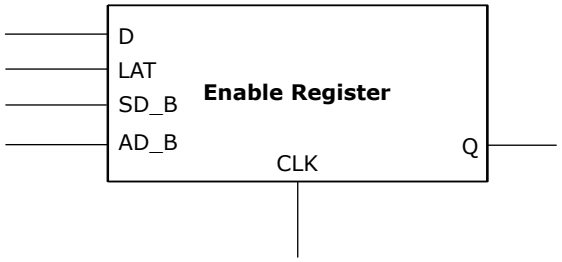
Table 8-2. I/O Output Register Ports

Ports	Types	Descriptions
TX_CLK	Input	Clock input
D/DR	Input	Data input
Q/QR	Output	Data output
OUTFF_EN	Input	Clock enable (active high)
OUTFF_SL	Input	Synchronous load (active high)
LAT	Input	Latch enable (active high)
SD_B	Input	Synchronous data
AD_B	Input	Asynchronous data (active low)

8.2.2.3. Enable Register (Ask a Question)

The following illustration shows enable register.

Figure 8-6. Enable Register



The following table lists the enable register pins and their descriptions.

Table 8-3. I/O Register Ports

Ports	Types	Descriptions
CLK	Input	Clock input
D	Input	Data input
Q	Output	Data output
LAT	Input	Latch enable (active high)
SD_B	Input	Synchronous data
AD_B	Input	Asynchronous data (active low)

8.2.2.4. I/O Register Combining (Ask a Question)

I/O register combining is supported on enable, input, and output of any I/O. This support is available using the set_i off command, which is included in a Compile Netlist Constraint (*.ndc) file and passed to the Libero SoC Compile engine for netlist optimization after synthesis.

Syntax:

```
set_ioff -port_name {portname} \
-IN_REG true/1|false/0 \

-OUT_REG true/1|false/0 \

-EN_REG true/1|false/0
```

Arguments

- <portname>: specifies the name of the I/O port to be combined with a register. The port can be an input, output, or in-out port.

- IN_REG: specifies whether the input register is combined into the port <portname>.
- OUT_REG: specifies whether the output register is combined into the port
- EN_REG: specifies whether the enable register is combined into the port <portname>.

I/O register combining is only permitted with one FF with an I/O. The FF needs to be connected to the I/O with a fanout of one. A bidirectional I/O where both D and Y pins are driven with registered signals can only allow one of the registers to be moved into the I/O pad.

There is another option to allow automatic I/O register combining. This option is enabled from the **Place and Route** configuration settings. Right-click **Place and Route** in the project navigator and select the **I/O Register Combining** checkbox. Enable this option to combine any register directly connected to an I/O when it has a timing Constraint. If there are multiple registers directly connected to a (bi-directional) I/O, select one register to combine in the following order: input-data, output-data, output-enable. Users can use the NDC constraint for more tightly controlling the use of I/O register combining.

Important: This feature is OFF by default. Users must turn it ON to enable combining.

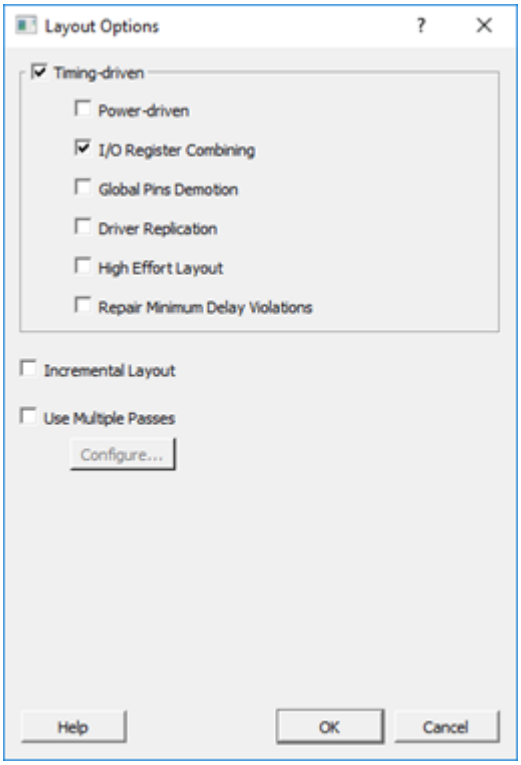
Every I/O has several embedded registers that you can use for faster clock-to-out timing, and external hold and setup. When combining these registers at the I/O buffer, some design rules must be met.

This feature is supported by all I/O standards.

Following are the rules to combining the I/O registers:

- You can combine only one register with an I/O IN, OUT, or EN.
- An input register cannot be combined to different I/Os.
- For input registers (INFF), the Y pin of an I/O needs to drive the D pin of a register with fanout of 1.
- For output registers (OUTFF), the Q pin of a register needs to drive the D pin of an I/O with fanout of 1.
- For enable registers (ENFF), the Q pin of a register needs to drive the E pin of an I/O with fanout of 1.

Figure 8-7. I/O Register Combining from Place and Route Layout Options

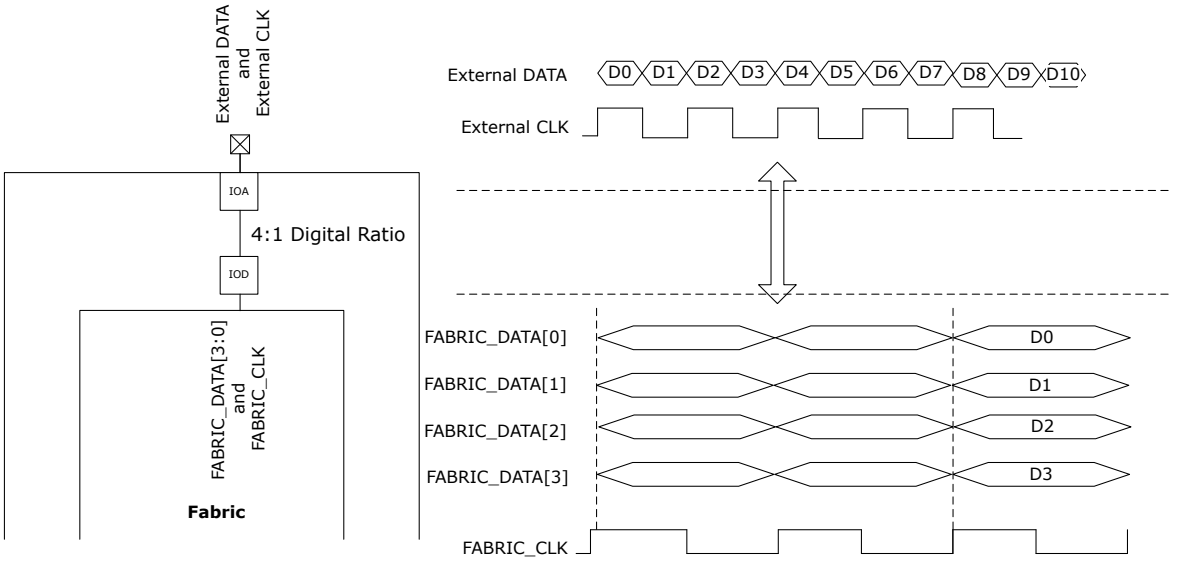


8.2.3. I/O Gearing (Ask a Question)

I/O gearing handles serial-to-parallel and parallel-to-serial conversion of multiple FPGA fabric signals to and from a single device I/O based on user clock settings, as shown in the following illustration. The gearbox either deserializes and transfers input data to a lower core clock speed, or transfers lower-speed data from the fabric to the high-speed output clock domain, and serializes it in the process. Libero SoC automatically configures these gearboxes based on the application settings. Generic IOD interfaces provide a complete solution from the I/O pins to the fabric. Generic IOD is supported by construction using Libero SoC configurators and limited to the defined list of use cases. See Generic I/O Interfaces for available support.

The following illustration shows the I/O gearing example, where high-speed serial data is passed from I/O to fabric through four signals at lower speed.

Figure 8-8. I/O Digital



8.2.4. I/O FIFO (Ask a Question)

The IOD block contains an I/O FIFO for phase compensation clock domain transfers. In DDR applications, the I/O FIFO is used for high-speed transfer data from the external DQS domain to the internal data clock domain. Libero SoC and Microchip memory controller cores configure the I/O FIFO based on the application settings.

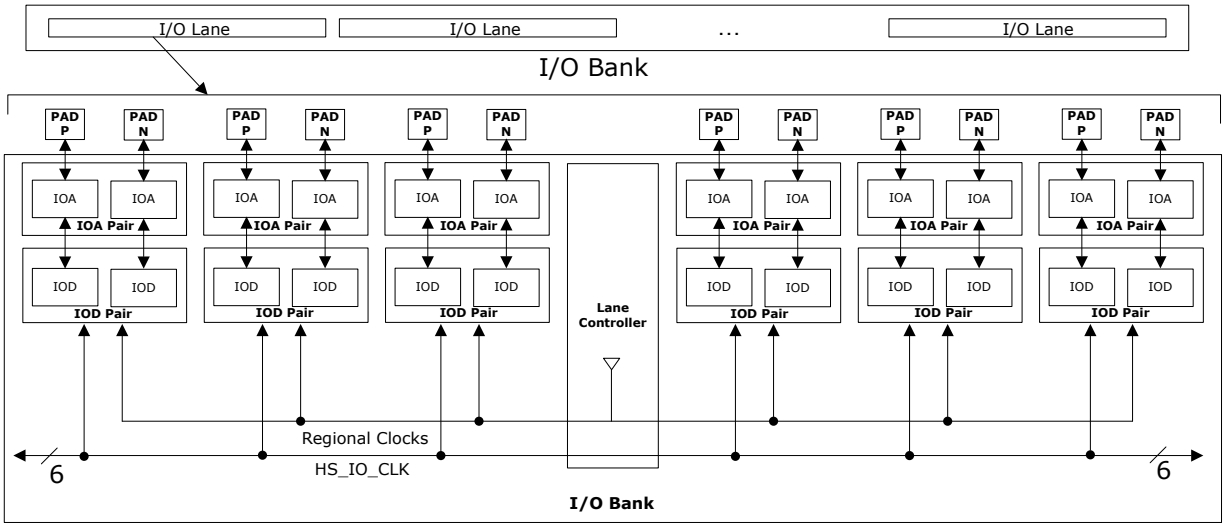
8.3. I/O Lanes (Ask a Question)

To support memory interfaces, I/O pairs are grouped into lanes, with multiple lanes per bank. Each lane consists of twelve I/Os (six I/O pairs), a lane controller, and a set of high-speed, low-skew clock resources. The uppermost lane on the western side of devices has less than six I/O pairs in each lane. The high-speed and low-skew clock resources in the I/O lane include a global clock network, regional clock networks, high-speed clock networks, and lane controller clock networks, see PolarFire Family Clocking Resources User Guide for more information.

The I/O lane is used for easy implementation of integrated PHY for memory. For example, a 32-bit SDRAM interface requires four I/O data lanes. Each data lane uses one I/O lane—two I/O pads are used for DQS, eight I/O pads are used for DQ bits, one pad is used for data mask (DM), and one I/O pad is used as a spare. The lane topology is also used to construct generic I/O interfaces, which requires high-speed and low-skew clocking.

The following illustration shows the I/O lanes diagram.

Figure 8-9. I/O Lanes



- **Global Clock Network**—is used to distribute high fan-out signals such as clocks and resets across the FPGA fabric with low-skew.
- **Regional Clock Networks**—are low-latency networks that distribute clocks only to a specific designated area based on the driving source. Regional clock networks are used to move data in and out of the fabric.
- **High-Speed I/O Clock Networks**—are used to distribute high-speed clocks along the edge of the device to service the I/Os. High-speed I/O clock networks are used to implement high-speed interfaces.

Regional and Global I/O clock performance varies around the periphery of the device. The Regional Clock maximum frequency is slower than the Global I/O clock. This is inherent to device design as the regional clock is meant to be utilized in close proximity to its source.

8.3.1. Lane Controller (Ask a Question)

The lane controller handles the complex operations necessary for the high-speed interfaces, such as DDR memory interfaces and CDR interfaces. To bridge the lane clock to the high-speed I/O clock, the lane controller is used to control an I/O FIFO in each IOD. This I/O FIFO interfaces with DDR memory by utilizing the DQS strobe on the lane clock. The lane controller can also delay the lane clock using a PVT-calculated delay code from the DLL to provide a 90° shift. Certain I/O interfaces require a lane controller to handle the clock-domain that results with higher gear ratios. For more information, see Generic I/O Interfaces.

The lane controller also provides the functionality for the IOD CDR. Using the four phases from the CCC PLL, the lane controller creates eight phases and selects the proper phase for the current input condition with the input data, see PF_IOD_CDR for more information. A divided-down version of the recovered clock is provided to the fabric (DIVCLK).

8.3.2. I/O Lanes in Each Bank (Ask a Question)

The following tables list the number of I/Os and lanes in each bank for each device and package option.

Table 8-4. PolarFire FPGA I/O Lanes in Each Bank

Devices	Packages	North Corner I/Os						South Corner I/Os					West Corner I/Os			
		Bank 0		Bank 7		Bank 1		Bank 3	Bank 2		Bank 6		Bank 4		Bank 5	
		HSIO	Lanes	HSIO	Lanes	HSIO	Lanes	JTAG	GPIO	Lanes	HSIO	Lanes	GPIO	Lanes	GPIO	Lanes
MPF100	FCSG325	36	3	0	0	48	4	13	48	4	0	0	38	3	0	0
MPF200	FCSG325_14.5x11	36	3	0	0	48	4	13	48	4	0	0	38	3	0	0
MPF200	FCSG536	60	5	0	0	60	5	13	96	8	0	0	84	7	0	0
MPF300	FCSG536	60	5	0	0	60	5	13	96	8	0	0	84	7	0	0
MPF100	FCVG484	60	5	0	0	60	5	13	96	8	0	0	68	5	0	0
MPF200	FCVG484	60	5	0	0	60	5	13	96	8	0	0	68	5	0	0
MPF300	FCVG484	60	5	0	0	60	5	13	96	8	0	0	68	5	0	0
MPF100	FCG484	48	4	0	0	48	4	13	84	7	0	0	64	5	0	0
MPF200	FCG484	48	4	0	0	48	4	13	84	7	0	0	64	5	0	0
MPF300	FCG484	48	4	0	0	48	4	13	84	7	0	0	64	5	0	0
MPF200	FCG784	72	6	24	2	60	5	13	96	8	0	0	92	7	44	3
MPF300	FCG784	72	6	24	2	60	5	13	96	8	0	0	92	7	44	3
MPF500	FCG784	72	6	24	2	60	5	13	96	8	0	0	92	7	44	3
MPF300	FCG1152	72	6	72	6	60	5	13	96	8	72	6	92	7	48	4
MPF500	FCG1152	72	6	96	8	60	5	13	96	8	96	8	92	7	72	6

Table 8-5. PolarFire SoC FPGA I/O Lanes in Each Bank

Devices	Packages	North Corner I/Os					South Corner I/Os				West Corner I/Os					
		Bank 6	Bank 8		Bank 0		Bank 3	Bank 1		Bank 9	Bank 4	Bank 2	Bank 5	Bank 7		Bank 6
		MSS HSIO	HSIO	Lanes	HSIO	Lanes	DEDIO	GPIO	Lanes	GPIO	Lanes	MSSIO	MSSIO	SGMII	GPIO	Lanes	MSS HSIO
MPFS025	FCS325_11x11	22	0	0	32	2	13	48	4	0	0	14	24	10	0	0	32
MPFS095	FCS325_14.5x11	22	0	0	32	2	13	48	4	0	0	14	24	10	0	0	32
MPFS095	FCS536	44	0	0	60	5	13	84	7	0	0	14	24	10	24	2	44
MPFS160	FCS536	44	0	0	60	5	13	84	7	0	0	14	24	10	24	2	44
MPFS250	FCS536	44	0	0	60	5	13	84	7	0	0	14	24	10	24	2	44
MPFS025	FCV484	44	0	0	60	5	13	48	4	0	0	14	24	10	0	0	44
MPFS095	FCV484	44	0	0	60	5	13	84	7	0	0	14	24	10	0	0	44
MPFS160	FCV484	44	0	0	60	5	13	84	7	0	0	14	24	10	0	0	44
MPFS250	FCV484	44	0	0	60	5	13	84	7	0	0	14	24	10	0	0	44
MPFS095	FCV784	44	60	5	84	7	13	72	6	60	5	14	24	10	0	0	44
MPFS160	FCV784	44	60	5	84	7	13	72	6	72	6	14	24	10	24	2	44
MPFS250	FCV784	44	60	5	84	7	13	72	6	84	7	14	24	10	24	2	44
MPFS250	FC1152	44	60	5	84	7	13	72	6	96	8	14	24	10	60	5	44
MPFS460	FC1152	44	60	5	120	10	13	72	6	120	10	14	24	10	96	8	44
MPFS460	CG1509	44	60	5	120	10	13	72	6	132	11	14	24	10	132	11	44

Important: Connectivity restrictions apply to the lanes listed as follows with regard to IOCDR and any IOD generic Rx interfaces using regional clock. This also implies a design cannot migrate from the MPF300, which has complete regional clock connectivity to the other devices with the listed impacted lanes.

The impacted lanes are as follows (as documented in the associated Package Pin Assignment Table (PPAT)): MPF100, MPF200: DDR_S_3 (Bank 2, Lane 3)
MPF500: DDR_S_6 (Bank 2, Lane 6), DDR_N_9 (Bank 7, Lane 9)

Full duplex 1GbE and SGMII IOCDR are supported in the GPIO banks and permit only one per lane. See Full Duplex 1GbE and SGMII IOCDR.

8.4. I/O Clock Networks (Ask a Question)

Each I/O contains a fabric clock connection, a high-speed I/O clock resource, and a lane controller clock resource for efficient clock distribution. All of these four clock networks can be used to interface with the IOD block.

There are some specific PolarFire FPGA IO clock differences along the North and South edges of the devices. The I/O span width of HS_IO_CLK trees on the North Edge is different across Banks 0, 1, and 7 in the MPF300/MPF500. There is no Bank 7 in MPF100/MPF200.

Although, Bank 2 is available in all PolarFire FPGA devices, the I/O span width of the HS_IO_CLK trees varies between device sizes. MPF100/200 device sub-divides Bank 2 into two sub-banks. This means the HS_IO_CLK tree is split between the available I/O within Bank 2. For MPF300/500, there is one continuous Bank 2 and also includes Bank 6. In these devices, the HS_IO_CLK is not split within any Bank.

For more information about global clock network, see PolarFire Family Clocking Resources User Guide.

8.4.1. Global Clock Resource (Ask a Question)

Each IOD has two global clock inputs from the fabric: one for the receive block (Receive IOD) and the other for the transmit block (Transmit IOD and Enable IOD). Libero SoC automatically routes the clock signals through the global clock network and connects to the two global clock inputs of the IOD block, if they are driven from the specified resources. The global clock network can be driven by any of the following:

- Preferred clock inputs (CLKIN_z_w)
- Oscillator clocks
- CCC (PLL/DLL)
- Fabric routing
- Clock dividers
- NGMUXs
- Transceiver reference clock inputs

For more information about global clock architecture, see PolarFire Family Clocking Resources User Guide.

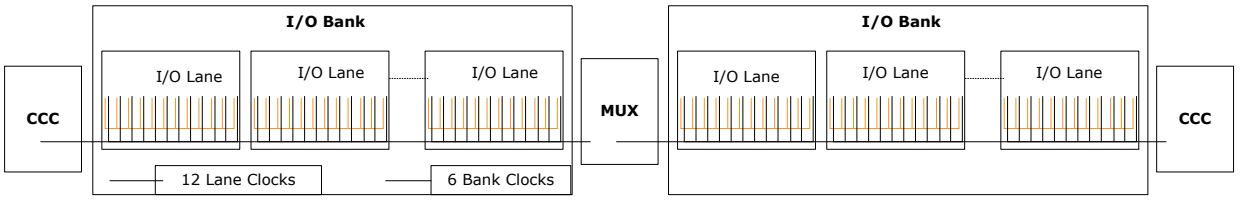
8.4.2. Regional Clock Networks (Ask a Question)

The regional clock networks are low-latency networks. They can only distribute clocks to a certain area of the device with low skew. They can be driven from the divided CDR clock and the divided high-speed IO clock. PolarFire FPGAs and PolarFire SoC FPGAs offer one regional clock buffer per I/O lane on the northern, southern, and western edges. The size of the region depends on the regional clock buffer location and does not overlap. For more information about regional lock buffer location, see PolarFire Family Clocking Resources User Guide.

8.4.3. Lane Clock Resources (Ask a Question)

Each lane has several clock networks in the I/O lane. The lane clock resources are distributed from each lane controller to each of the 12 IODs within a lane. The lane clock resource is not controllable as Libero SoC automatically uses the lane clock resource based on the I/O configuration.

Figure 8-10. Distribution of the Lane Clock

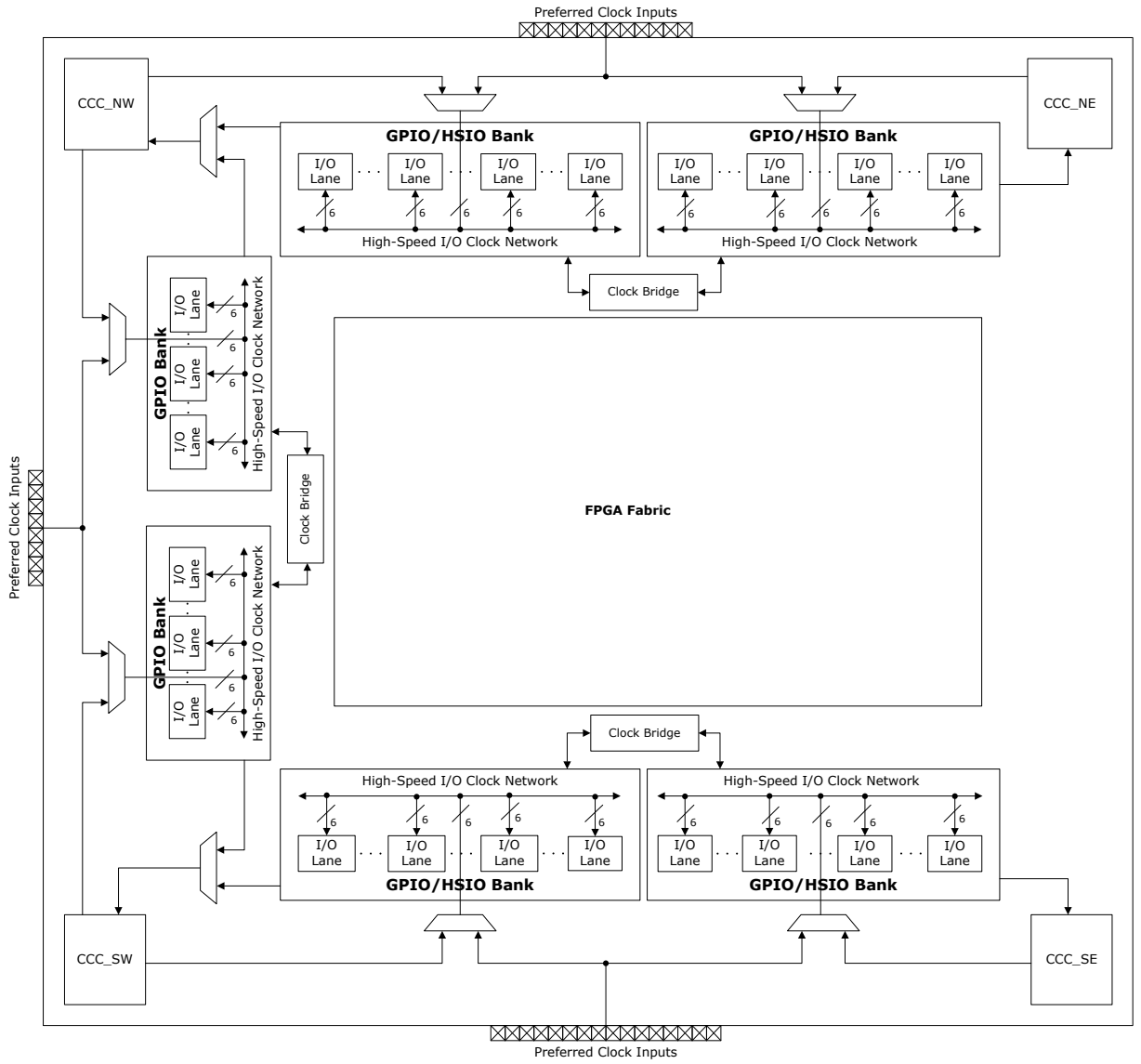


8.4.4. High-Speed I/O Bank Clock Resource (HS_IO_CLK) (Ask a Question)

High-speed I/O bank clock networks are integrated into I/O banks and distribute clocks along the entire I/O bank with low-skew. They are used to clock data in and out of the I/O logic when implementing the high-speed interfaces. The high-speed I/O clock networks are located on the east corner of the FPGA fabric. Each I/O bank can have six high-speed I/O clocks. High-speed I/O clocks from adjacent banks on the same edge can be bridged to build large I/O interfaces. HS_IO_CLK bridging is allowed only for fractional IOD Rx interfaces (See RX_DDR Fractional Aligned/Fractional Dynamic Interfaces).

High-speed I/O clock networks are driven either from I/Os or CCCs. The high-speed clocks can be configured to feed reference clock inputs of adjacent CCCs. HS_IO_CLKs are transparent as they are setup by Libero SoC based on configuration.

Figure 8-11. Distribution of the HS_IO_CLK



8.4.5. Bit Slip (Ask a Question)

BITSLIP is used to align the de-serialized input data burst into the fabric (called word or bit alignment). Serial input data streams require a matching high-frequency clock (HS_IO_CLK), which is derived from the serial input signals to the FPGA inputs. Using BITSLIP allows for word framing by providing a control signal generated in the FPGA fabric and by parallel word logic running at parallel word clock rates. The Lx_BIT_SLIP input control is synchronized to the HS_IO_CLK clock allowing word framing by suppressing one HS_IO_CLK pulse. Assertion of the Lx_BITSLIP control signal allows the word framing to change by only one bit position. Slipping the received data by one bit effectively shifts the word boundary by one bit and only occurs once per word. This operation happens once at initial data startup. Slip is initiated by a rising edge of the Lx_BIT_SLIP signal from the core fabric. It generates a single high-speed clock pulse in the bank clock (HS_IO_CLK) domain. This pulse is used for glitch less and synchronous stopping of the clock. Enabling the BITSLIP exposes the Lx_BIT_SLIP port that can be used to rotate the 8-bit word from the IOD to match the proper alignment of the data per lane. A typical bit slip sequence is as follows:

- Allows bit and word alignment of data.

- Try a slip, evaluate, and iterate until alignment is achieved.

The **Libero SoC Generic I/O Interface** configurators allow optional use of the BITSLIP function.

The bit slip function is used in one of the four ways. Following are the examples:

DDRX2 Modes: (Incrementing round robin)

It slips the word 1-bit at a time. By activating the slip function four times using rising edge of Lx_BIT_SLIP, every word combination can be analyzed during input training to find the required word alignment.

The first bit in the word changes with each activation of the slip pulse. (0,1,2,3,0,1,2,....)

(Example:1000, 0100, 0010, 0001)

DDRX4 Modes: (scrambled, round robin)

It slips the word to different starting positions, one at a time. By activating the slip function eight times using rising edge of Lx_BIT_SLIP, every word combination can be analyzed during input training to find the required word alignment.

The first bit in the word changes with each activation of the slip pulse.

As an example, starting from a word with a first bit such as 0, the pattern of slips is 0, 5, 6, 3, 4, 1, 2, 7, 0, 5,... (-3,+1,-3,+1...). The next slip (-3 or +1) is always a function of the last slip.

After an ARST_N, the first slip always starts with a -3 slip.

(Example of words after each slip starting after a reset: 01101000, 01000011, 10100001, 00001101, 10000110, 00110100, 00011010, 11010000)

DDRX5 Modes: (scrambled, round robin)

It slips the word to different starting positions, one at a time. By activating the slip function 10 times using rising edge of Lx_BIT_SLIP, every word combination can be analyzed during input training to find the required word alignment. The first bit in the word changes with each activation of the slip pulse.

As an example, starting from a word with a first bit such as 0, the pattern of slips is 0, 7, 8, 5, 6, 3, 4, 1, 2, 9, 0, 7,... (-3,+1,-3,+1...) The next slip (-3 or +1) is always a function of the last slip.

After an IOD reset, the first slip always starts with a -3 slip.

(example of words after each slip starting after a reset: 0111110000, 1110000011, 1111000001, 100001111, 1100000111, 0000111110, 0000011111, 0011111000, 0001111100, 1111100000)

DDRX3.5 does not include the Lx_BIT_SLIP capability. Word alignment must be accomplished using FPGA hosted IP.

8.5. Generic I/O Interfaces (Ask a Question)

Many pre-defined interfaces are available from the Libero SoC I/O configurator. You can select an interface from the list that closest matches their needs. See Generic IOD Interface Implementation for more information about software supported configurations. Based on targeted data rate, configurations use static settings that determine the clock or data relationships fixed by Libero SoC programming of delay elements within the IOD. Dynamic configuration uses dedicated logic controlled by fabric-based training IP that samples and adjusts internal timing elements to optimize the clock to data relationships. See Dynamic IOD Interface Training.

When building generic high-speed DDR interfaces, it is required to follow the Interface Rules described for each type of interface. The I/O supports a number of interface modes that can be selected to build the required data interface. The Package Pin Assignment Tables (PPAT) for device and package combination is available. The PPAT is used as a reference to select the proper pins with connectivity for the required resources needed for the interface. You must also be aware of performance specifications for IOA types when building their particular I/O interface. Select an I/O

type that matches the desired maximum performance rate by referencing the respective PolarFire FPGA Datasheet or PolarFire SoC FPGA Datasheet.

I/O blocks are used to construct dedicated memory interfaces. These interfaces are generated by Libero SoC using dedicated memory interface configurators for LPDDR3, DDR3, and DDR4 interfaces.

8.5.1. RX DDR Interfaces (Ask a Question)

The IOD block are assembled using a combination of modules—Delay, IREG or IGEAR, FIFO, Gearing, Lane Controller, and Soft Training IP (TIP, not built automatically by Libero SoC). The I/O makes a direct use of clock topologies around the perimeter of the I/O ring to build synchronous I/O interfaces including lane clocks and bank (HS_IO_CLKs) clocks, local, and global clocks.

Purpose built input capture circuitry uses DDR registers, which captures incoming data on both the rising and falling edges of the clock incoming clock. The RX DDR interfaces are constructed in several input widths and clocking variations using the Libero SoC I/O interface configurators.

In FPGA generic I/O interfaces, there are three types of external interface definitions—centered, aligned, and fractional-aligned. In a centered I/O interface—at the device input pins—the clock is centered in the data opening. In an aligned external interface—at the device pins—the clock and data transition are aligned or edge-on-edge. Fractional aligned IOD mode is used when the receive clock is a fraction of the data rate. Interfaces use either a static or dynamic optimization methods to achieve specific data rate targets. Static uses Libero SoC generated data and clock delay tunings. Using constraints, you can adjust the data delay of a static interface. Dynamic uses IOD capabilities to adapt the interface for optimal performance. Static interfaces are turn-key using Libero SoC whereas dynamic requires user integration to optimize the interface for maximum performance.

The clock source in both types of interfaces can be sourced for a global, regional, or lane clock. For more information about clocking topologies, see PolarFire Family Clocking Resources User Guide.

The Generic I/O Interfaces use a naming convention as follows:

- Direction_Gearing_Capture clock_Fabric clock_Clock to data relationship
- TX (direction), DDR (gearing), R (regional), C (Centered) ==> TX_DDR_R_C
- DDRX (direction), B (HS_IO_CLK), FA (Fractional Clock Aligned), DYN (Dynamic alignment), FDYN (Fractional Clock with dynamic alignment)

Both the device families include the following generic RX DDR interface types.

Table 8-6. Generic RX DDR Interfaces[1,2]

Name	Clock Type	Gearing Ratio	Description
RX_DDR_G_A	Continuous	1	Rx DDR w/Global aligned
RX_DDR_R_A	Continuous	1	Rx DDR w/Regional clock aligned
RX_DDR_G_C	Continuous	1	Rx DDR w/Global centered
RX_DDR_R_C	Continuous	1	Rx DDR w/Regional clock centered
RX_DDRX_B_G_A	Continuous	2, 3.5, 4, 5	Rx DDR geared w/Bank aligned using Global clock
RX_DDRX_B_G_C	Continuous	2, 3.5, 4, 5	Rx DDR geared w/Bank centered using Global clock
RX_DDRX_B_R_A	Continuous	2, 3.5, 4, 5	Rx DDR geared w/Bank aligned using Regional clock
RX_DDRX_B_R_C	Continuous	2, 3.5, 4, 5	Rx DDR geared w/Bank centered using Regional clock

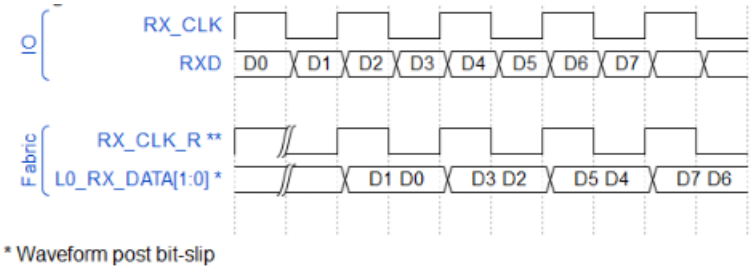
Notes:

1. For more information about maximum operating frequency, see respective PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, RT PolarFire FPGA Datasheet, or RT PolarFire SoC Datasheet.
2. Regional clock interfaces use the generated HS_IO_CLK to capture the data and then transfer to the regional clock within the FPGA fabric.

8.5.2. RX_DDR_G_A/ RX_DDR_R_A—Aligned Interfaces with Static Delays (Ask a Question)

The RX_DDR_G_A and RX_DDR_R_A interfaces are used when the DDR data and clock signals are aligned at the external input pins as shown in the following figure. This interface uses a continuous clock. Internally, the aligned interface is required to adjust the clock to satisfy the capture flip-flop setup and hold times. The adjustments are done by input delay settings, which are automatically applied from the Libero SoC software. The interfaces shown in the following figure use a gearing ratio of 1 and the maximum X1 data rate. For more information about data rate, see respective PolarFire FPGA Datasheet or PolarFire SoC FPGA Datasheet. There are two interface configurations based on clock source topology being either global or lane-based.

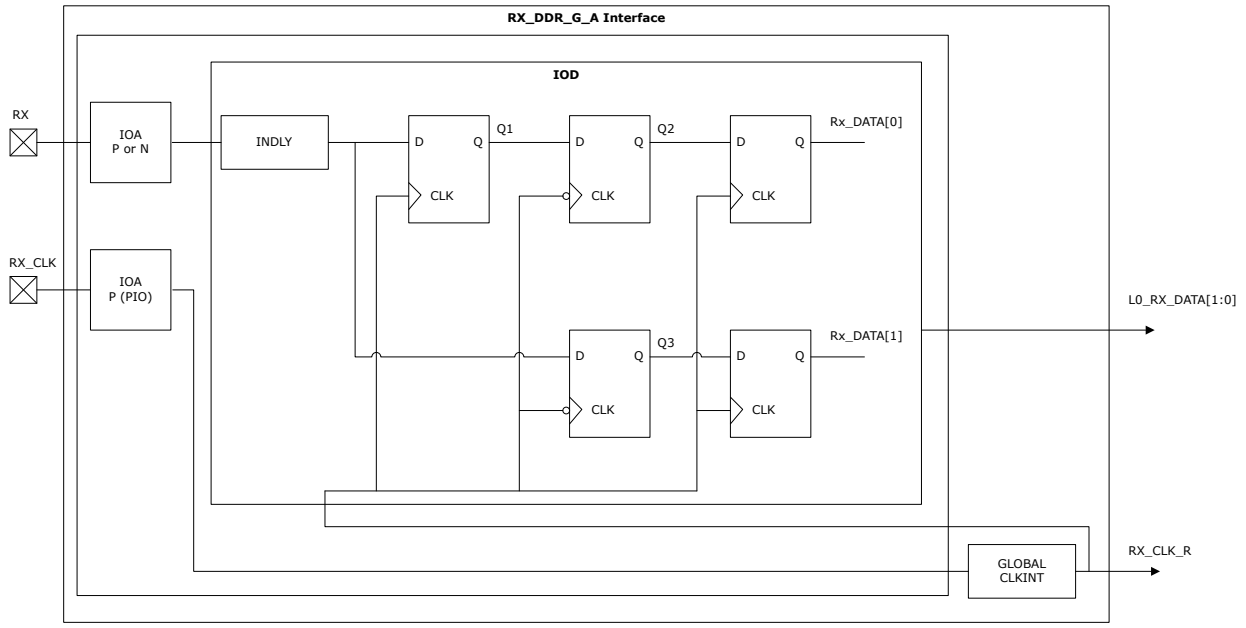
Figure 8-12. Aligned Data and Clock Waveform



In the RX_DDR aligned interface using a global clock assignment, it receives RX data and RX_CLK clock through I/Os and passes RX_DATA and RX_CLK_R to the fabric. The input clock is passed directly to the GLOBAL CLKINT that is sourced to the IOD logic. Libero SoC statically sets the input delay cells within the IOD to cancel RX vs RX_CLK injection time to flip-flop, plus an additional offset to internally center the data/clock relationship.

Global CLKINT resource drives the receive clock for fabric interface RX_CLK_R into the fabric.

Figure 8-13. RX_DDRX1 Aligned Interface Using Global Clock

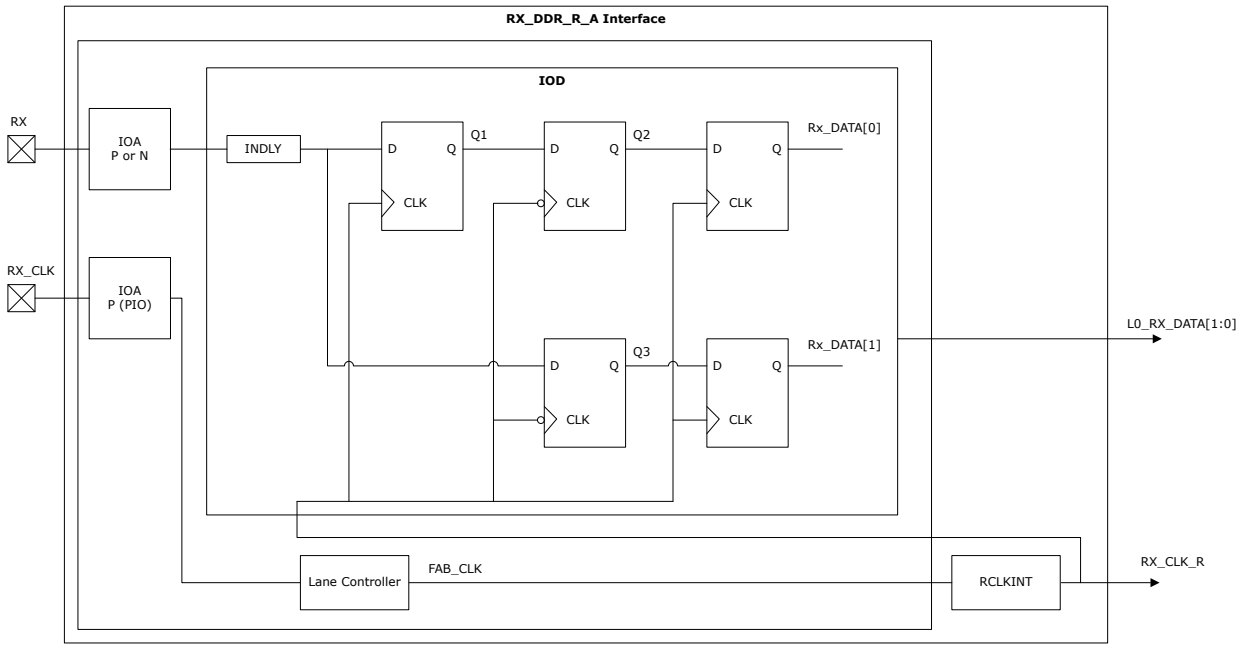


The RX_DDR aligned interface using a lane clock assignment receives RX data and RX_CLK clock through I/Os and passes RX_DATA to the IOD. This is an aligned interface using a regional system clock distribution. This uses a continuous clock. RX_CLK is sent to the lane controller. The lane

controller manages the skew and passes the FAB_CLK to a RCLKINT. The clock is sent to both the IOD and to the fabric from RCLKINT. Libero SoC statically sets the input delay cells within the IOD to cancel RX vs RX_CLK injection time to flip-flop, plus an offset to internally center the data/clock relationship.

The receive clock for fabric interface RX_CLK_R, is driven by RCLKINT resource into the fabric.

Figure 8-14. RX_DDRX1 Aligned Interface Using Regional Clock



8.5.2.1. Interface Ports (Ask a Question)

The following table lists the RX_DDR_[G:R]_A interface mode ports.

Table 8-7. RX_DDR Aligned Interface Mode Ports

Port	I/O	Description
RX	Input	Input DDR data. Supports up to 128-bits for _G interfaces and 11-bit for _R interfaces.
RX_CLK	Input	Input DDR clock.
L#_RX_DATA[n:0]	Output	DDR output to FPGA fabric. 'n' equals the output pins from the geared DDR component to the fabric where the even numbered pin is the rising edge data and the odd numbered pin is the falling edge data of the DDR signal. The number of fabric pins are based on the number of I/Os and the gearing ratio. L# is associated with the # of external input pins up to 128 maximum.
RX_CLK_R/ RX_CLK_G	Output	Receive clock to FPGA fabric using a global (G), regional (R) clock.

(1) Other pins are visible when advanced options are used. See Generic IOD Interface Implementation.

8.5.2.2. Interface Selection Rules (Ask a Question)

The following rules apply when assigning a pin to the RX_DDR_G_A aligned interface:

- Up to 12 single-ended data I/O and six differential data I/O.
- RX_CLK input must be placed in an I/O with the CLKIN_z_w function in the same bank as other I/Os.
- One IOD per data I/Os.
- One IOA per data and clock I/Os.

- RX IOA can be freely placed.

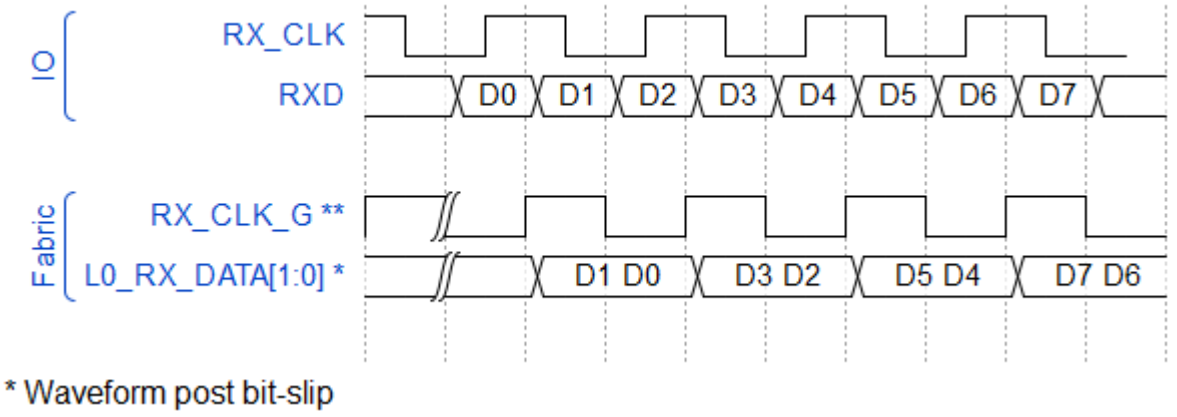
The following rules apply when assigning a pin to the RX_DDR_R_A aligned interface:

- Up to 11 single-ended data I/O and five differential data I/O.
- Uses one LANECTRL to connect to regional clock.
- Uses one regional clock.
- RX and RX_CLK I/Os must be placed in the same I/O lane.
- RX_CLK input must be placed in the P side I/O with the DQS function in the lane.
- RX and RX_CLK I/Os must be placed in the same bank (RX and RX_CLK I/O pins can be shared across banks 0 and 7).
- One IOD per data I/Os.
- One IOA per data and clock I/Os.

8.5.3. RX_DDR_G_C and RX_DDR_R_C—Centered Interfaces with Static Delays (Ask a Question)

The RX_DDR_G_C and RX_DDR_R_C interfaces have clock and data signals at the external input pins with the clock centered along the incoming data and uses a continuous clock as shown in the following figure. This interface strategy is similar to the aligned. The Libero SoC controlled input delay is set to cancel RX vs RX_CLK injection time to flip-flop. This is used to balance the clock and data delay—to the first flip-flop—to maintain the setup and hold requirements by compensating for the internal delays.

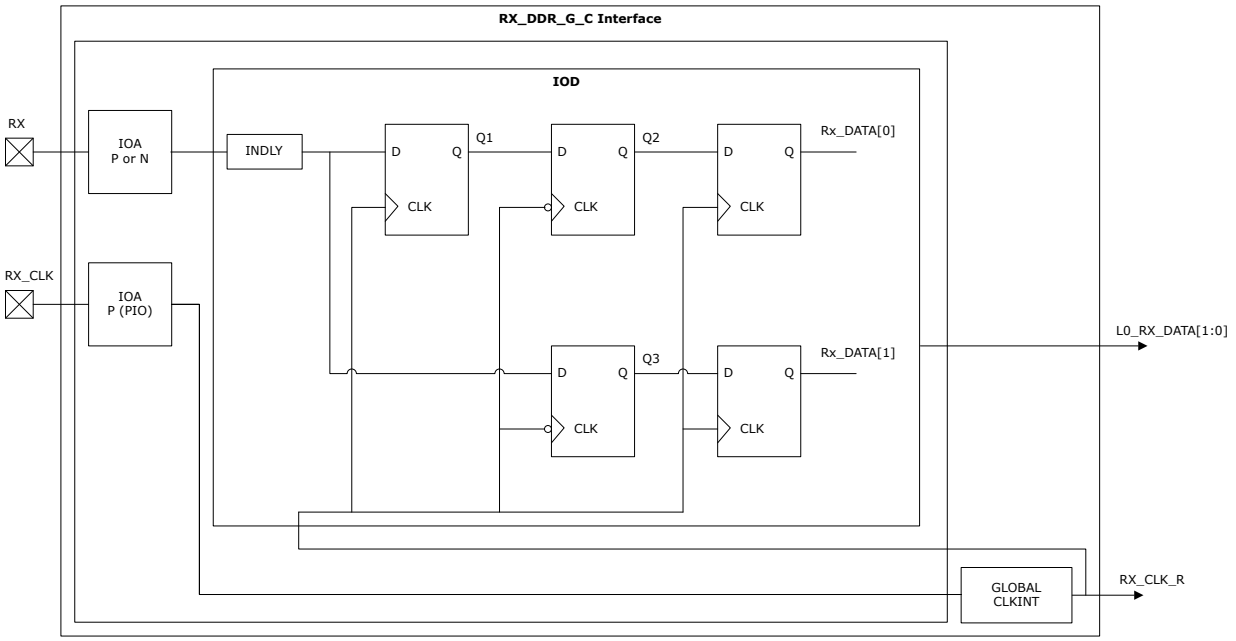
Figure 8-15. Centered Data and Clock Waveform



Using a global clock assignment receives RX data and RX_CLK clock through I/Os and passes RX_DATA and RX_CLK_R to the fabric. The input clock is passed directly to the GLOBAL CLKINT, sourced to the IOD logic, and forwarded to the fabric.

Global CLKINT resource drives the receive clock for fabric interface RX_CLK_R into the fabric.

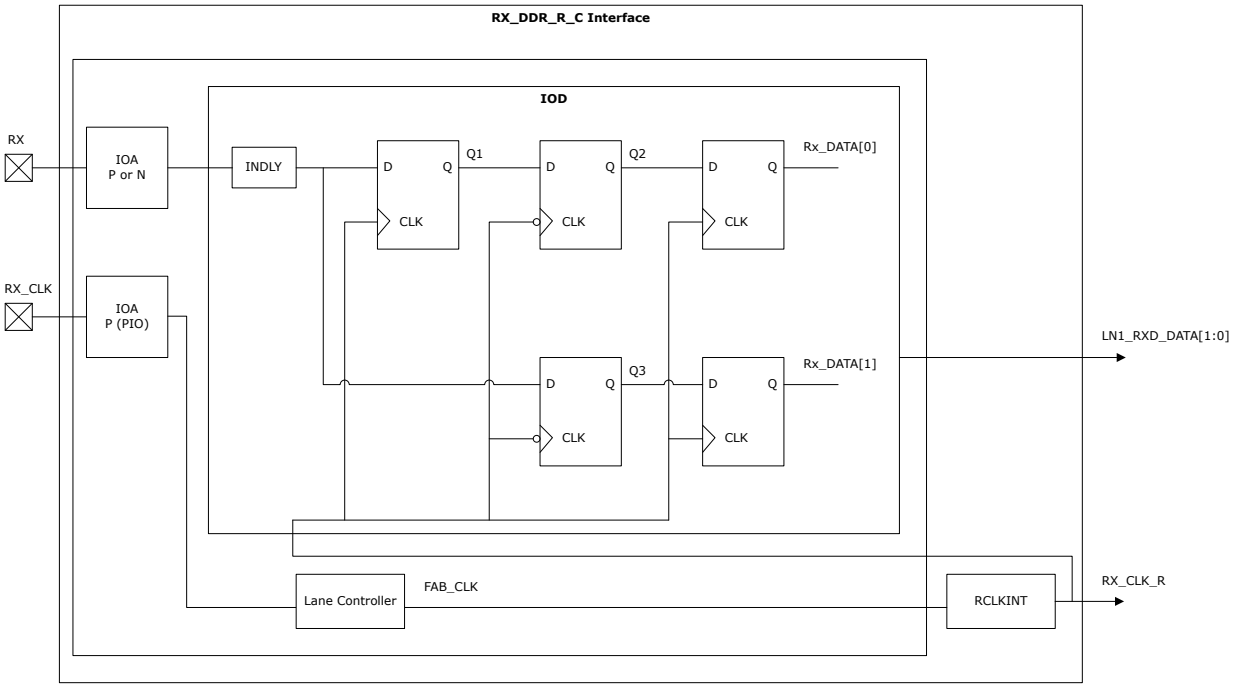
Figure 8-16. RX_DDRX1 Centered Interface Using Global Clock



The RX_DDR centered interface using a lane clock assignment receives RX data and RX_CLK clock through I/Os, passes RX_DATA to the IOD, and RX_CLK_R to the lane controller. This uses a continuous clock. The lane controller manages the skew and passes the FAB_CLK to RCLKINT. The input clock is sent to both the IOD and to the fabric from RCLKINT.

RCLKINT resource drives the receive clock for fabric interface RX_CLK_R into the fabric.

Figure 8-17. RX_DDRX1 Centered Interface Using Regional Clock



8.5.3.1. Interface Ports (Ask a Question)

The following table lists the RX_DDR_[G:L:B]_C interface mode ports.

Table 8-8. RX_DDR Centered Interface Mode Ports

Port	I/O	Description
RX	Input	Input DDR data. Supports up to 128-bits for _G interfaces and 11-bit for _R interfaces.
RX_CLK	Input	Input DDR clock.
L#_RX_DATA[m:0]	Output	DDR output to FPGA fabric. 'm' equals the output pins from the geared DDR component to the fabric where the even numbered pin is the rising edge data and the odd numbered pin is the falling edge data of the DDR signal. The number of fabric pins are based on the number of I/Os and the gearing ratio. L# is associated with the # of external input pins up to 128 maximum.
RX_CLK_R/ RX_CLK_G	Output	Receive clock to FPGA fabric using a global (G) or regional (R) clock.

(1) Other pins are visible when advanced options are used. See Generic IOD Interface Implementation.

8.5.3.2. Interface Selection Rules (Ask a Question)

The following rules apply when assigning a pin to the RX_DDR_G_C centered interface:

- Up to 12 single-ended data I/O and six differential data I/O.
- RX_CLK input must be placed in an I/O with the CLKIN_z_w function in the same bank as other I/Os.
- One IOD per data I/Os.
- One IOA per data and clock I/Os.
- RX IOA can be freely placed.

The following rules apply when assigning a pin to the RX_DDR_R_C centered interface:

- Up to 11 single-ended data I/O and five differential data I/O.
- Uses one LANECTRL to connect to regional clock.
- Uses one regional clock.
- RX and RX_CLK I/Os must be placed in the same I/O lane.
- RX_CLK input must be placed in the P side I/O with the DQS function in the lane.
- RX and RX_CLK I/Os must be placed in the same bank (exception on device with bank7, I/Os can be either in both bank0 and bank7).
- One IOD per data I/Os.
- One IOA per data and clock I/Os.

8.5.4. RX_DDRX_B_G_C and RX_DDRX_B_G_A/RX_DDRX_B_R_A Interfaces with Static Delays (Ask a Question)

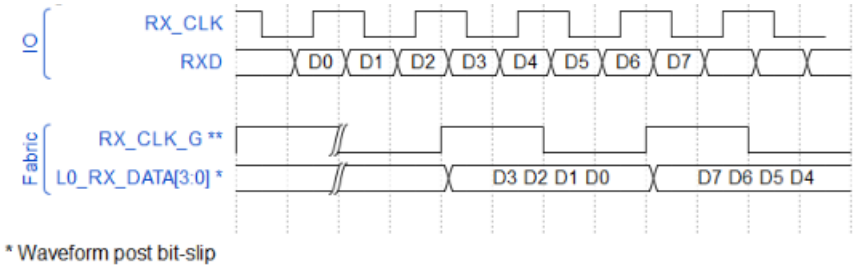
RX_DDR interfaces can use x2, x3.5, x4, and x5 gearing using bank and lane oriented, high-speed I/O clock networks that provide low-skew, clocks distributed along the edge of the device to service the I/Os. Used to clock data into the I/O logic when implementing the I/O interfaces, the clocks are tightly managed to support wide source synchronous interfaces. The clock domain transfer for the data from the high-speed IO clock to the low-speed system clock is guaranteed by design. These modes permit wider data transfers to the fabric hence achieving more data throughput with lower fabric clock transfers.

The RX_DDRX[2,3.5,4,5]B_G_A/_B_R_A and RX_DDRX[2,3.5,4,5]B_G_C interfaces are also supported by static settings when the user is aware of the input and clock relationship at the boundary of the device. Similar to the RX_DDRX1 interfaces, the Libero SoC IOD configurator creates a component that meets the gearing criteria and is correct by construction from the pads to the fabric interface

making use of the correct input pins required for clock and data. For each high-speed input receiver, the component is generated with the appropriate fabric pins based on the gearing ratio. For example, if a single high-speed input is intended to be geared by 4, then the component has 8 pins. The 8-pins has a relative pin name LN0_RXD_DATA[7:0] where as [0:1], [2:3], [4:5], [6:7] are the DDR equivalent for x4 geared data to the fabric.

The precise granularity of the embedded IOD block architecture limits the controllable impact that the software place and route process can have on timing results. The timing through the elements of the IOD blocks are guaranteed by construction across the process, voltage, and temperature range that is specified in the data sheet. Users must not perform any additional worse case timing analysis on the IOD paths associated with the embedded high-speed clocks and related elements of the IOD blocks. If attempted, the timing analysis algorithms computes overly pessimistic IOD expectations that does not accurately portray the actual device performance.

Figure 8-18. Rx_geared Waveform



8.5.4.1. Interface Ports (Ask a Question)

The following table lists the RX_DDR_B_C and RX_DDR_B_A interface mode ports.

Table 8-9. RX_DDR_B_C and RX_DDR_B_A Interface Mode Ports[1]

Port	I/O	Description
RX	Input	Input DDR data. Supports up to 128-bits for _G interfaces and 11-bit for _R interfaces.
RX_CLK	Input	Input DDR clock.
ARST_N	Input	Asynchronous reset to IOD and lane controller. ARST_N inputs are independent asynchronous resets to both the Rx and Tx IOD blocks.
HS_IO_CLK_PAUSE	Input	Toggling the HS_IO_PAUSE: – Resets the IOD RX state machines. This reset re-synchronizes pattern to HS_IO_CLK (bank clock) and RXCLK. – Resets any adjustment done through SLIP operation. – Resets the IOD TX state machines. This reset synchronizes HS_IO_CLK and TXCLK. – HS_IO_CLK_PAUSE must be pulsed after PLL lock is asserted in fractional aligned mode allowing the I/O Gearing state machine to detect the phase difference between fabric clock and clock coming out of PLL. – HS_IO_PAUSE does not disrupt delay line value settings.
L#_RX_DATA[m:0]	Output	DDR output to FPGA fabric. 'm' equals the output pins from the geared DDR component to the fabric where the even numbered pin is the rising edge data and the odd numbered pin is the falling edge data of the DDR signal. The number of fabric pins are based on the number of I/Os and the gearing ratio. L# is associated with the # of external input pins up to 128 maximum.
RX_CLK_R/ RX_CLK_G	Output	Receive clock to FPGA fabric using a global (G) or regional (R) clock. Global and regional can be used for aligned interfaces. Center-aligned interfaces can only use global clock.

(1) Other pins are visible when advanced options are used. See Generic IOD Interface Implementation.

8.5.4.2. Interface Selection Rules (Ask a Question)

The following rules apply when assigning a pin to the RX_DDRX_B_G_A, RX_DDRX_B_R_A, and RX_DDRX_B_G_C interfaces:

- Interface uses two ICB_CLKDIVDELAY and three HS_IO_CLK.
- RX_CLK input must be placed in an I/O with the CLKIN_z_w function in the same bank as other I/Os.
- RX and RX_CLK I/Os must be placed in the same bank (exception on device with bank7, I/Os can be either in both bank0 and bank7).
- One IOD per data I/Os.
- One IOA per data and clock I/Os.
- IOA from two different interfaces (TX/RX/DDR/QDR/OCTAL/CDR) cannot be placed in the same I/O lane.

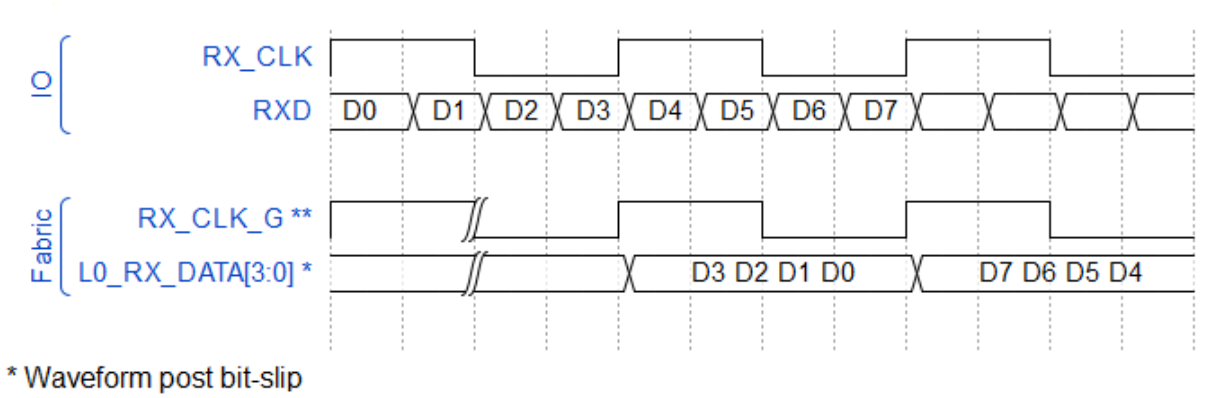
8.5.5. RX_DDR Fractional Aligned/Fractional Dynamic Interfaces (Ask a Question)

The DDR fractional aligned IOD mode is used when the receive clock is a fraction of the data rate. A CCC PLL is inserted by Libero SoC into the clock path with a multiplier of 1, 2, 4, 8, or 10 to match data bit rate. For example, source synchronous clock input RX_CLK (which is data-rate / 4) is provided as a reference clock to a fabric PLL, and generates the HS_IO_CLK which is 2X the RX_CLK. With statically trained interface, the static delays to ensure the HS_IO_CLK clock edge alignment within the RXD data bit window. This pre-instantiated PLL also generates the fabric clock (equal to the RX_CLK or data-rate / 4), which is used by the user logic in the fabric to clock the RX_DATA bits coming out of the IOD macro into the fabric.

Both RX_DDRX_FA and RX_DDRX_FDYN include PLLs as required to receive a clock that is slower than the data rate. RX_DDRX_FA IOD macro use Libero predefined static settings to adjust the clock and data alignment. RX_DDRX_FDYN IOD macro is similar but it also includes the control and status ports required for the alignment which is controlled by the training IP. Both operate similarly except for the training aspect of the RX_DDRX_FDYN. The following figure shows functional waveforms for RX_DDRX_FA and RX_DDRX_FDYN.

The following figures shows the waveform diagram of fractional aligned data and clock.

Figure 8-19. Fractional Aligned Data and Clock Waveform



8.5.5.1. Interface Ports (Ask a Question)

The following table lists the port names and description of fractional aligned interface mode.

Table 8-10. Fractional Aligned Interface Mode Ports

Port	I/O	Description
RX	Input	Input DDR data. Supports up to 11 bits wide and all bits must fit within a lane.
RX_CLK	Input	Input DDR clock.

Table 8-10. Fractional Aligned Interface Mode Ports (continued)

Port	I/O	Description
ARST_N	Input	Asynchronous reset to IOD and lane controller. ARST_N inputs are independent asynchronous resets to both the Rx and Tx IOD blocks. It holds associated interface PLLs in power-down. Asserting ARST_N to the TX/RX IODs also resets any updated delay values set by dynamic training and reverts to the initial static delays set by the Libero SoC. ARST_N does not reset any static clock training values.
HS_IO_CLK_PAUSE	Input	Toggling the HS_IO_PAUSE: – Resets the IOD RX state machines. This reset re-synchronizes pattern to HS_IO_CLK (bank clock) and RXCLK. – Resets any adjustment done through SLIP operation. – Resets the IOD Tx state machines. This reset synchronizes HS_IO_CLK and TXCLK. – HS_IO_PAUSE does not disrupt delay line value settings.
L#_RX_DATA[m:0]	Output	DDR output to FPGA fabric. 'm' equals the output pins from the geared DDR component to the fabric where the even numbered pin is the rising edge data and the odd numbered pin is the falling edge data of the DDR signal. The number of fabric pins are based on the number of I/Os and the gearing ratio. L# is associated with the # of external input pins up to 128 maximum.
RX_CLK_R/RX_CLK_G	Output	Receive clock to FPGA fabric using a global (G) or regional (R) clock.
PLL_LOCK	Output	Lock status of the included PLL used in clock path.
CLK_TRAIN_DONE	Output	Indicates HS_IO_CLK and system clock training is complete. See HS_IO_CLK and System Clock Training.

8.5.5.2. Interface Selection Rules (Ask a Question)

The following rules apply when assigning a pin to the RX_DDRX_B_G_FA interfaces:

- RX_CLK input must be placed in an I/O with the CCC_CLKIN_z_w function in the same bank as other I/Os. This pin allows connection to the PLL reference clock. For more information about the available pins for each device, see PolarFire Family Clocking Resources User Guide. Other preferred clock pins are not suited for this connection.
- RX and RX_CLK I/Os must be placed in the same bank (exception on device with bank7, I/Os can be either in both bank0 and bank7).
- Interface uses a PLL to generate high-speed clock.
- One IOD per data I/Os.
- One IOA per data and clock I/Os.
- IOA from two different interfaces (TX/RX/DDR/QDR/OCTAL/IO_CDR) cannot be placed in the same I/O lane.

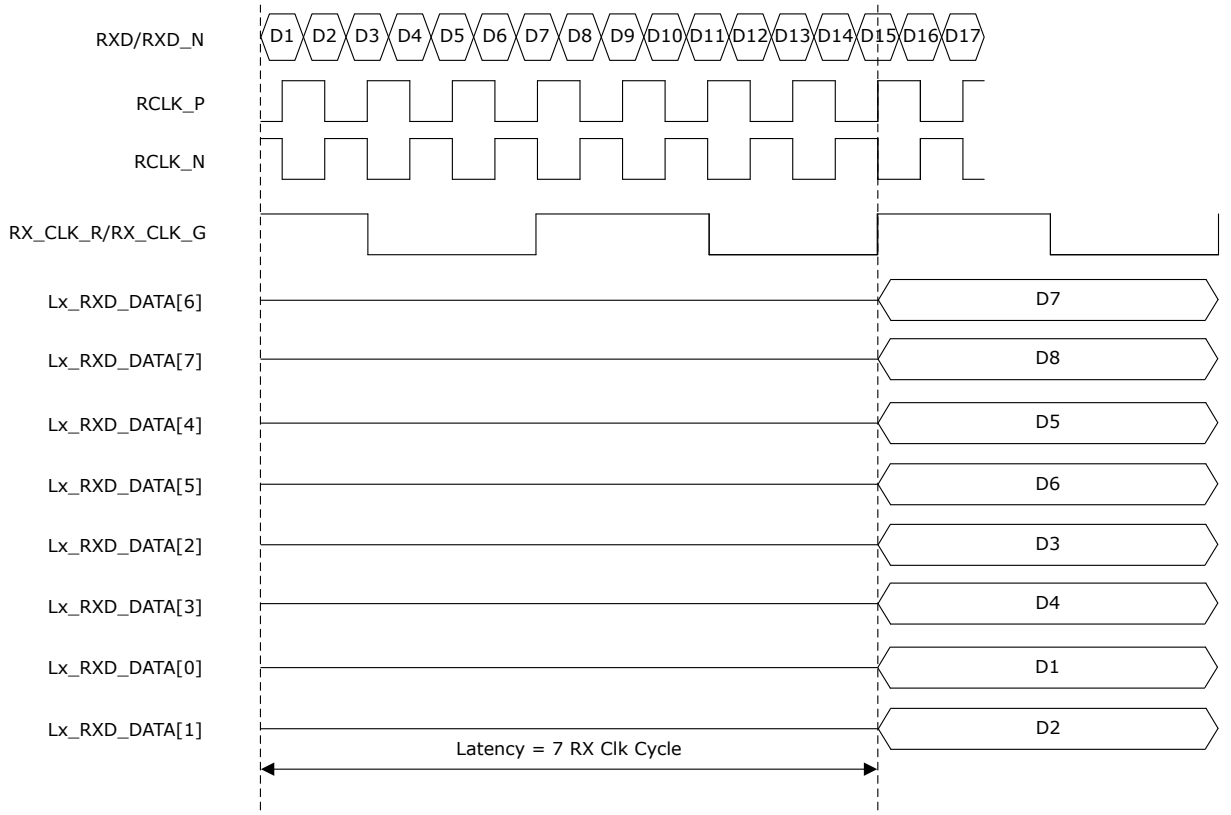
8.5.6. RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN (Ask a Question)

The RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN interface is used to capture DDR data using dynamic control. The clock and data relationship can be adjusted dynamically when the device receives the DDR data. The RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN interface uses the digital ratio of 2, 3.5, 4, and 5. These interfaces can use single-ended, voltage-referenced, or differential I/O standards.

The interface receives the data RXD and the clock RX_CLK through I/O and passed the data Lx_RXD_DATA and fabric clock (RX_CLK_FAB) to the fabric. The receive clock input (RX_CLK) is passed through the lane controller to generate RX_CLK_FAB, which is driven by RCLKINT.

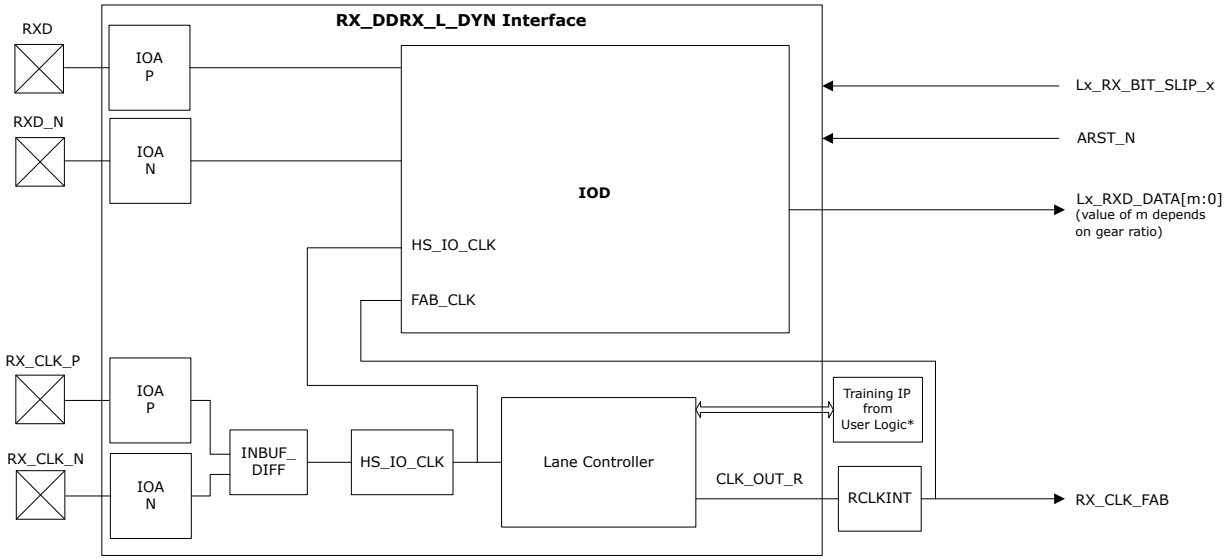
The following illustration shows the signal waveform of RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN interface when slip input is not used.

Figure 8-20. RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN Waveform



The following illustration shows the block diagram of RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN interface.

Figure 8-21. Block Diagram of the RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN Interface



Note: For information about connections between IOD block and user training IP, see Dynamic Delay Control.

8.5.6.1. Interface Ports (Ask a Question)

The following table lists the RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN interface mode ports.

Table 8-11. RX_DDRX_L_DYN Ports[1]

Port	I/O	Description
RXD/RXD_N	Input	Differential input DDR data
RX_CLK_P/RX_CLK_N	Input	Differential input clock
ARST_N	Input	Asynchronous reset to IOD and lane controller. Asserting ARST_N to the TX/RX IODs also resets any updated delay values set by dynamic training. ARST_N does not reset any static clock training values. When using dynamic IOD training, it falls back to zero delay taps after ARST_N is toggled and it is expected to be retrained. It does not revert to any initial static setting. Only static IOD configurations fall back to the initial static value found in PDC when reset.
Lx_BIT_SLIP	Input	Bit slip input (per lane) from fabric is initiated by a rising edge of the slip signal from the core fabric. Lx_BIT_SLIP is not available for DDRX3.5 gearing.
Lx_RXD_DATA[m:0]	Output	DDR output to FPGA fabric, value depends on digital ratio
RX_CLK_R/RX_CLK_G	Output	Receive clock to FPGA fabric using a global (G) or regional (R) clock.
CLK_TRAIN_DONE	Output	Indicates HS_IO_CLK and system clock training is complete. See HS_IO_CLK and System Clock Training.
CLK_TRAIN_ERROR	Output	Indicates HS_IO_CLK and system clock training did not complete. See HS_IO_CLK and System Clock Training.

(1) For more information, see Dynamic Delay Control.

The RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN interface has bit slip input from fabric, called Lx_BIT_SLIP. The slip input pin is used for word alignment. The slip function is used in 2, 4, and 5 Digital Modes—slip 1 bit at a time.

8.5.6.2. Interface Selection Rules (Ask a Question)

The following conditions are applicable when assigning pins to the RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN interface:

- Interface uses two ICB_CLKDIVDELAY and three HS_IO_CLK.
- RX_CLK input must be placed in an I/O with the CLKIN_z_w function in the same bank as other I/Os.
- RX and RX_CLK I/Os must be placed in the same bank (exception on device with bank7, I/Os can be either in both bank0 and bank7).
- One IOD per data I/Os.
- One IOA per data and clock I/Os.
- IOA from two different interfaces (TX/RX/DDR/QDR/OCTAL/CDR) cannot be placed in the same I/O lane.

8.5.7. TX DDR Interfaces (Ask a Question)

The following table lists the clock-to-data conditions of TX DDR interfaces.

Table 8-12. TX DDR Interfaces

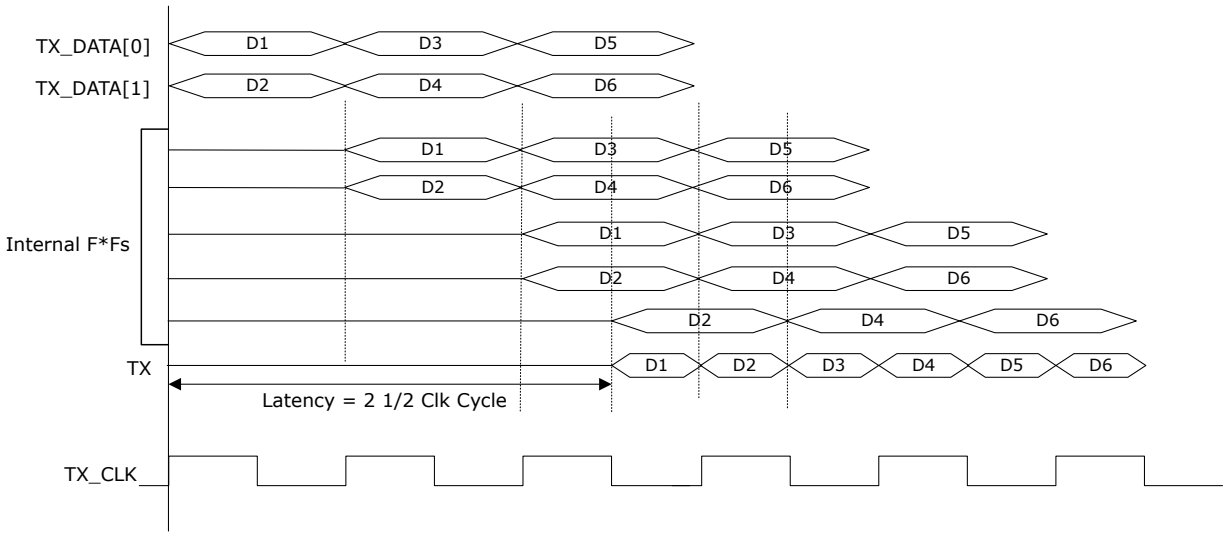
Interface Name	Topology	Gearing Ratio	Clock-to-Data Condition
TX_DDR_G/B_A	TX DDR	1	From a global clock source, aligned clock, and data

8.5.7.1. TX_DDR_G/B_A (Ask a Question)

The TX_DDR_G _A interfaces implement the DDR transmit interface where clock edges are aligned with the DDR data. The IOD block uses the fabric clock (TX_FAB_CLK) that is routed on a global clock resource to capture the transmitted data from fabric and transmit it through the TX pin.

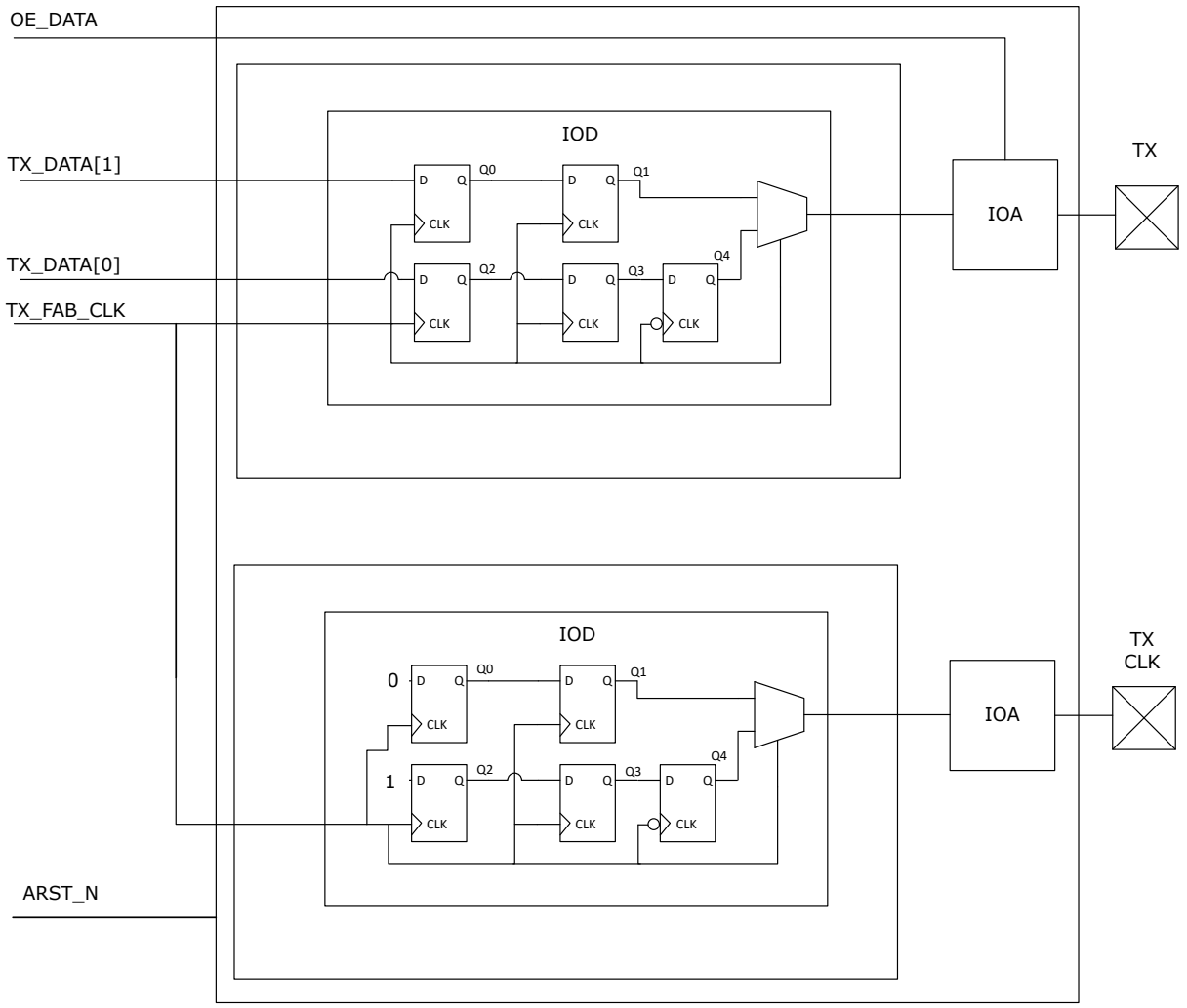
The following figure shows the TX_DDR_G_A interface signal waveform.

Figure 8-22. TX_DDR_G_A Interface Signal Waveform—TXDDRX1



The following figure shows the block diagram of the TX_DDR_G_A interface.

Figure 8-23. TX_DDR_G/B_A Interface Block Diagram—TX_DDRX1



8.5.7.2. Interface Ports (Ask a Question)

The following table lists the TX_DDR_G_A interface mode ports.

Table 8-13. TX_DDR_G/B_A Interface Mode Ports

Port	I/O	Description
TX_DATA[m:0]	Input	DDR transmit data from fabric. 'm' equals the input pins to the DDR component from the fabric where the even numbered pin is the data transmitted on the falling edge of TX_CLK and the odd numbered pin is the data transmitted on the rising of TX_CLK of the DDR signal. The number of fabric pins are based on the number of I/Os and the gearing ratio.
TX_CLK_G	Input	DDR transmit clock from fabric, and routed through global clock network.
TXD/TXD_N(m)	Output	DDR output to IOAs
TX_CLK	Input	DDR clock to IOAs
HS_IO_CLK_PAUSE	Input	Toggles the HS_IO_CLK_PAUSE Resets the IOD TX state machines This reset synchronizes HS_IO_CLK and TXCLK. It takes 5 to 10 clock cycles after deassertion of HS_IO_CLK_PAUSE until valid TX_DATA is accepted by the IOD. HS_IO_PAUSE does not disrupt delay line value settings.

Table 8-13. TX_DDR_G/B_A Interface Mode Ports (continued)

Port	I/O	Description
ARST_N	Input	Asynchronous reset to IOD and lane controller Asserting ARST_N to the TX IODs resets the IOD TX gearbox stopping the TXD and TX_CLK signals. During reset, the programmable output delay are set to 0. After the reset is released, the programmable output delay are reloaded to the programmed values.

Note: Other pins are visible when advanced options are used. See Generic IOD Interface Implementation.

8.5.7.3. Interface Selection Rules (Ask a Question)

The following conditions are applicable when assigning pins to the TX_DDR_G_A interface:

- TX and TX_CLK I/Os are freely placed. The TX data and TX_CLK skew is equal to the global clock network.
- One IOD per data and clock I/Os.
- One IOA per data and clock I/Os.

Note: At least, one CCC/PLL is required for clock phasing.

An optional feature enables the transmit parallel clock to be sent out synchronously with data. This feature is useful for applications such as CameraLink. The option is found in the Advance tab of the TX_DDR configurator GUI > Misc section > Enable user control of clock pattern. The generated TX_DDR component has a port PF_IOD_TX_CLK:TX_DATA_0 that is in the module with the TX_CLK_DATA ports.

8.6. Latency (Ask a Question)

The latency listed in the following table is an approximation and based on a specific fixed relationship for HSIO_CLKS clocks and regional clocks. Due to the flexibility and training associated with the DDR interfaces, the latency can be different than that listed by ±1 cycle.

Table 8-14. Latency for the Rx/Tx CLK Interface

IOD Mode	Direction	Latency Cycle (Rx/Tx CLK)
RX_DDRX1	Input	1
RX_DDRX2	Input	4
RX_DDRX3P5	Input	6
RX_DDRX4	Input	7
RX_DDRX5	Input	9
TX_DDRX1	Output	2.5
TX_DDRX2	Output	5.5
TX_DDRX3P5	Output	6
TX_DDRX4	Output	10.5
TX_DDRX5	Output	13.5

9. Generic IOD Interface Implementation (Ask a Question)

The I/O architecture includes many functional features to support source synchronous I/O interfaces such as DDR and QDR memory controllers, common interfaces such as RGMII, MIPI D-PHY, 7:1 LVDS, and several other non-memory user interfaces. Some interfaces such as memory interface solutions user soft-training IP to move data from the high-speed Bank I/O clock (HSIO_CLK) to the global clock of the interface.

The Libero SoC software offers many enhanced capabilities to streamline these interfaces easily into FPGA designs. The software is used to configure and generate all the high-speed interfaces such as IOD Generic RX and IOD Generic TX. The software generates a complete HDL module including clocking requirements for each of the interfaces. The Libero SoC built components are correct by construction containing the complete data path from the I/O pins to the fabric. Libero SoC software includes the following I/O interface configurators in the DirectCore catalog.

- IOD Generic RX
- IOD Generic TX

The following steps must be followed to successfully design a high-speed I/O gearing interface using Libero SoC software.

1. Determine the type of interface to implement for list of defined interfaces.
2. Use the I/O Interface configurators to build the interface.
3. Use available pre-sets within the I/O interface configurators for typical application interfaces
4. Add needed clock resources such as CCC.
5. Review package/device pin-out assignment tables for valid clock and data pins for each interface before making pin assignments. The smallest possible device/package combination that may be targeted to reduce architectural migration issues. Also see the Consolidated IOD Rules for pre-place and route guidance.
6. Confirm timing and clock constraints.
7. Define desired I/O standard for single-ended or differential I/O.
8. See the PolarFire FPGA or PolarFire SoC FPGA Device PPAT for IO planning of the design.

9.1. Software Primitives (Ask a Question)

Several software primitives are used to implement DDR interfaces. The Libero SoC I/O configurator builds and generates the component based on these primitives.

9.1.1. Input DELAY (Ask a Question)

The DELAY block is used to delay the input data from the input pin to the Input register (IREG). It adjusts among the input data bus for any skews. The data input to this block can be delayed using:

- Static—these are pre-determined delay values (for Zero Hold time, delay based on Interface Type) set by the Libero SoC software.
- DYNAMIC—uses the calibrated codes from DLL of the CCC to maintain correct timing across the system.
- Dynamic/Training IP (TIP)—the data input to this block can also be dynamically updated using Dynamic Delay controls connected to fabric IP logic. See Dynamic Delay Control.

The DELAY can be adjusted through Libero SoC. This can be done using physical design constraints (PDC) or the IOEditor GUI. The DELAY has 256 setting taps that can be adjusted to match the physical connections on the PCB. The PDC values over writes the static settings that are configured by Libero SoC defaults. For more information about Input DELAY, see Programmable I/O Delay.

9.1.2. Input Register (IREG) (Ask a Question)

The Input IREG gearing logic data path uses three sets of registers:

- Shift register
- Update register
- Transfer register

The purpose of these registers is to implement Input gearing, de-serialization of the high-speed pad signals to lower speed parallel core signals, and the clock domain transfers, as required for the specific interface standard.

9.1.3. Input FIFO (Ask a Question)

After sampling valid DDR data, the positive and negative edge data needs to cross clock domains between the external synchronizing signal (for example, DQS for DDR memory controllers) and the internal system clock. The input FIFO also provides certainty of data being received at the FPGA with slightly different arrival times.

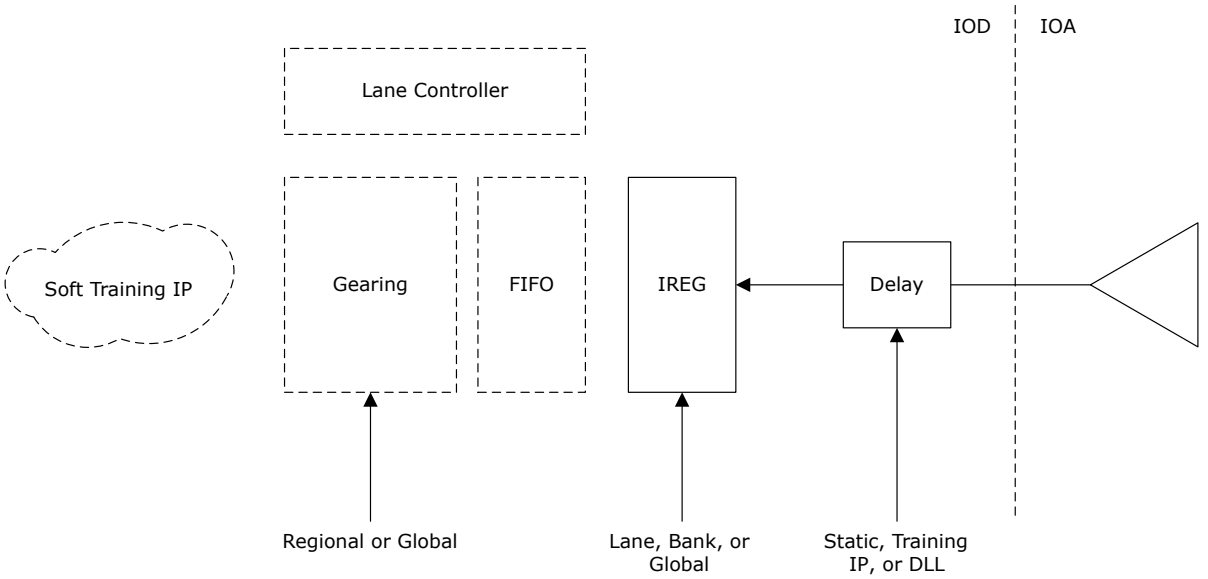
The input FIFO for each I/O is composed of two 8 flip-flop deep registers. One register is used for the input data associated with positive edge of clock and the other register is used for the input data associated with the negative edge of the clock. Both registers run on the negative clock edge, by using a previous half cycle transfer to put DDR input data all on one clock edge. There is a 3-bit write pointer and a 3-bit wide read pointer. The FIFO is used for clock domain transfers. For more information about Input FIFO, see I/O FIFO.

9.1.4. Input Gear Box (Ask a Question)

The IGEAR is composed of three sets of data registers to de-serialize the input data and transfer it to a lower core speed. It uses three sets of registers:

- Shift: running on the high-speed input clock.
- Update: running on the high-speed input clock, but controlled by an update signal from the clock controller. The update is based on the de-serialization mode required to reduce the frequency to the core speed (X2, X3.5, X4, and X5).
- Transfer: running on the core system clock. It is required to guarantee timing is met in the transition from the update register to the system clock.

Figure 9-1. IOD Modules used within a Generic DDRX I/O Interface



9.2. I/O Interface Configurators (Ask a Question)

I/O interface configurators assemble interfaces from the device input and output pins to the fabric. The configurator includes IOD blocks and the connectivity required for the interface. The configurators include tabs that show the configured component with the ports. The receiver configurator includes an interactive waveform diagram that updates the use case based on the inputs to the configurator GUI. The GUI also includes simple design rule checks to prevent from crating modules that are not allowed by the architecture.

9.2.1. IOD Templates (Ask a Question)

Many IOD templates are available for ease entry of interface settings. The interfaces (see Generic I/O Interfaces) are captured by the templates. Select a desired preset in the left pane and right-click. Click **Apply** to load the related interface configuration to the GUI. Click **View** to see the specific configuration settings. When applied, the templates auto-fills the Configuration settings within the tab for the applied template. This is a simple method of applying legal combinations for IOD configurations. Modify according to specific requirement. It also navigates the user to use the available configurations.

9.2.2. IOD Generic RX (Ask a Question)

The following table lists the receive interface software names and their related data.

Table 9-1. Receive Interface

Software Name	Ratio	Clock to Data Relationship	I/O Clock	Fabric Clock	Lane Organization	One Lane Max	Dynamic Bit Training
RX_DDR_G_A	1	Aligned	Global	Global	✗	✗	✗
RX_DDR_R_A	1	Aligned	Regional	Regional	✓	✓	✗
RX_DDR_G_C	1	Centered	Global	Global	✗	✗	✗
RX_DDR_R_C	1	Centered	Regional	Regional	✓	✓	✗
RX_DDRX_B_G_A	2, 3.5, 4, 5	Aligned	High-speed I/O Clock	Global	✓	✗	✗
RX_DDRX_B_R_A	2, 3.5, 4, 5	Aligned	High-speed I/O Clock	Regional	✓	✓	✗
RX_DDRX_B_G_C	2, 3.5, 4, 5	Centered	High-speed I/O Clock	Global	✓	✗	✗

Table 9-1. Receive Interface (continued)

Software Name	Ratio	Clock to Data Relationship	I/O Clock	Fabric Clock	Lane Organization	One Lane Max	Dynamic Bit Training
RX_DDRX_B_R_C	2, 3.5, 4, 5	Centered	High-speed I/O Clock	Regional	✓	✓	✗
RX_DDRX_B_G_FA	2, 3.5, 4, 5	Fractional Aligned	High-speed I/O Clock	Global	✓	✗	✗
RX_DDRX_B_G_DYN	2, 3.5, 4, 5	Dynamic	High-speed I/O Clock	Global	✓	✗	✓
RX_DDRX_B_R_DYN	2, 3.5, 4, 5	Dynamic	High-speed I/O Clock	Regional	✓	✓	✓

Important: For Generic RX IOD interfaces with a gearing ratio of 1, the lane controller functions are not available. Lane controller functions are only available in x2, x4, x3.5, x5 gearing ratios. As shown in preceding table, for RX_DDR interfaces with ratio equal to 1, you cannot use a high-speed I/O clock, only global or regional clocks can be used.

The following figure shows the IOD Generic Receive Interfaces.

Figure 9-2. IOD Generic Receive Interfaces—Configuration Tab

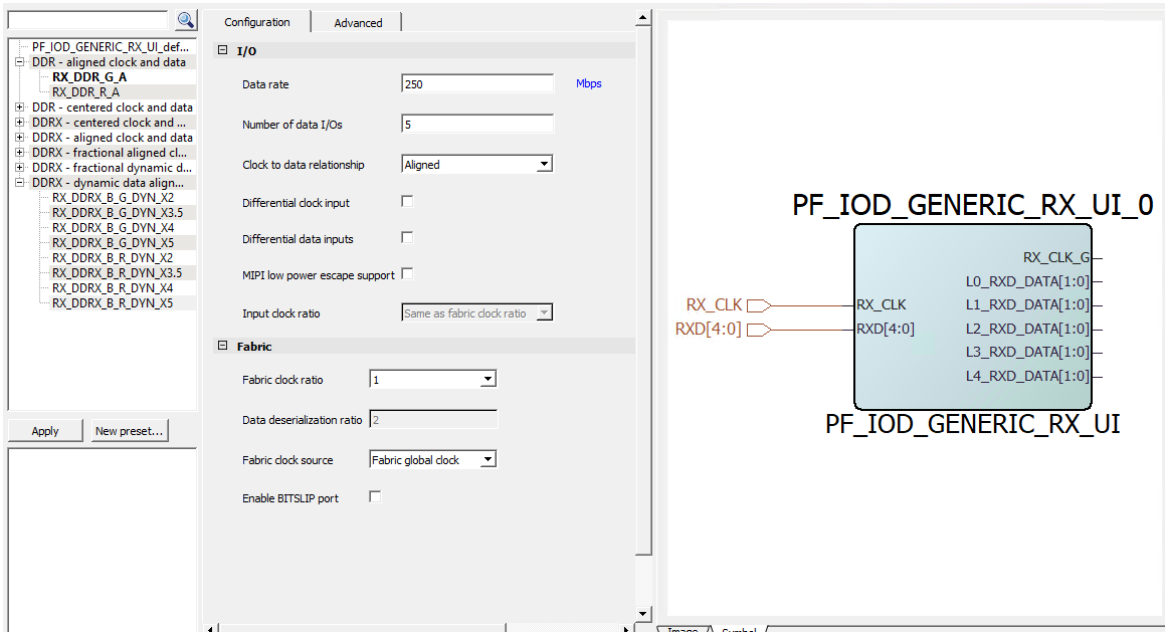


Table 9-2. IOD Generic Receive Interfaces—Configuration Tab

GUI Option	Selections
Data rate	User Input[1]
Number of data I/Os	User Input – Number of desired RX data inputs (1 to 128)
Clock to data relationship	Aligned, Centered, Dynamic, Fractional-aligned, and Fractional-dynamic[2]
Differential clock inputs	Disable (single-ended) and Enabled (differential)
Differential data inputs	Disable (single-ended) and Enabled (differential)

Table 9-2. IOD Generic Receive Interfaces—Configuration Tab (continued)

GUI Option	Selections
MIPI low power escape support	Disable and Enable
Fabric Clock Ratio	1, 2, 3.5, 4, 5
Data deserialization ratio	Predefined ports to the fabric from IOD component
Fabric clock source	Fabric regional clock Fabric global clock
Enable BITSLIP port	Disable and Enable Exposes BITSLIP pin when enabled. See Bit Slip for more information about Bit Slip. Not available for 3.5 Fabric clock ratio.

(1) See Receiver Interface (right panel) for valid data rates (Figure 9-2).

Figure 9-3. IOD Generic Receive Interfaces—Advanced Tab

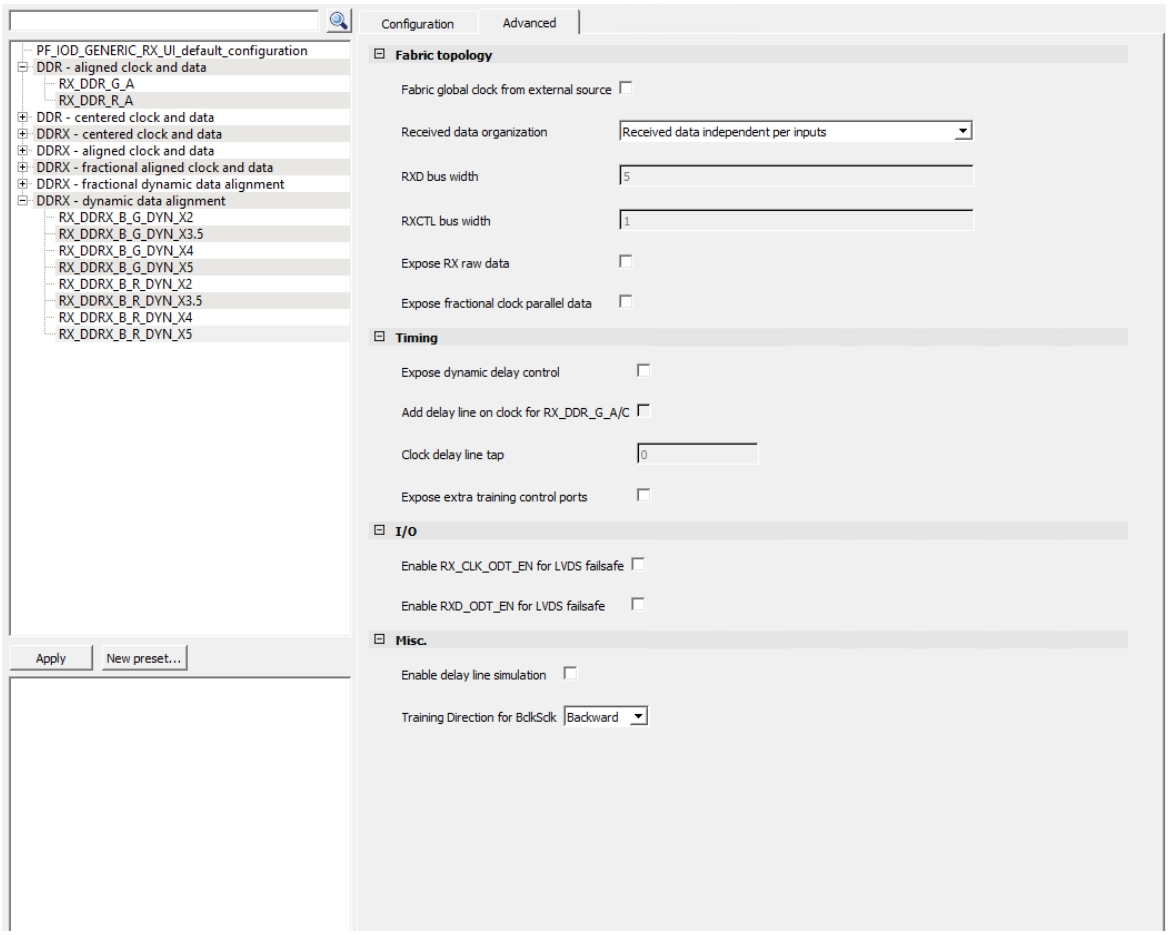


Table 9-3. IOD Generic Receive Interfaces—Advanced Tab

GUI Option	Selections
Fabric global clock for external source	Disable and Enable It is enabled when you want to use the same RX_CLK_G clock for the transmitter and the receiver.

Table 9-3. IOD Generic Receive Interfaces—Advanced Tab (continued)

GUI Option	Selections
Received data organization	Received data spread over inputs, Received data independent over inputs, Received data spread over inputs with data/Control split.
RXD bus Width	This allows organizing the splitting of the data bus.
RXCTL bus Width	This allows organizing the splitting of the data bus.
Expose Rx raw data	Disable and Enable RXD_RAW_DATA ports are exposed on the module. Expose raw data is available for all fabric clock ratios except for ratio 5.
Expose fractional clock parallel data	Disable and Enable For fractional interfaces, RXD_CLK_DATA specifies the bit-slips needed to re-frame data.
Expose dynamic delay control	Disable and Enable See Table 9-4.
Add delay line on clock for RX_DDR_G_A/C	Enables the static delay chain to be added to the clock path
Clock delay line tap	Number of delay taps to be added. For information, see Programmable I/O Delay.
Enable RX_CLK_ODT_EN for LVDS failsafe	For information, see Dynamic ODT or Fail-Safe LVDS.
Enable RXD_ODT_EN for LVDS failsafe	For information, see Dynamic ODT or Fail-Safe LVDS.
Expose Extra Training Control Ports	When selected, this option adds advanced diagnostic ports on IOD core to connect fabric for the purpose of analyzing and debugging the I/O interfaces. See Table 10-4 for information.
Enable delay line simulation	When selected, enables the delay line simulation.
Training Direction for BclkSclk	• Forward: BCLKSCLK training performed in the forward direction. • Backward: BCLKSCLK training performed in the backward direction. This is option is enabled by default. **Note:** This parameter is used only for PLL-based clock training. The change in direction can mitigate potential divider pulse glitches as the backward direction lengthens the pulse width, thus reducing the likelihood of glitch occurrences.

9.2.3. Dynamic Delay Control (Ask a Question)

Dynamic receiver delay controls are exposed on the IOD component by enabling it in the IOD configurator. On the IOD configurator -> **Advanced** (tab) -> **Debug** (pane), select the **Expose dynamic delay control** checkbox to add ports as shown in Figure 9-3. These ports are automatically exposed when selecting any of the RX_DDRX_DYNAMIC interfaces.

Table 9-4. Dynamic Delay Control Ports

Port	I/O	Description
DELAY_LINE_MOVE	Input	Change delay setting on rising edge
DELAY_LINE_DIRECTION	Input	Direction of delay setting change
DELAY_LINE_LOAD	Input	Asyn. Reload flash settings for delay

Table 9-4. Dynamic Delay Control Ports (continued)

Port	I/O	Description
DELAY_LINE_OUT_OF_RANGE	Output	Delay setting has reached max or min range. The delay_line_load signal asynchronously reloads the initial static flash bit delay settings. The delay_line_move signal is a pulse and changes the delay setting by ±1 increment each time it is pulsed according to the delay_line_direction signal value. "1" increases up the delay setting by one increment "0" decreases down the delay setting by one increment When the delay setting reaches the minimum value or the maximum value of the delay chain, the delay chain controller generates an delay_line_out_of_range output to indicate that it has reached the end of the delay chain. The delay setting stops at this min or max setting, even if the delay_line_move signal is still pulsing.
EYE_MONITOR_EARLY[n:0]	Output	The EYE_MONITOR_EARLY asserts if the data edge is close to the clock edge on the early side of clock. This flag indicates that the delay settings must be moved down.
EYE_MONITOR_LATE[n:0]	Output	The EYE_MONITOR_LATE asserts if the data edge is close to the clock edge on the late side of clock. This flag indicates that the delay setting must be moved up.
EYE_MONITOR_CLEAR_FLAGS[n:0]	Input	Use the EYE_MONITOR_CLEAR_FLAGS input signal to clear the "early" and "late" flags. This signal is from the fabric and indicates that the delay chain settings is incremented or decremented as a function of the previous flag settings.
EYE_MONITOR_WIDTH[2:0]	Input	Use the input signals "EYE_MONITOR_WIDTH<2:0>" to programably set a minimum delay space requirement between the data edges and the clock edges. The programmable delay settings are programmed in delay increments of 1, 2, 3, 4, 5, 6, or 8. This delay setting is between the clock edge and the data edge. This delay setting is then used to generate flags if the data edges are closer to the clock edges than the minimum setting. By allowing these signals to be dynamically controlled from the core, you can determine the relative size of the eye opening.

Important: EYE_MONITOR_* ports are not available for RX_DDR interfaces with a gearing ratio of 1.

For statically delayed RX_DDR_G_A/C interfaces, using static IOD delay tap values instead of using dynamic IOD interface training, the **Advanced** tab of the IOD configurator allows to enter a programmable input delay on the input clock. The programmable delay for the input data signals can be specified in the Libero SoC I/O Editor or in a user I/O `.pdc` constraints file. However, note that whenever the **Expose Dynamic Delay Control** setting is enabled in the IOD configurator, the corresponding programmable I/O delay tap for the data inputs is set to 1 by default. This applies to all cores using dynamic IOD delay controls such as Generic IOD TX/RX, DDR memory, QDR memory, Octal SPI Flash PHY, and so on. Exposing the dynamic delay controls also prevents from specifying programming input delay values using the I/O Editor or a user `.pdc` I/O constraint file. In other words, adding IOD data programmable input delay taps in the I/O Editor or user `.pdc` constraints is only allowed for statically delayed IODs where the entire dynamic control interface (DELAY_LINE_* ports) is disabled.

9.2.4. IOD Generic TX (Ask a Question)

The following table lists the transmit interface software names and their related data.

Table 9-5. Transmit Interface

Software Name	Ratio	Clock to Data Relationship	I/O Clock	Fabric Clock	Max Data Rate (Mbps)
TX_DDR_G_A	1	Aligned	Global	Global	500
TX_DDR_G_C	1	Centered	Global	Global	500
TX_DDR	1	No forwarded clock	Global	Global	500

Table 9-5. Transmit Interface (continued)

Software Name	Ratio	Clock to Data Relationship	I/O Clock	Fabric Clock	Max Data Rate (Mbps)
TX_DDRX_B_A	2, 3.5, 4, 5	Aligned	High-speed I/O Clock	Global	1000, 1600, 1600, 1600
TX_DDRX_B_C	2, 3.5, 4, 5	Centered	High-speed I/O Clock	Global	1000, 1600, 1600, 1600
TX_DDRX_B	2, 3.5, 4, 5	No forwarded clock	High-speed I/O Clock	Global	1000, 1600, 1600, 1600

The following figure shows the IOD Generic Transmit Interfaces configurator.

Figure 9-4. IOD Generic Transmit Interfaces—Configuration Tab

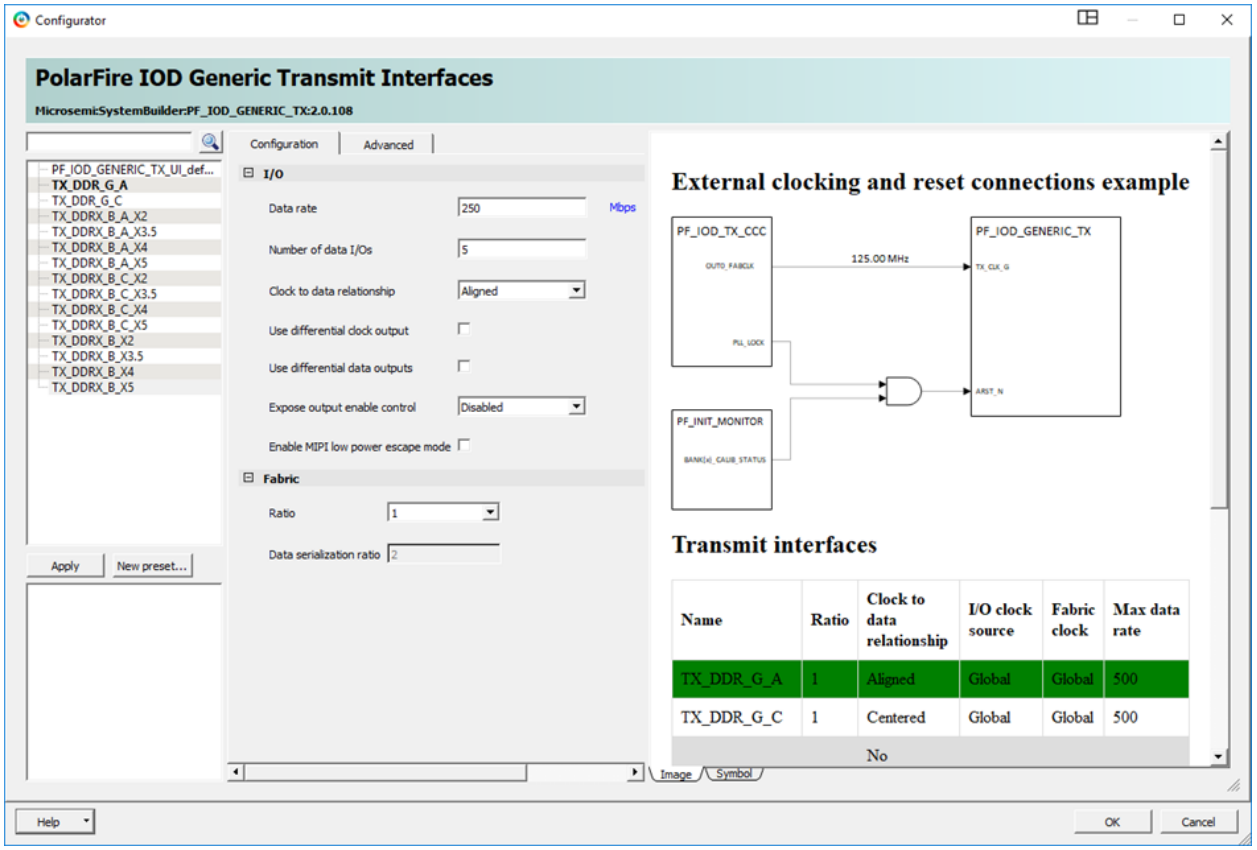


Table 9-6. IOD Generic Transmit Interfaces—Configuration Tab

GUI Option	Selections
Data rate	User Input[1]
Number of data I/Os	User Input
Clock to data relationship	Aligned, Centered, No Forwarded Clock
Use differential clock output	Disable (single-ended) and Enabled (differential)
Use differential data outputs	Disable (single-ended) and Enabled (differential)
Enable MIPI low power escape mode	Enable/Disable
Ratio	1, 2, 3.5, 4, 5
Data Serialization Ratio	Derived from the ratio setting

(1) See Transmit Interface (right panel) for valid data rates (Figure 9-4).

Figure 9-5. IOD Generic Transmit Interfaces—Advanced Tab

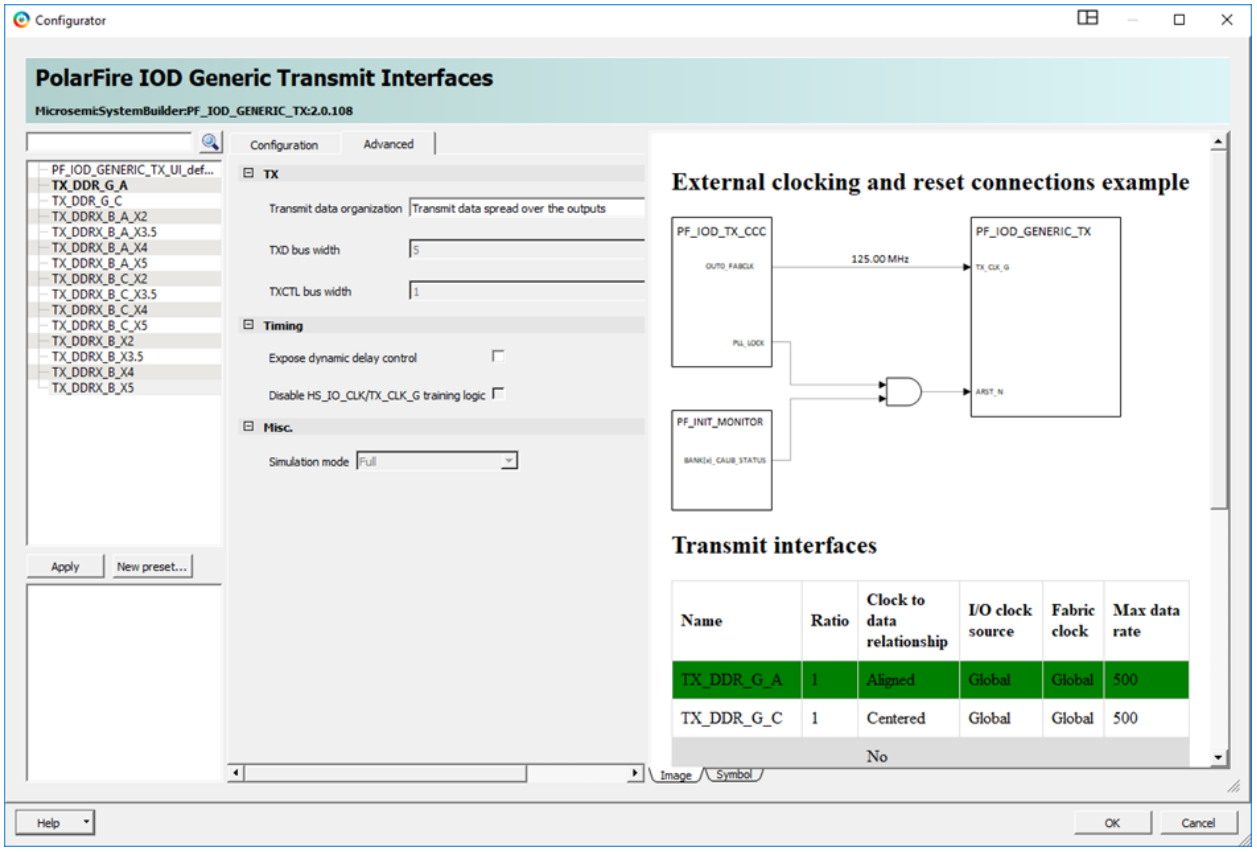


Table 9-7. IOD Generic Transmit Interfaces—Advanced Tab

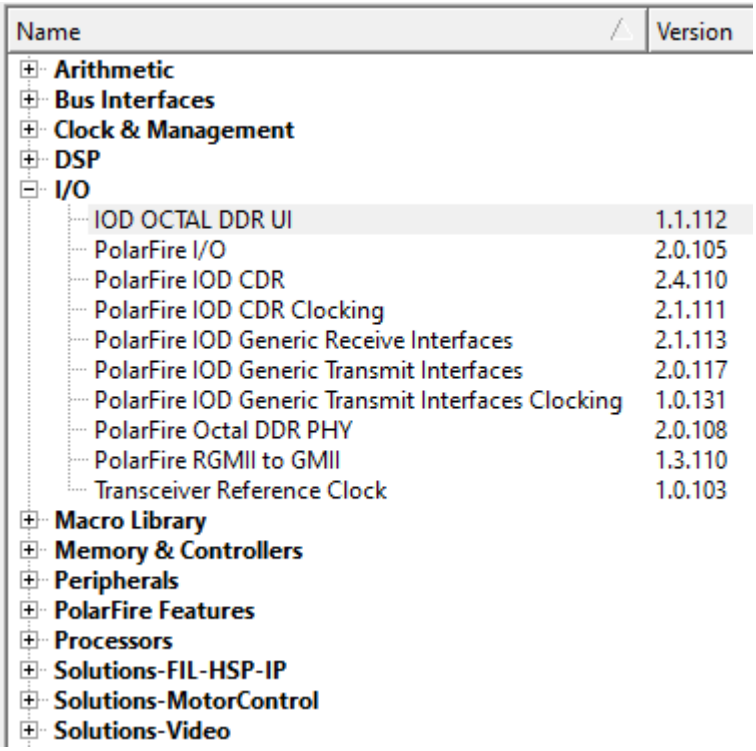
GUI Option	Selections
Transmit data organization	• Transmit data spread over outputs with data/Control split: X I/Os are associated to data and N-X I/Os to control that means it is arranged to have two data buses on the fabric side (one for data and one for control). • Transmit data spread over outputs: On the fabric, it is arranged as one wide bus. This arrangement means all rising edge of all I/Os, all falling edge of all I/Os, all rising edge+1 of all I/Os, and so on. • Transmit data independent over outputs: It is one data bus per I/O.
TXD bus width	This allows organizing the splitting of the data bus.
TXCTL bus width	This allows organizing the splitting of the data bus.
Expose dynamic delay control	Disable and Enable. See Table 9-4.
Simulation mode	Full
Enable User Control of Clock Pattern	When enabled, this provides two top-level input ports to the TX IOD macro, which can be used to provide option to transmit parallel clock synchronously with data.
PF_IOD_TX_CCC	A dedicated PF_IOD_TX_CCC is specifically incorporated in the IOD transmit solution. When using a TX_CCC for multiple IOD generic TX, it requires two IOD generic TX components. • One component with the training logic that is instantiated only once. And • another component, with the training logic turned off. This option is available in the **Advanced** tab of the PF IOD Generic Transmit Interface UI. The second component can be instantiated as required.

Important: Post layout simulation of the IOD is not supported.

9.3. Basic I/O Configurator - PF_IO (Ask a Question)

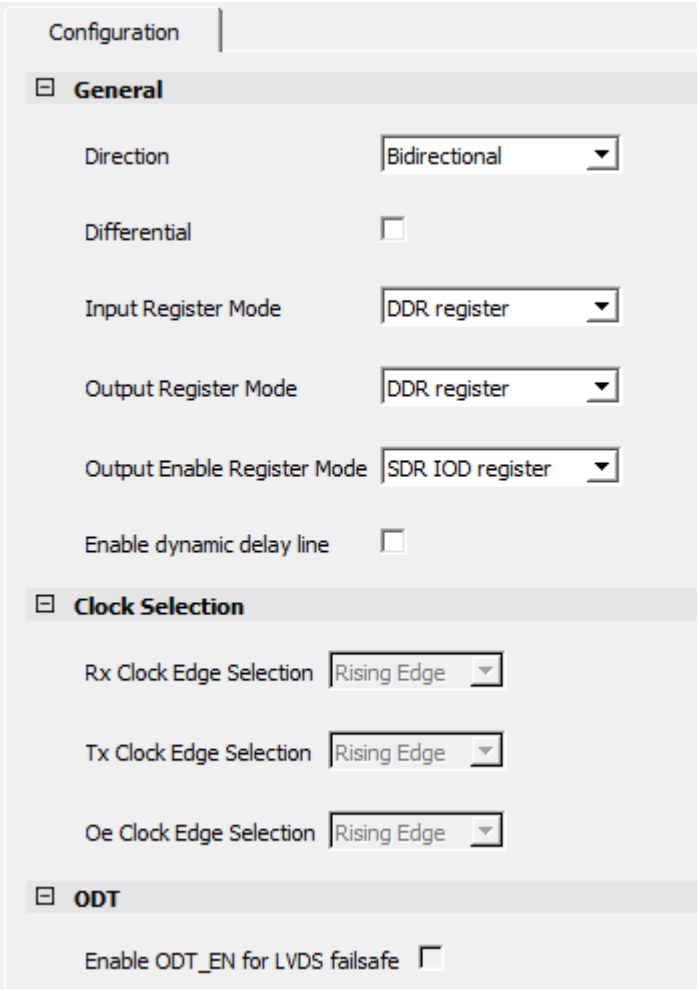
A basic I/O configurator is available in the Libero SoC catalog. It is capable of building simple I/O macros. For information about I/O macros, see PolarFire FPGA and PolarFire SoC FPGA Macro User Guide.

Figure 9-6. I/O Configurator



The I/O configurator uses a single tab GUI for configuring the I/O component. The GUI includes a symbol depiction of the macro as configured by the user.

Figure 9-7. I/O Configuration Tab



The Direction pull-down allows selection of Bidirectional, Input, Output, and Tribuf. It has a checkbox for selection of single-ended or differential I/O. The configurator does not provide the capability to choose a specific I/O standard. You must use the IO Editor or PDC to pick an associated single-ended or differential standard.

A Register Mode pull-down allows selections of non-registered, SDR registered, or DDR registered interfaces. Non-registered modes generate simple I/O buffer components. Registered modes construct simple registered interfaces by adding SDR or DDR resources to the input, output, or bidirectional. This capability is for simple DDR applications. With the I/O Configurator, DDR modes uses IOD elements to construct DDR1X configurations without the low-skew clock management capabilities. For information about DDR1X waveform, see Figure 8-15 and Figure 8-22. For source-synchronous designs, you must target IOD interfaces, which includes low-skew clock management. See I/O Interface Configurators.

The **Enable dynamic delay line** check box selection adds the capability to control the delay chain structure in the input or output paths. By default, this is not enabled. The fast path from the input or output buffer is used. When enabled (checked), the component includes the delay logic and controls for fabric hosted IP to control the tuning of the path.

The **Clock Selection** pane allows the user to select clock inversion for the Input, Output, and Output Enable registers in the SDR register mode. In the **Clock Selection** pane, the **Rx Clock Edge Selection**, **Tx Clock Edge Selection**, and **Oe Clock Edge Selection** drop-down options are provided to select data sampling on either rising edge or falling edge of the clock.

The **Enable ODT_EN for LVDS failsafe** check box exposes an enable port to differential input macros. This enable pin is used in conjunction with the capability to dynamically enable/disable the ODT resistor when needed for applications such as fail-safe LVDS.

9.4. I/O Interface Timing Constraints (Ask a Question)

Libero SoC is capable of generating SDC timing constraints for design components used in IOD interfaces. These derived SDC constraints are based on the configuration of the IOD blocks including the sub-blocks required for the specific IOD functionality. The derived SDC constraints are placed in the `<root>_derived_constraints.sdc` file.

For static IOD Rx and Tx interfaces, static timing analysis can be performed using the auto-generated derived constraints. Dynamic IOD Rx interfaces use a training operation on hardware to adapt the I/O timing to the PCB characteristics. For this, the derived SDC uses a wider range of delay and which cannot be used to accurately perform timing analysis of external setup/hold timing of IOD generic Rx when configured in Dynamic mode.

When using IOD Rx and Tx, use derived constraints for timing and physical constraints. Based on the timing violations from SmartTime, define the appropriate tap delays (IN_DELAY for IOD Rx and OUT_DELAY for IOD Tx) using the **I/O Attribute Editor**, and use the generated PDC file for Place and Route. Ensure that there are no timing violations in the SmartTime before programming the device. User must define the tap delays as required.

10. Protocol-Specific I/O Interfaces (Ask a Question)

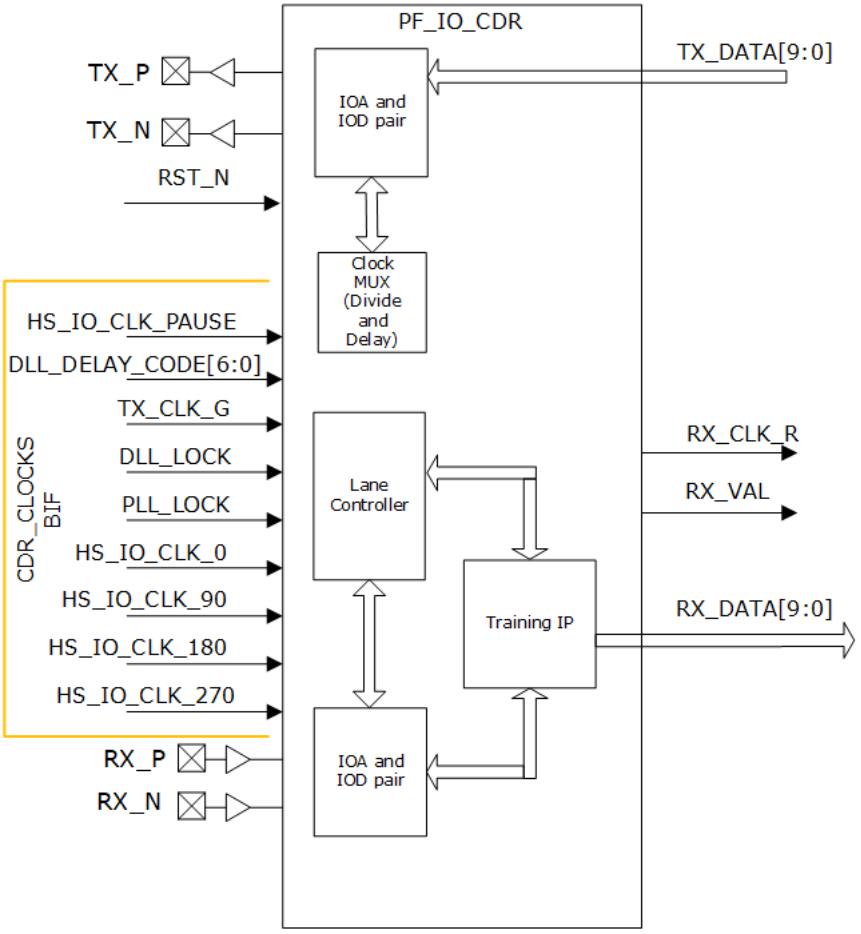
This section describes about protocol-specific I/O interfaces.

10.1. PF_IOD_CDR (Ask a Question)

The PF_IOD_CDR interface provides an asynchronous receiver and a transmit interface for serial data transfers. This interface can support up to 1 GbE transfers. It supports serial protocols and other similar encoded serial protocols. PF_IOD_CDR uses a 10:1 digital ratio to provide a 10-bit data and clock interface for both transmit and receive modes. In the receive mode, the clock recovery circuit is used in the lane controller to generate the recovered clock. The PF_IOD_CDR interface is compatible with CoreTSE, CoreTSE_AHB, and CoreSGMII configured in TBI mode. For information about reference design using PF_IOD_CDR, see AN4623: PolarFire FPGA 1G Ethernet Loopback Using IOD CDR (Earlier DG0799).

The following illustration shows the PF_IOD_CDR transmit and receive interface.

Figure 10-1. PF_IOD_CDR Transmit and Receive Interface Modes

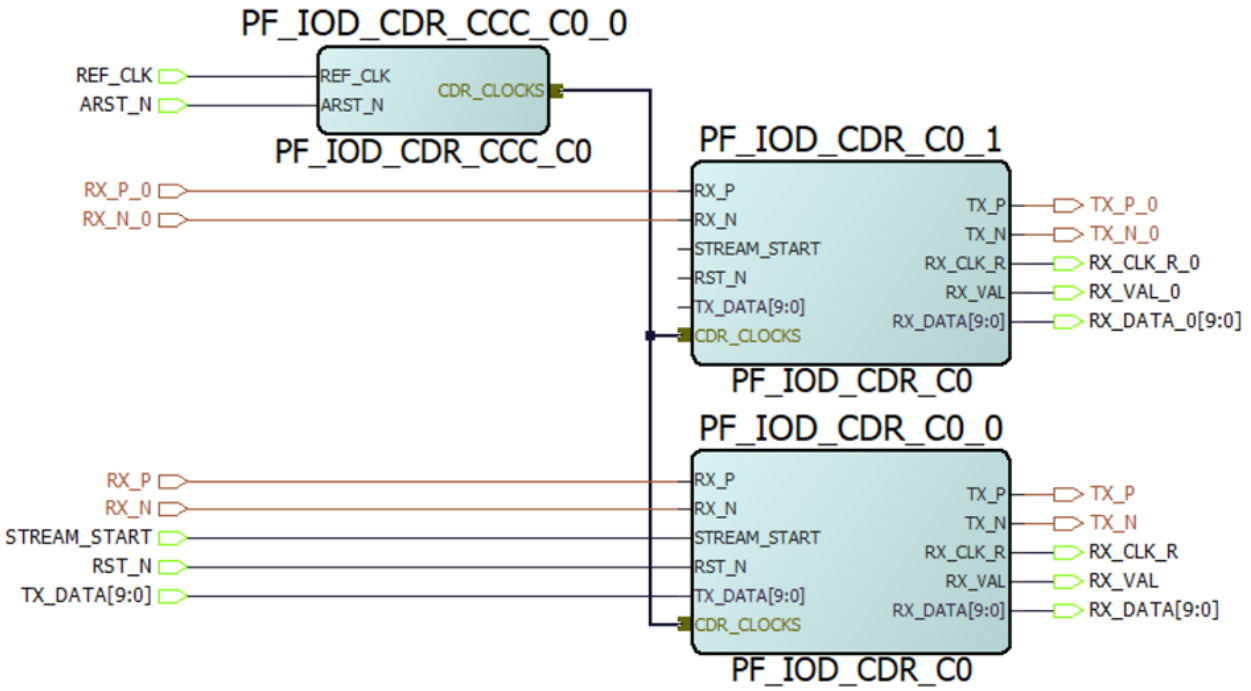


The IOD_CDR solutions requires two purposes built IP cores.

- PF_IOD_CDR
- PF _IOD_CDR_CCC

These two cores permit master and slave sharing. A BIF is available to connect the clock outputs from PF_IOD CDR CCC to PF_IOD CDR.

Figure 10-2. SmartDesign of IOD_CDR Topology



10.1.1. IOD CDR (Ask a Question)

The following figure shows the IOD CDR configurator.

Figure 10-3. IOD CDR Configuration

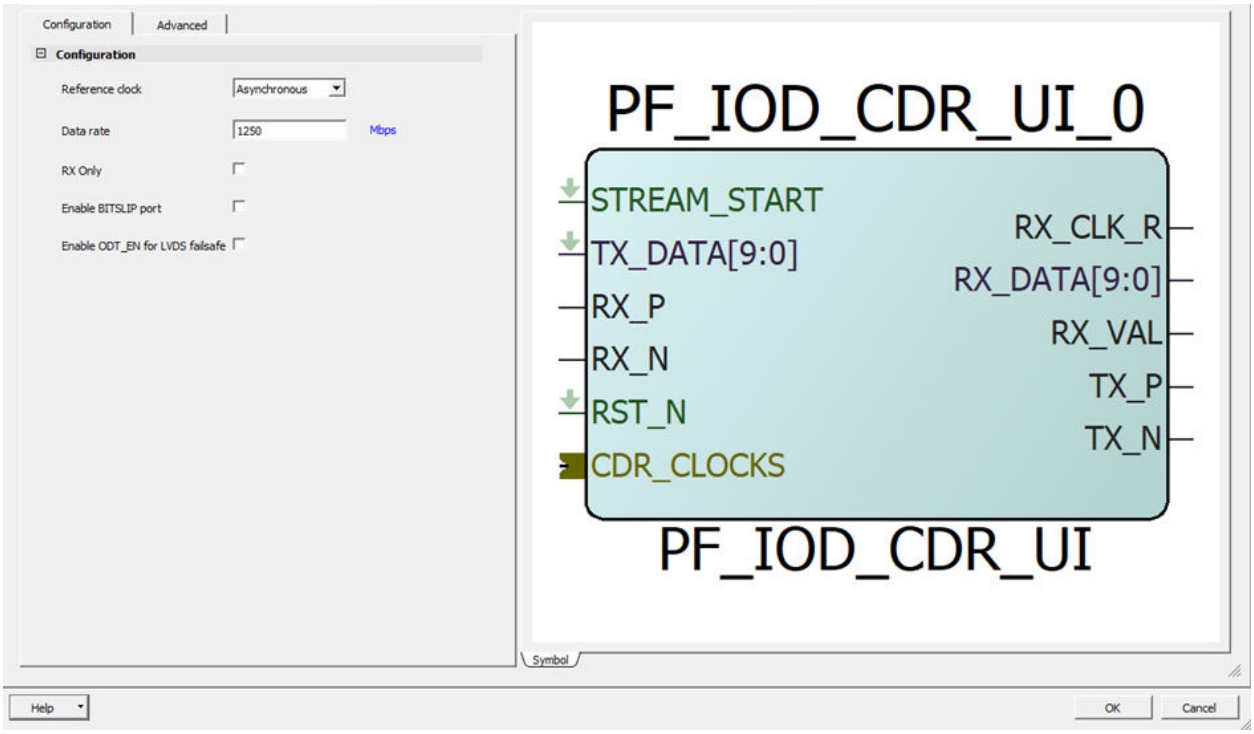


Table 10-1. IOD CDR Configuration

GUI Option	Selections
Data rate	User Input – 1250 Mb/s maximum
Reference Clock	Asynchronous (300 PPM Rx/Tx clocking)/Synchronous (0 PPM Rx/TX clocking) (Default Asynchronous)
RX Only	RX Only (when checked)
Enable BITSLIP port	Disabled and Enabled

10.1.2. Receive Interface (Ask a Question)

The PF_IOD_CDR receive interface uses four high-speed bank clocks and generates the recovered clock. The lane controller in the IOD includes a clock recovery block. It uses the incoming data and the four bank clocks and generates RX_CLK_R, also known as DIVCLK. The downstream IP or logic uses this clock. The serial data is received on an IOA pair and sent to the associated IOD block. The IOD block uses a 10:1 digital ratio. The IOD block uses the recovered clock to capture the serial data stream to the core.

The CDR requires four phases of the HS_IO_CLK running at half the frequency of the serial data rate. The RX_CLK_R into the fabric includes jitter from the switching of the phase, which creates this clock.

10.1.3. Transmit Interface (Ask a Question)

The PF_IOD_CDR transmit interface converts the parallel data into a serial data stream using the IOD interface. It receives the parallel data TXD[9:0] and transmits it through the I/O ports such as TX_P and TX_N. The PF_IOD_CDR transmit interface uses the same PLL used in the receive interface. The transmit clock generated is connected to the pin TX_CLK_G of the PF_IOD_CDR. The source clock is connected to HS_IO_CLK_0.

The following table shows the PF_IOD_CDR interface associated ports.

Table 10-2. PF_IOD_CDR Interface Associated Ports

Port	I/O	BIF	Description
HS_IO_CLK_0[1]	Input	CDR_CLOCKS	Bank clock with phase 0 is used for both receive and transmit interface. Frequency must be half the rate of the serial data input.
HS_IO_CLK_90[1]	Input	CDR_CLOCKS	Bank clock with phase 90 is used for the I/O clock recovery. Frequency must be half the rate of the serial data input.
HS_IO_CLK_180[1]	Input	CDR_CLOCKS	Bank clock with phase 180 is used for the I/O clock recovery. Frequency must be half the rate of the serial data input.
HS_IO_CLK_270[1]	Input	CDR_CLOCKS	Bank clock with phase 270 is used for the I/O clock recovery. Frequency must be half the rate of the serial data input.
PLL_LOCK	Input	CDR_CLOCKS	Lock signal from CCC-PLL.
HS_IO_CLK_PAUSE	Input	CDR_CLOCKS	Toggling the HS_IO_PAUSE: – Resets the IOD RX state machines. This reset re-synchronizes pattern to HS_IO_CLK (bank clock) and RXCLK. – Resets any adjustment done through SLIP operation. – Resets the IOD TX state machines. This reset synchronizes HS_IO_CLK and TXCLK. – HS_IO_PAUSE does not disrupt delay line value settings.
DLL_LOCK	Input	CDR_CLOCKS	Lock signal from CCC-DLL.
TX_CLK_G	Input	CDR_CLOCKS	Transmit clock from the Fabric.
DLL_DELAY_CODE[6:0][2]	Input	CDR_CLOCKS	Delay code bus input from DLL-CCC. DLL delay code for 90° phase of the data.

Table 10-2. PF_IOD_CDR Interface Associated Ports (continued)

Port	I/O	BIF	Description
DLL_VALID_CODE	Input	CDR_CLOCKS	Delay code valid input from Master IO_CDR Lane.
CDR_START	Input	CDR_CLOCKS	Start signal from the Master IO_CDR Lane.
STREAM_START[3]	Input	—	High input indicates valid serial input stream. STREAM_START signals to the CDR locks to a valid incoming serial data stream. This signal must not be tied high. It must be controlled to go high to indicate the incoming data stream is valid. This is extremely important at start-up or power-up.
TX_DATA[9:0]	Input	—	Transmit parallel data.
ODT_EN	Input	—	On Die Termination Enable Input. Optional pin is used with LVDS Fail Safe operation. See Dynamic ODT or Fail-Safe LVDS for information.
RX_P	Input Pad	—	Serial data input (P side).
RX_N	Input Pad	—	Serial data input (N side).
RX_BIT_SLIP[4]	Input	—	This port is used to rotate the parallel data word from the IOD to match the proper alignment of the data per lane.
RST_N[5]	Input	—	Active asynchronous low reset input.
RX_CLK_R	Output	—	Recovered clock for the fabric interface is divided by five from the HS_IO_CLK. This clock is routed using a regional clock.
RX_VAL	Output	—	The CDR is locked to the incoming serial data after indication by STREAM_START that the incoming data is valid. When the IO_CDR locks to the data, indicated by RX_VAL going high, any disruptions of data stream will not cause RX_VAL to change.
TX_P	Output Pad	—	Serial data output (P side).
TX_N	Output Pad	—	Serial data output (N side).

(1) PLL takes any reference clock input frequency (default 125 MHz) and outputs 625 MHz clock with 0, 90, 180, and 270 degree shift on four outputs.
(2) DLL takes 625 MHz reference clock input from the PLL output in Clock Reference Mode and outputs delay code as quarter of the clock cycle. The delay code is used in calculating of fine tune delay of CDR clock phase.
(3)For more information, see AN4623 REVB (or later), which provides an instance of a valid rx stream detection implemented in the SSDetect block.
(4) User optional pin enabling the BITSLIP exposes the Lx_BIT_SLIP.
(5)Resets the IOD block of the IOCDR. Does not reset DLL.

Table 10-3. Advanced Tab Options

GUI Option	Selections
Jump Size Step	Do not change default
Expose Diagnostic Ports	When checked, ports expose. (see Table 10-4)

Table 10-3. Advanced Tab Options (continued)

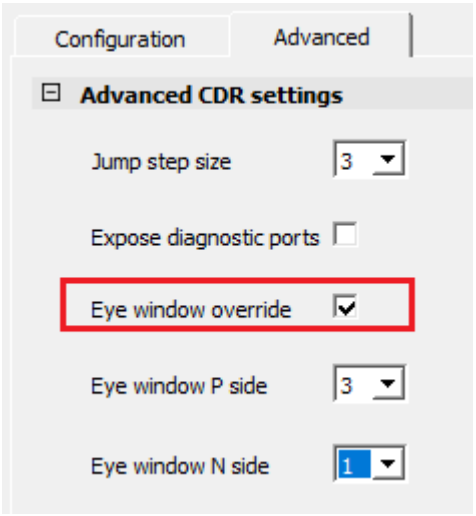
GUI Option	Selections
Eye window override	This option allows the user to customize the width of the Eye Monitor at the Receive end (RX_P and RX_N). Once the option is selected, the user can customize the value of **Eye window P side** and **Eye window N side** within the range of 1–7 as shown in the following figure. **Figure 10-4.** Eye Window Override Option  **Note:** For data rates lower than 1 Gbps, the default eye window sizes of 5 and 6 might not be sufficient for data streams with high jitter. In this case, the **Eye window override** option can be used to increase the width of the Eye Monitor for achieving better performance.
Flag Window Size	Do not change default

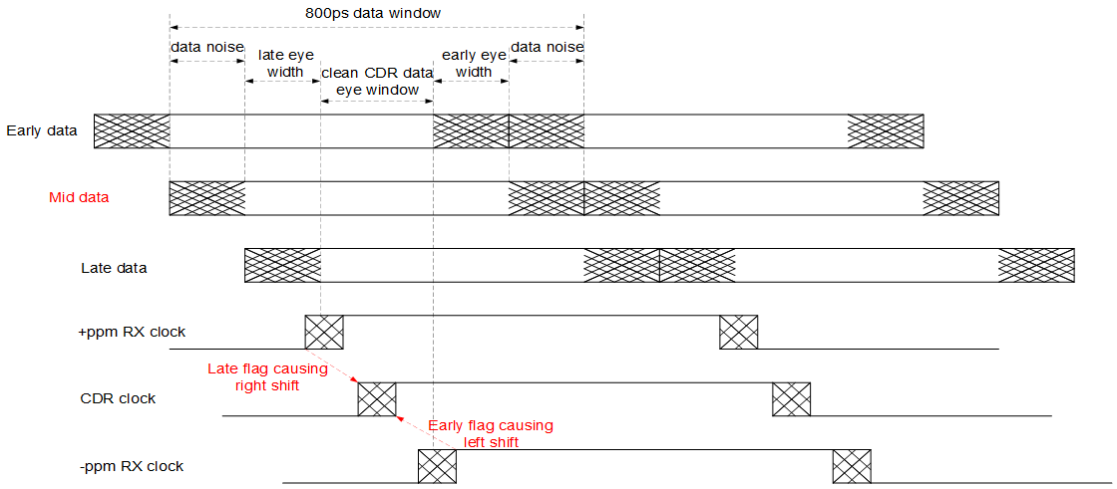
Table 10-4. Advanced Diagnostic Ports

Port	I/O	Description
SELA_LANE[10:0]	Output	SELA/SELB bits [10:0] toggles when the internal CDR clock is switched from delay line A to B or vice versa. Bits[9:8] - Course phase selection for CDR clock (0, 90, 180, 270). Bits [7:0] - Fine phase selection for the CDR clock, 25pS delay taps (0....90) 25pS is typical (not PVT calibrated). See respective device family datasheet for more information about range.
SELB_LANE[10:0]	Output	
EARLY_N	Output	EYE_MONITOR_EARLY and EYE_MONITOR_LATE flag outputs indicate whether the data edges are closer to the clock edges than this minimum setting.
LATE_N	Output	
CDR_READY	Output	Output asserts when CDR is locked and stays high until reset.
SWITCH_LANE	Output	Ports toggle when there is a clock phase shift. Accompanying with EARLY/LATE flags; this indicates if the phase shift is increasing or decreasing the delay line when SWITCH_LANE output asserts. If both flags are high with SWITCH_LANE high, there is clock jitter and/or causing data errors.
CLR_FLAGS_N	Input	Port is used to clear the flags and restarts the early/late monitor after each time the status is recorded.

Important: 25 pS is typical and not PVT calibrated. For range information, see respective PolarFire FPGA Datasheet or PolarFire SoC FPGA Datasheet.

The following figure shows the IOCDR Flags.

Figure 10-5. IOD CDR Flags



The advanced diagnostic ports are intended for monitoring IOD operation and for interface debugging. Users can monitor these flags with counters for analyzing the operation of the IOD training.

10.1.4. IOD CDR Clocking (Ask a Question)

A dedicated CCC is generated by Libero SoC to support IOD CDR interfaces. PF_IOD_CDR_CCC configurator provides the options to generate the module. The REF_CLK input is required based on the Data rate and CCC PLL clock multiplier. The PF_IOD_CDR_CCC does not allow divider control for the divider generating TX_CLK_G because the IOCDR only works in ratio 5 and is preset by Libero SoC. Libero SoC provides the proper timing constraints (as shown) when derived in the constraints manager.

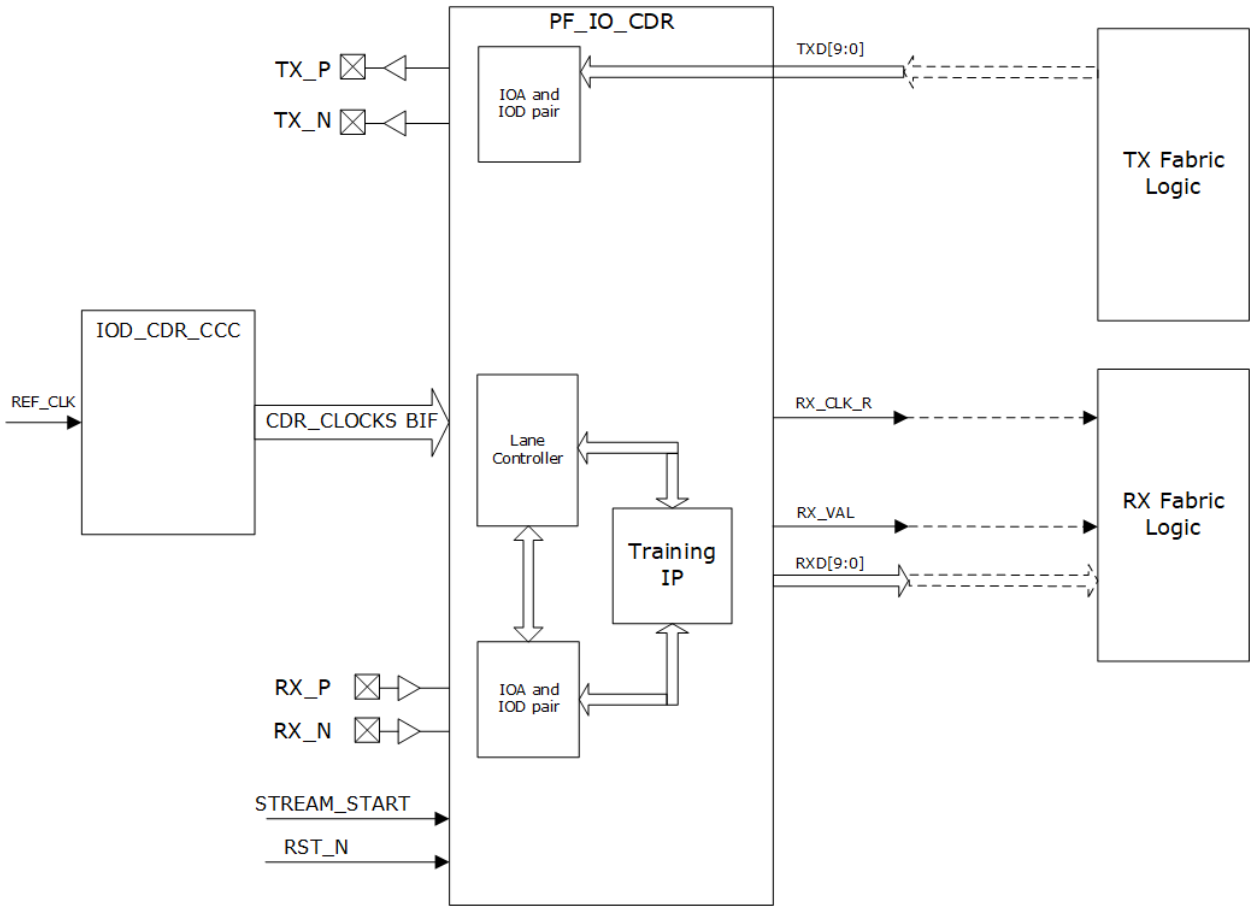
```
create_generated_clock -name {PF_IOD_CDR_CCC_C0_0/PF_CLK_DIV_0/I_CD/Y_DIV} -edges {1 7
11} -source [ get_pins { MY_DESIGN/PF_IOD_CDR_CCC_C0_0/PF_CLK_DIV_0/I_CD/A } ] [ get_pins
{ MY_DESIGN/PF_IOD_CDR_CCC_C0_0/PF_CLK_DIV_0/I_CD/Y_DIV } ]
```

10.1.4.1. HS_IO_CLK Generation Using PF_IOD_CDR_CCC (Ask a Question)

The PF_IOD_CDR receive interface is sourced by a single PLL driving four bank clocks of 0, 90, 180, and 270 degrees running at the data rate. PF_IOD_CDR_CCC is available in the Libero SoC IP catalog. The PF_IOD_CDR transmit interface uses fabric clock on OUT0 port of the PLL and generates the transmit clock.

The following illustration shows the PF_IOD_CDR interface connected to the IOD_CDR_CCC and fabric logic.

Figure 10-6. Using PF_IOD_CDR Interfaces

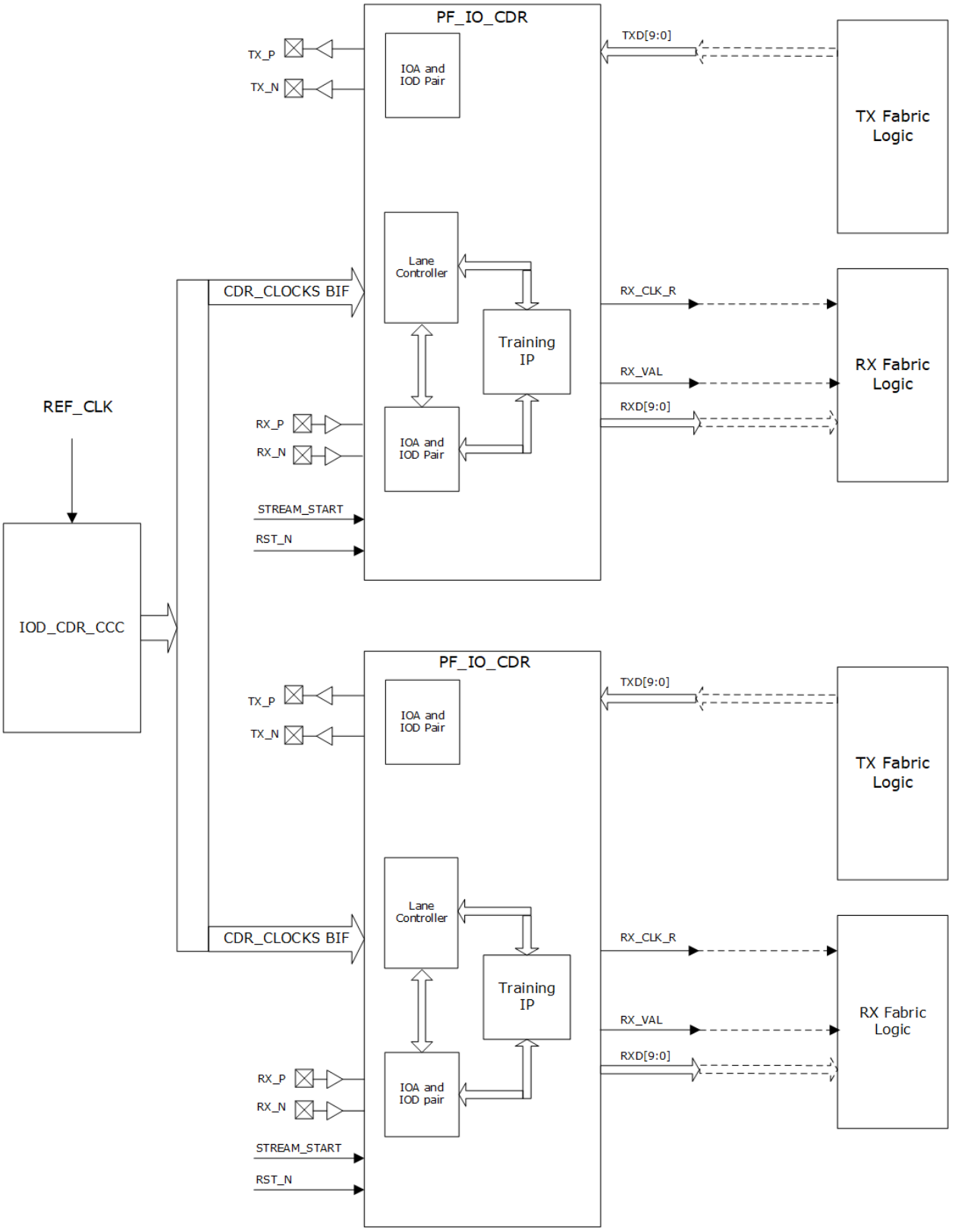


10.1.5. Clock Sharing (Ask a Question)

The same PLL is shared between the PF_IOD_CDR receive and transmit interfaces, as shown in Figure 10-7. In addition, multiple PF_IOD_CDR interfaces can share the same PLL on the adjacent vertical and horizontal edges. For instance, the PLL_SW_0 interface can drive the PF_IOD_CDR interface on the southern and western edges (see PolarFire Family I/O Banks).

The following illustration shows multiple PF_IOD_CDR transmit and receive interfaces.

Figure 10-7. Multiple PF_IOD_CDR Transmit and Receive Interfaces



10.1.5.1. Interface Selection Rules (Ask a Question)

Follow these rules when assigning a pin for the PF_IOD_CDR interface:

- One differential input IOA, one differential output IOA.
- Four IOD associated with IOA, one floating IOD.
- The floating IOD is placed in the N side IOD site with the function DQS.
- N side IOA with the function DQS cannot be used.
- One PF_IOD_CDR_CCC can be shared with multiple instances of PF_IOD_CDR as long as they are at the same data rate and placed in the same group of lanes.
- PF_IOD_CDR_CCC uses one PLL, one DLL and one LANECTRL.
- Transmit and receive IOA must be placed in the same lane.
- IOA from two different interfaces (TX/RX/DDR/QDR/OCTAL/CDR) cannot be placed in the same I/O lane.

10.1.5.2. Full Duplex 1GbE and SGMII IOCDR (Ask a Question)

Full duplex 1GbE and SGMII IOCDR are supported in GPIO/HSIO banks and permit only one per lane. The following table lists the IGbE and SGMII IOCDR per Device/Package for PolarFire FPGA.

Table 10-5. 1GbE and SGMII IOCDR Per Device/Package

Type/Size/Pitch	1GbE/SGMII IOCDR per Device/Package			
	MPF100	MPF200	MPF300	MPF500
FCSG325 (11x11, 11x14.5*, 0.5 mm)	7	7	—	—
FCSG536 (16x16, 0.5 mm)	—	15	15	—
FCVG484 (19x19, 0.8 mm)	13	13	13	—
FCG484 (23x23, 1.0 mm)	12	12	12	—
FCG784 (29x29, 1.0 mm)	—	18	18	18
FCG1152 (35x35, 1.0 mm)	—	—	19	19

The following table lists the IGbE and SGMII IOCDR per Device/Package for PolarFire SoC FPGA.

Table 10-6. 1GbE and SGMII IOCDR Per Device/Package

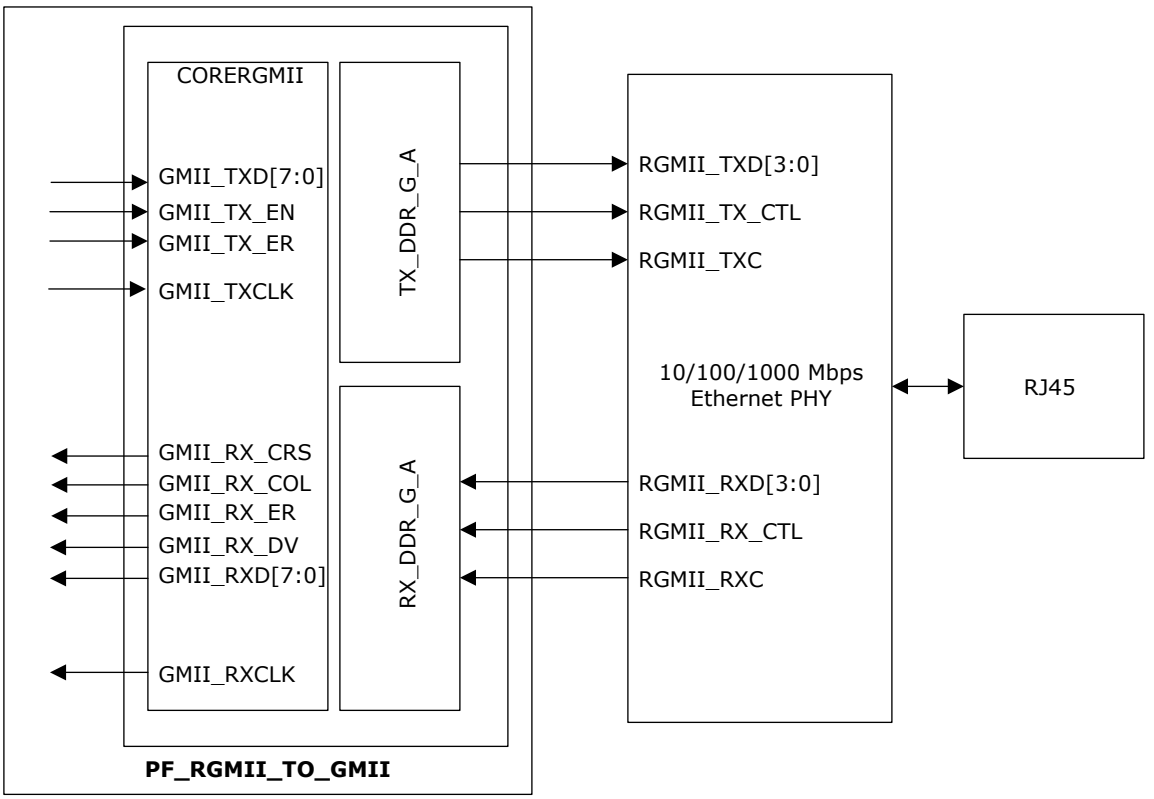
Type/Size/Pitch	1GbE/SGMII IOCDR per Device/Package
	MPFS250
FCSG325 (11x11, 11x14.5*, 0.5 mm)	
FCSG536 (16x16, 0.5 mm)	9
FCVG484 (19x19, 0.8 mm)	7
FCG484 (23x23, 1.0 mm)	
FCG784 (29x29, 1.0 mm)	15
FCG1152 (35x35, 1.0 mm)	19

(1) There are always two SGMII lanes available from MSS SGMII Bank.

10.2. RGMII to GMII Converter (Ask a Question)

Reduced Gigabit Media Independent Interface (RGMII) is a standard interface, which helps in reducing the number of signals required to connect a PHY to a MAC. RGMII to GMII converter provides the interface between a standard gigabit media independent interface (GMII) to RGMII conversion. The IP core is compatible with the RGMII specification v2.0 that is designed to support both the device family using the IOD blocks used with GPIO or HSIO buffers.

Figure 10-8. RGMII to GMII Block Diagram



The fifteen-signal GMII fabric interface adapts to six-signal RGMII interface by using both edges of the clock. All signals are synchronous with a 125 MHz clock signal. The RGMII data signals switch on the positive and negative edges of the clock. The two control signals are multiplexed—one arrives on the positive clock edge, the other on the negative edge. The PF_IOD_GENERIC_TX converts GMII signals (MAC side) to RGMII signals (PHY side), and the PF_IOD_GENERIC_RX converts the RGMII signals into GMII signals and passes the signals to the CoreRGMII IP block before transmission to the MAC. Externally, a 1000BASE-T Ethernet PHY is connected to RGMII through GPIO or HSIO.

See UG0687: PolarFire FPGA 1G Ethernet Solutions User Guide for more information.

The following table lists the GMII/RGMII ports and description.

Table 10-7. GMII Ports

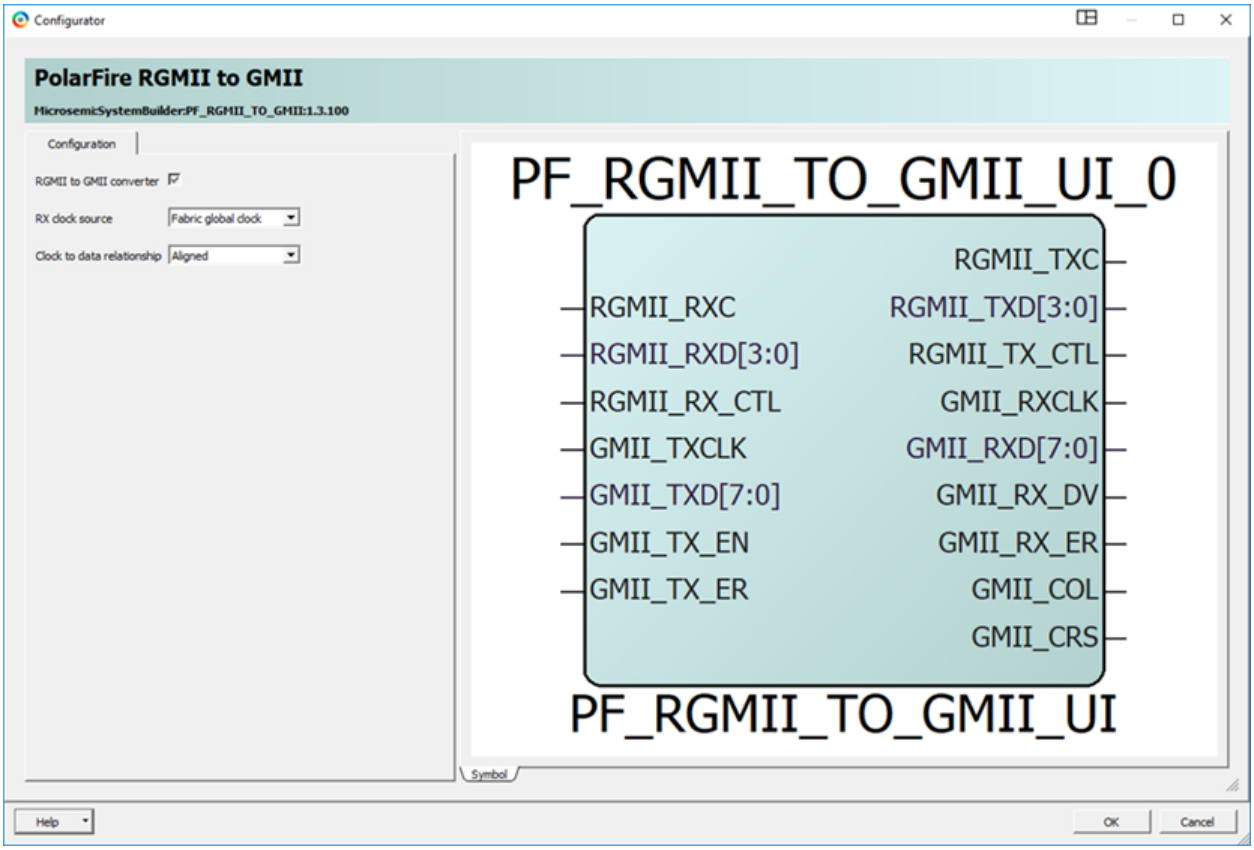
Port	I/O	Description
GMII_TXCLK	Input	Clock from fabric (GTXCLK)
GMII_TXD [7:0]	Input	GMII transmit data
GMII_TX_EN	Input	Transmit enable
GMII_RXCLK	Output	Clock to fabric depending on RX clock configurator option, either fabric global or fabric regional through the iod_generic_tx block
GMII_TX_ER	Input	Transmit error
GMII_RXD[7:0]	Output	MII receive data
GMII_RX_DV	Output	Receive data valid
GMII_RX_ER	Output	Receive error
GMII_COL	Output	Collision, considered asynchronous
GMII_CRS	Output	Carrier sense, considered asynchronous
RGMII_TXD[3:0]	Output	Transmit data to PHY

Table 10-7. GMII Ports (continued)

Port	I/O	Description
RGMII_TX_CTL	Output	Transmit Control To PHY. The TX_CTL signal carries: – GMII_TX_EN on the rising edge – TX_EN or GMII_TX_ER on the falling edge
RGMII_RXD[3:0]	Input	Receive data from PHY
RGMII_RX_CTL	Input	Receive control from PHY. The RX_CTL signal carries: – gmii_rx_dv (data valid) on the rising edge – gmii_rx_dv xor gmii_rx_er on the falling edge
RGMII_RXC	Input	RGMII receive clock
RGMII_TXC	Input	RGMII transmit clock

The following figure shows the RGMII to GMII configurator.

Figure 10-9. RGMII to GMII Configurator



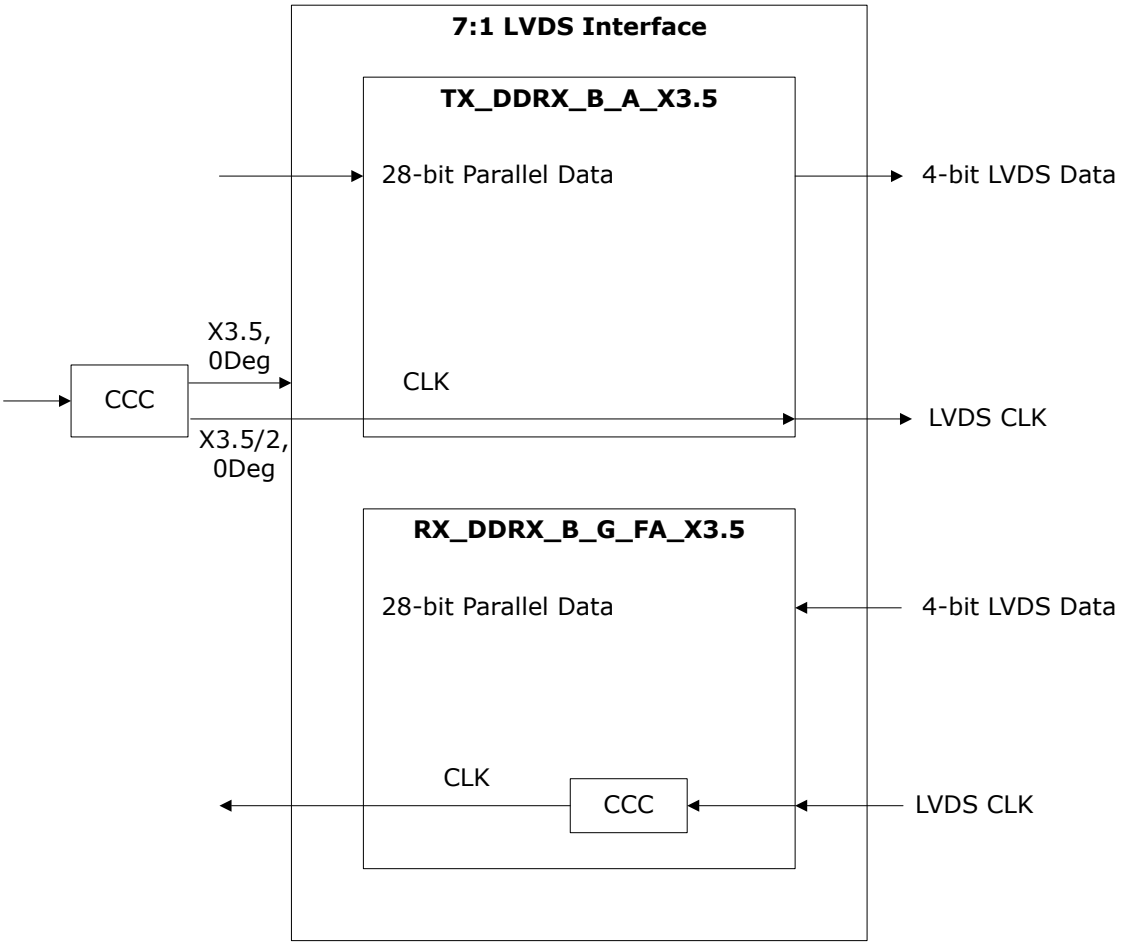
Both RX and TX IOD sub-modules are within the PF_RGMII_TO_GMII conversion module. Both blocks are pre-configured for the proper clock and data alignment and gearing ratios. You are not required to change the default setting for these modules but may need to be aware of the actual configurations for informational purposes. Designs using the PF_RGMII_TO_GMII conversion module must reference the pin selection rules discussed in Interface Selection Rules.

10.3. LVDS 7:1 (Ask a Question)

A typical source-synchronous interface application is the 7:1 LVDS video interface (used in Channel Link, Flat Link, and Camera Link). This has become a common standard in many products including consumer devices, industrial control, medical, and automotive telematics. The display interface is a source synchronous LVDS interface. Seven data bits are serialized for each cycle of the low-speed

clock. Typically, the interface consists of four (three data, one clock) or five (four data, one clock) LVDS pairs. The four pairs translate to 21 parallel data bits and five pairs translate to 28 parallel data bits.

Figure 10-10. Example of 7:1 LVDS Interface—Four Data and One Clock



10.3.1. 7:1 LVDS Receive Interface (Ask a Question)

The LVDS 7:1 receive module receives LVDS data and an LVDS clock from the FPGAs LVDS IOA inputs. The source-synchronous LVDS clock is passed to the fabric clock conditioning circuitry (CCC) block while the LVDS data is sent to the RX_DDRX_B_G_FA (fractional aligned clock and data) using 3.5 gearing ratio. The receive block uses double data rate registers to capture data on both the rising and falling edge of the input clock. RX_BIT_SLIP is not available in DDRX3P5 gearing, which requires the bit and word alignment to be part of the FPGA fabric IP. The data is deserialized to 7-bit data that is sent to the fabric with a forwarded clock.

Figure 10-11. RX_DDRX_B_G_FA Interface—Configuration Tab

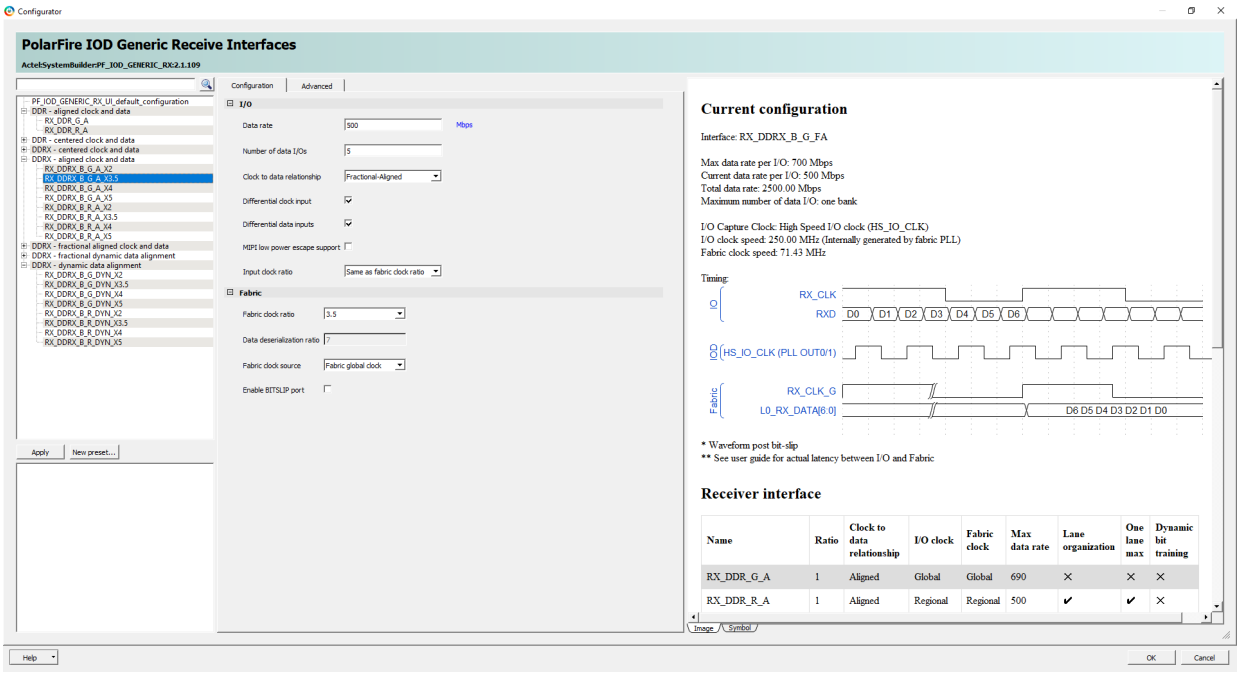
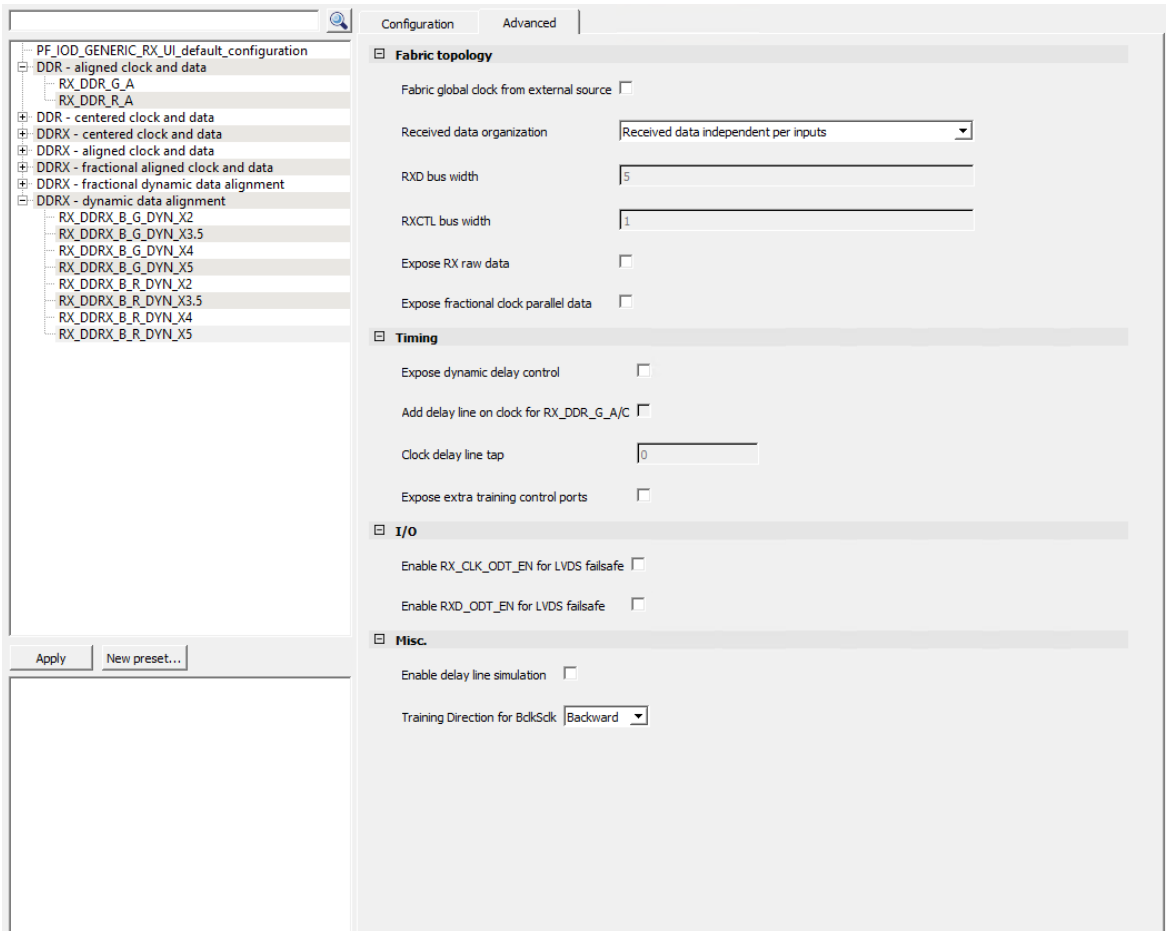


Figure 10-12. RX_DDRX_B_G_FA Interface—Advanced Tab



10.3.2. 7:1 LVDS Transmit Interface (Ask a Question)

The transmit block uses double data rate registers of the TX_DDRX_B_A_X3.5 to transmit data on both the rising and falling edges of the clock. It multiplies the parallel clock by 3.5 and uses the clock to transmit seven serial bits of data in one parallel clock cycle and serialize the data into a single LVDS data stream. HS_IO_PAUSE needs to be pulsed after the clocks are stable. This forces all gearbox to be framed in the same cycle (including the one used to generate the clk). This assures synchronization of the data word. Word starts with the rising edge of the forwarded fractional clock.

Figure 10-13. TX_DDRX_B_A_X3.5—Configuration Tab

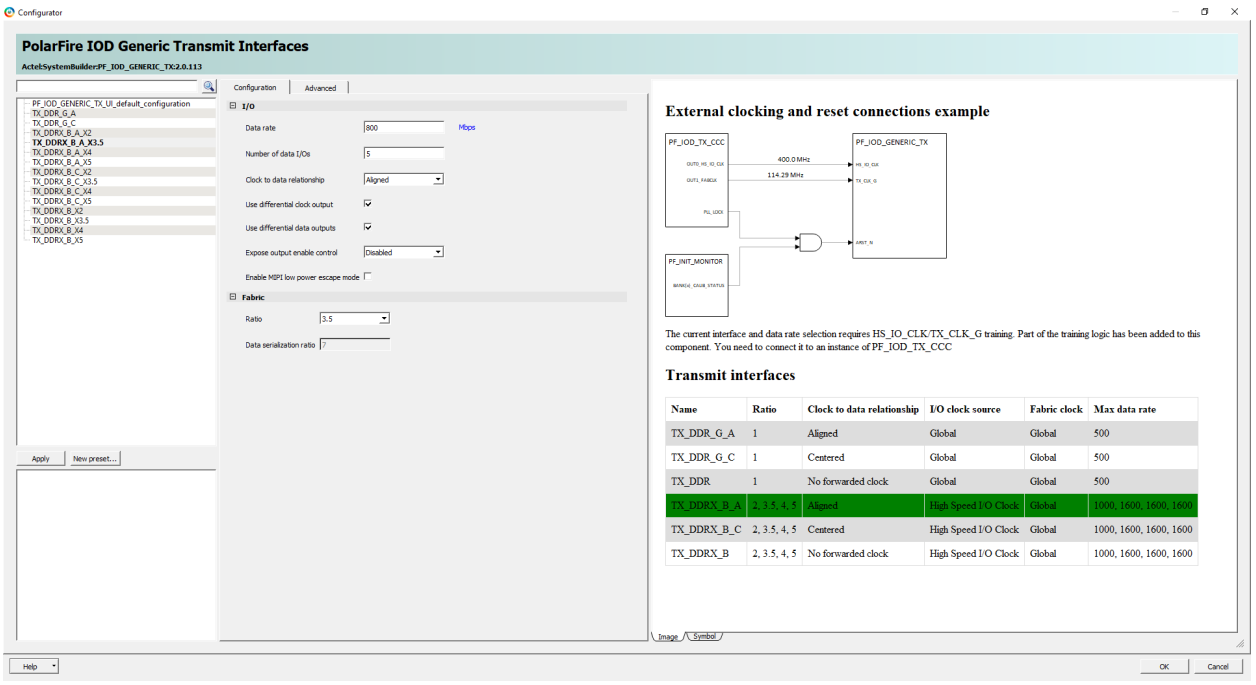
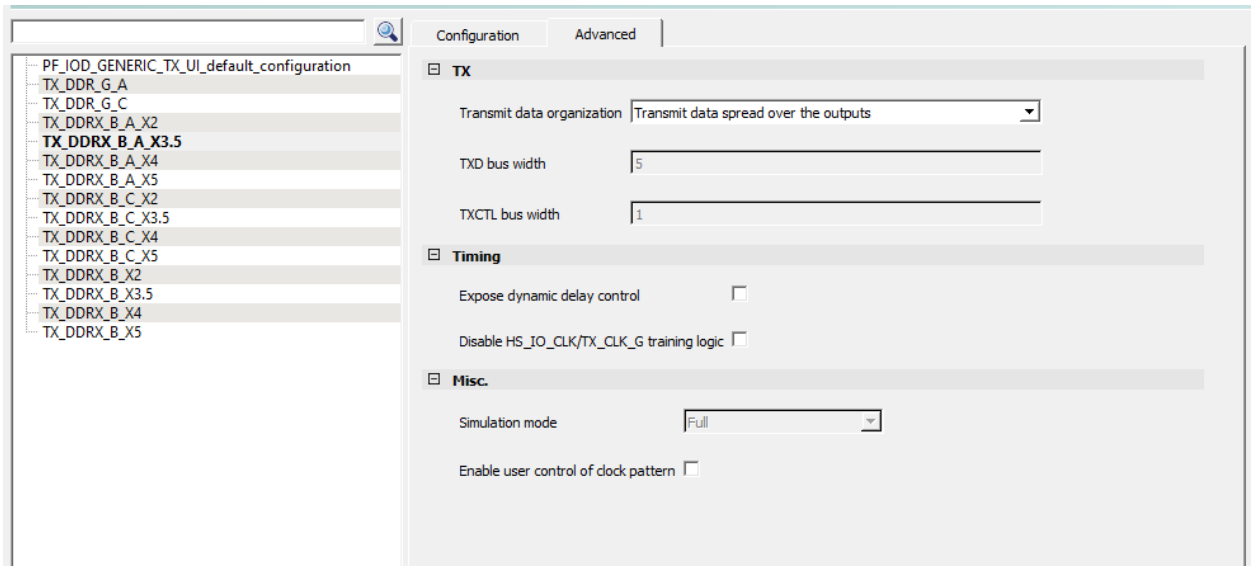


Figure 10-14. TX_DDRX_B_A_X3.5—Advanced Tab



10.4. SpaceWire Clock and Data Recovery (For RT PolarFire and RT PolarFire SoC FPGA Only) (Ask a Question)

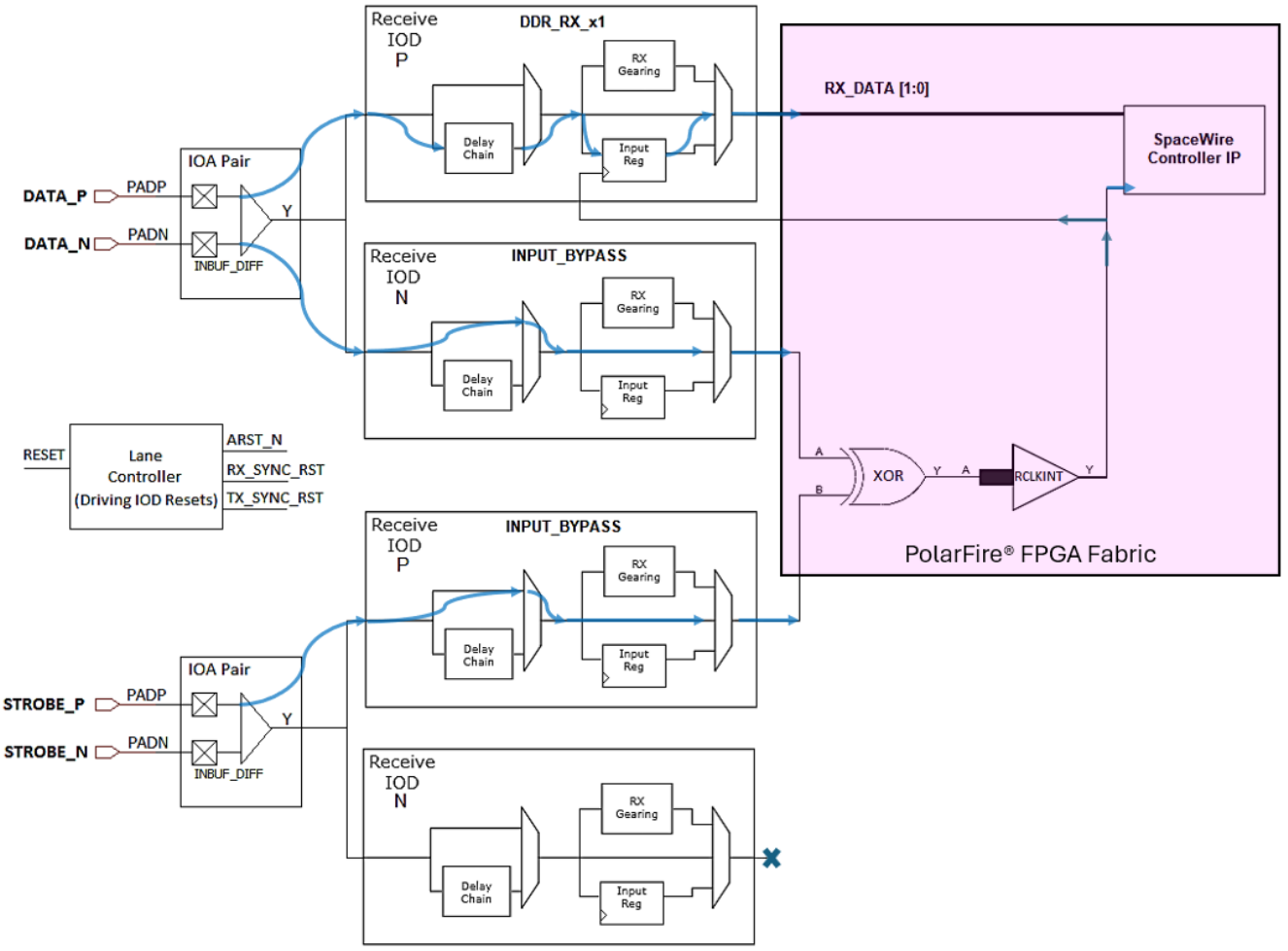
SpaceWire is a spacecraft onboard data-handling network that connects instruments to the mass-memory, data processors, and control processors, which is already in orbit or being designed into more than 100 spacecrafts. The RT PolarFire and RT PolarFire SoC FPGA does not have dedicated SpaceWire clock and data recovery circuits like those found on RTG4 FPGAs. The RT PolarFire and RT PolarFire SoC FPGA performs SpaceWire clock recovery using FPGA fabric logic while delaying the data path in parallel to compensate for the clock recovery delay.

To implement the SpaceWire RX PHY in the RT PolarFire and RT PolarFire SoC FPGA, PolarFire IOD interfaces are used in specific modes for data versus strobe input signals at up to 400 Mbps on -1 speed grade, and 300 Mbps on STD speed grade, with static delays. The Rx DATA is delayed using the IOD input delay chain. For RT PolarFire and RT PolarFire SoC SpaceWire RX interfaces, static IOD delays are sufficient instead of dynamic IOD delays. Designs must use an iterative approach in the Libero SoC design flow to find the correct static delay tap value to pass Static Timing Analysis (STA) and achieve hardware functionality across the full operating conditions range. Typical application results show that IOD static input delay settings in the range of 50 delay taps are sufficient for timing analysis of internal FPGA paths. For external timing checks, use an iterative or post-layout "`edit_io`" approach to fine-tune the delay settings.

Figure 10-15 shows the SpaceWire RX PHY implementation in RT PolarFire and RT PolarFire SoC FPGA. Each Input I/O has an IOA and IOD receiver component. Thus, a differential input pair has a P-side and N-side IOA and IOD receiver, also called IOA pair and IOD pair. In the following figure, the IOD pair used for the DATA input has a split configuration. DATA_IOD_P is configured for DDR_RX with a ratio of 1 and uses the IOD delay chain to compensate for the clock recovery delay. This is where the incoming DDR data is registered using the recovered RX_CLK. In contrast, the DATA_IOD_N is configured for Input Bypass mode so that the non-delayed DATA_N can be XOR'd with the STROBE_P input. The STROBE_P input passes through the STROBE_IOD_P block, which is also configured for Input Bypass mode. Finally, you can take the RX_DATA[1:0] and the recovered RX_CLK from the SpaceWire RX PHY for use with their third-party SpaceWire Controller/Router IP Core (not part of the Libero SoC DirectCore IP catalog).

Important: The STROBE_IOD_N output to the FPGA fabric is not used, even though, it is shown in the following figure for completeness, as part of the differential input pair. The SpaceWire RX PHY reset is connected to the I/O Lane Controller, and the Lane Controller outputs are used to reset the individual IODs.

Figure 10-15. SpaceWire RX PHY Implementation in RT PolarFire and RT PolarFire SoC FPGA

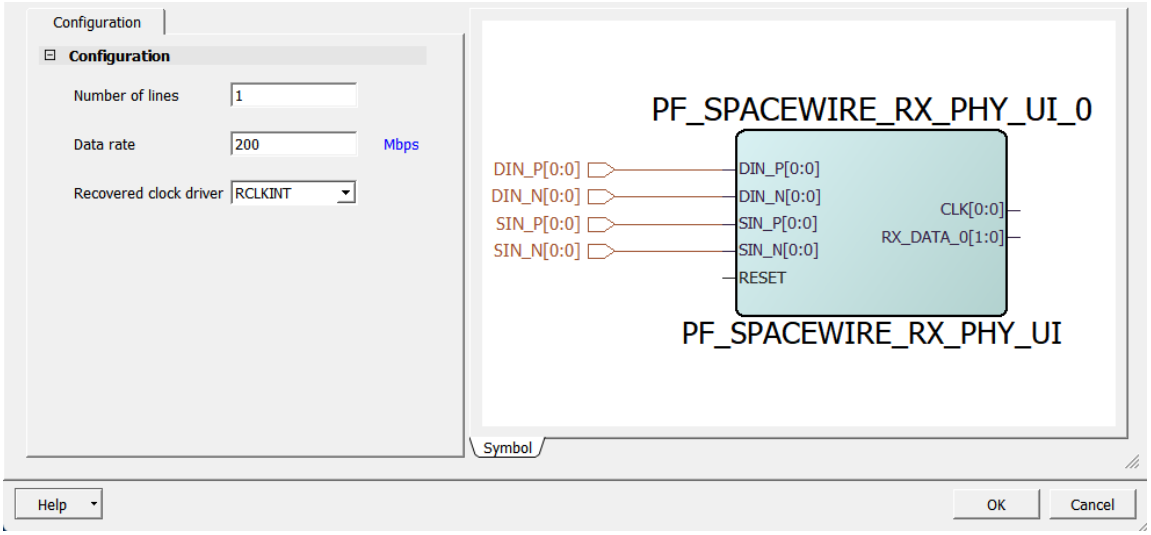


Additional Guidelines:

- When using differential DATA and STROBE, each SpaceWire RX PHY requires four I/O pins: two for DATA (P and N) and two for STROBE (P and N). From the board point of view, DATA is one LVDS Input Pair and STROBE is one LVDS Input Pair.
  - Each differential input pair uses an IOA pair, as in normal LVDS, through the use of an INBUF_DIFF macro.
  - After the INBUF_DIFF, the STROBE input is setup like a typical LVDS on PolarFire, where the INBUF_DIFF digital output Y only feeds the IOD_P, and the IOD_N is unused.
  - The DATA input builds on the normal LVDS setup. In this case, the INBUF_DIFF digital output (Y) feeds both the IOD_N and IOD_P. The IOD_N connects to the fabric XOR to recover the SpaceWire clock. The IOD_P is used to delay and register the SpaceWire DDR input data, clocked by the output of the XOR. Therefore, the DATA IO Digital (IOD) receiver pair is configured independently for DATA_P versus DATA_N IODs.
- The best location for SpaceWire Rx inputs is next to either of the two vertical stripes of Row Global Buffers (RGBs) feeding the logic sectors in the RTPF500 FPGA fabric. This minimizes clock insertion delay and also minimizes the length of the regular fabric routing used between the XOR and the RGB.

  - The further the SpaceWire RX input pins are placed from an RGB column, the lower the maximum achievable data rate.
  - The best bank that matches the preceding requirements is Bank 2 of RTPF500-CG1509.
- To minimize skew between the STROBE_IN and DATA_IN, it is highly recommended to assign these inputs to input pins in the same I/O Lane.
  - Use the Graphical I/O Editor in Libero SoC Constraint Manager to see the IOD view of the package pins and ensure that pins selected are part of the same I/O Lane.
- To reset the IOD in DDR_RX mode, that is, the DATA_P IOD in the preceding figure, the Lane Controller's reset output signals must be used. Therefore, all the I/Os used for a SpaceWire RX Interface, sharing a common RX PHY reset, must be placed in the same lane so that they can use the same Lane Controller (per RX PHY reset).
  - You can then drive the Reset input of the Lane Controller, which propagates the reset to the IODs by making the appropriate connections to the ARST_N and RX_SYNC_RST ports.
  - Since there is only one reset per Lane Controller, the user must decide whether multiple SpaceWire RX interfaces can share a reset or whether they must be placed in independent I/O lanes.
- In Libero SoC v2022.2 and later, the IP Catalog includes PF_SPACEWIRE_RX_PHY core to implement the SpaceWire RX PHY using the underlying IOD and Lane Controller building blocks.

Figure 10-16. RT PolarFire and RT PolarFire SoC SpaceWire RX PHY



  - DO NOT try to manually modify the instances of the PF_IOD and PF_LANECTRL in the generated IP core RTL. These macros are instantiated by the PF_SPACEWIRE_RX_PHY IP core based on the user core configuration selections.
  - As shown in the preceding figure, the RESET input to the PF_SPACEWIRE_RX_PHY core is Active-High, as indicated by the port name which excludes a trailing "N".
- In the preceding figure, RX_DATA[0] is the received data bit corresponding to the rising edge of the recovered clock, and RX_DATA[1] is the received data bit corresponding to the falling edge of the recovered clock.
- At higher data rates, such as 400 Mbps, if you encounter difficulty closing timing, the fabric logic clocked by the recovered clock, passing through the RCLKINT must be placement constrained (through floor planning PDC and inclusive region constraint) to reside in the same fabric logic

sector to ensure that a single RGB is sufficient. A fabric logic sector is six logic clusters wide by nine logic clusters tall, as described in the PolarFire Family Clocking Resources User Guide.

- The XOR, which is used to create the receive clock, must be placed in the logic cluster above the strobe (and not the data). This ensures a smaller delay to the XOR input from the strobe compared to the delay from the data to the XOR. The strobe delay line can then be used to minimize the skew at the XOR.
- The fanout of the recovered clock must be added in a region that spans the first sector row of 6x9 clusters just above the I/O. The goal is to minimize the internal skew of the recovered clock inside the FPGA (and so avoid significant minimum delay issues) by limiting the recovered clock driver to one RGB.
- The data captured by the SpaceWire RX PHY and the respective recovered clock can eventually be transferred to the system clock domain through techniques such as an asynchronous FIFO (independent read and write clocks).
- Alternatively, if the recovered SpaceWire clock has a load larger than a sector, then the impact of the RCLKINT and RGB replication performed by the Place and Route process must be analyzed during timing analysis.

10.4.1. Example Timing Constraints for a x2 SpaceWire RX Interface (Ask a Question)

Consider an example design where the PF_SPACEWIRE_RX_PHY core is used to generate two SpaceWire RX inputs, with the programmable input delay set to 50 static delay taps on the respective DATA_IN_N ports. Furthermore, assume that you renamed the top-level inputs from the default DIN_P/N[1:0] and SIN_P/N[1:0] to more descriptive names, according to their system/board naming convention.

With these port names, the following SDC constraints can be used for Synthesis, Place and Route, and Timing Verification:

```
#SpaceWire Input Timing Constraints
########################################

#Create clock on both strobe and data inputs since both create clock edges
create_clock -name {SpW_0_RX_CLK} -period 7 \
        [get_ports {SpW_0_STROBE_IN_P SpW_0_DATA_IN_P}]
create_clock -name {SpW_1_RX_CLK} -period 7 \
        [get_ports {SpW_1_STROBE_IN_P SpW_1_DATA_IN_P}]

#Specify clock jitter on the SpW RX clock recovered from RX DATA and STROBE inputs
set_input_jitter 0.200 [ get_clocks { SpW_0_RX_CLK } ]
set_input_jitter 0.200 [ get_clocks { SpW_1_RX_CLK } ]

#External Timing Constraints:
########################################
# Max Delay of    0 ns    : for latching data from same edge as launched
#            -X ns    : X is skew between S and D from external SpW TX,
#                     + board skew, etc, assume 100ps below
#            to    : FF capturing RX data, e.g. Data IOD RX_P:
#                PF_SPACEWIRE_RX_PHY_C0_0/PF_IOD_DATA_P_0/I_IOD_0/RX_P

set_max_delay -0.100 \
                -to [get_pins {PF_SPACEWIRE_RX_PHY_C0_0/PF_IOD_DATA_P_0/I_IOD_0/RX_P}]
set_max_delay -0.100 \
                -to [get_pins {PF_SPACEWIRE_RX_PHY_C0_0/PF_IOD_DATA_P_1/I_IOD_0/RX_P}]

#Min Delay -3.5 ns    : hold on previous edge using clock_period / 2
#         + X ns    : X is skew between S and D from external SpW TX,
#                   + board skew, etc, assume 100ps below
#            to    : FF capturing RX data, e.g. Data IOD RX_P:
#                PF_SPACEWIRE_RX_PHY_C0_0/PF_IOD_DATA_P_0/I_IOD_0/RX_P

set_min_delay -3.4 \
                -to [get_pins {PF_SPACEWIRE_RX_PHY_C0_0/PF_IOD_DATA_P_0/I_IOD_0/RX_P}]
set_min_delay -3.4 \
                -to [get_pins {PF_SPACEWIRE_RX_PHY_C0_0/PF_IOD_DATA_P_1/I_IOD_0/RX_P}]

#Normally for source synchronous inputs, input delays use the formulas:
```

```
#input max dly= max data trace dly + Tco of ext TX - min clk trace dly
#input min dly= min data trace dly + min Tco of ext TX - max clk trace dly
#
#However, in SpW RX, it's S vs. D skew affecting the "data valid window"
#Thus, commented lines below since skew considered in max/min delays above
#set_input_delay 0 -clock {SpW_0_RX_CLK} [get_ports {SpW_0_DATA_IN_N SpW_0_DATA_IN_P}]
#set_input_delay 0 -clock {SpW_1_RX_CLK} [get_ports {SpW_1_DATA_IN_N SpW_1_DATA_IN_P}]

#Constraint below tells place and route to minimize DATA delay through XOR
set_max_delay 0 -from [ get_clocks { SpW_0_RX_CLK } ] \
        -through [get_cells {PF_SPACEWIRE_RX_PHY_C0_0/XOR2_0}] \
        -to [get_pins {PF_SPACEWIRE_RX_PHY_C0_0/PF_IOD_DATA_P_0/I_IOD_0/RX_P}]

set_max_delay 0 -from [ get_clocks { SpW_1_RX_CLK } ] \
        -through [get_cells {PF_SPACEWIRE_RX_PHY_C0_0/XOR2_1}] \
        -to [get_pins {PF_SPACEWIRE_RX_PHY_C0_0/PF_IOD_DATA_P_1/I_IOD_0/RX_P}]
```

Important:

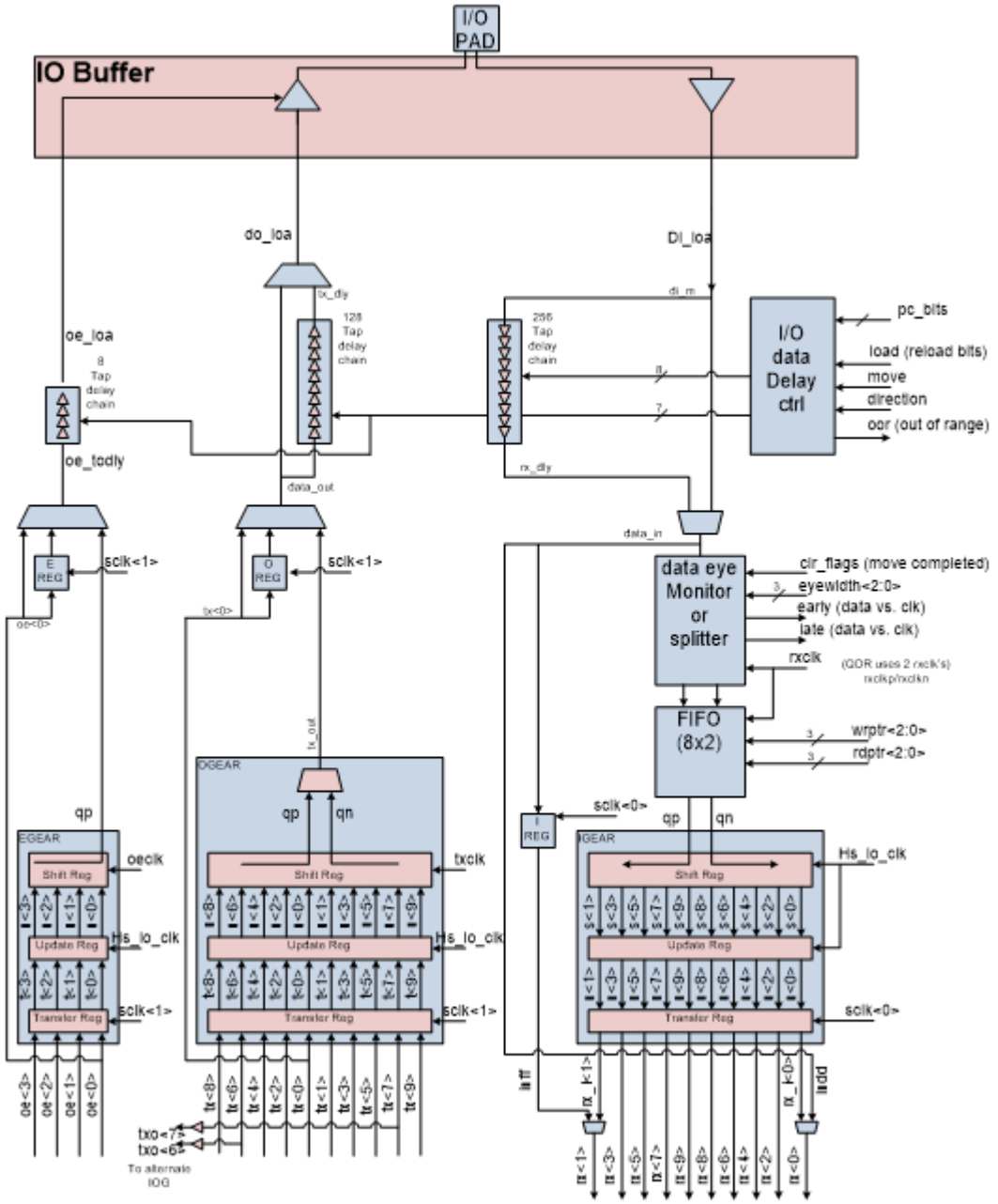
- The path through the XOR must be minimized during Place and Route.
- Synthesis does not recognize some of the paths in the SDC, but Place and Route and Verify Timing do recognize the paths.
- You can have separate SDCs for Synthesis, Place and Route, and Verify Timing. Using independent SDC files for each design step can minimize the run-time, if evaluating the impact of constraint changes to Timing Verification independently from Synthesis and Place and Route.
- For the initial Place and Route, do not run min-delay repair. Once max delay timing is met, review the hold time violations, if any, on the DATA_IN path to the I/O register to understand how much additional programmable input delay would be needed to meet timing at the DATA IOD_P DDR input register. Then run an incremental Place and Route with min-delay repair enabled. Review the resulting min-delay-repair report and the resulting timing reports to confirm whether the automatically added I/O delay meets your required slack margin.
- In the Libero SoC I/O Editor, the programmable Input Delay setting will appear on both the DATA_P and DATA_N rows. However, the delay will only impact the timing of the SpaceWire RX IOD_P side of the DATA input as that IOD is configured in DDR_RX_x1 mode, using the DDR input I/O Flip-Flop (IOFF).
- As mentioned previously, to achieve higher data rates, such as 400 Mbps, use the following process to account for recovered clock skew. It can be obtained and used to fine tune the strobe IOD delay line tap setting.
  - In the timing report explorer, enable the skew column. Look for significant skew for reg-to-reg paths in the recovered clock domain. The delay from the data to the XOR input must be higher than the delay from the strobe to the XOR input.
  - Update the delay line settings of the strobe IOD to "skew in ps / 30".
  - Run Place and Route in incremental mode and with min-delay repair enabled.
  - Finally, if the external timing requirements are still not met, adjust the data IOD delay line setting. Increase the delay tap to fix a hold issue and decrease the delay tap to fix a setup issue. The min-delay repair in the incremental step above tries to fix external hold time violations automatically by adjusting the I/O delay taps. This step can also employ the Libero SoC design flow step called **Edit Post Layout Design**, using a separate PDC file (that is, not managed by Libero Constraints Manager) containing `edit_io` commands. This allows faster evaluation of I/O delay tuning to close timing, without requiring a new Place and Route until the final delay tap settings are reached. Refer to the Libero SoC Design Flow User Guide for more information about the Post Layout Editing of I/O Delay Parameters.
- In the example constraints above, the input_delay of 0 ns assumes an ideal situation where incoming DATA path is matched to incoming STROBE path from the transmitting device and on the board. You must update the SDC to account for the actual input_delay from the external TX and board routing, or account for this external skew in the `set_max_delay` and `set_min_delay` constraints shown earlier.

11. Dynamic IOD Interface Training (Ask a Question)

11.1. Clock to Data Margin Training (Ask a Question)

Margin control training of the IOD interface maximizes the valid window by continuously monitoring and controlling the delays using the dynamic delay control signals. This operation is used to compensate for the PVT variations with high-speed source synchronous interfaces. The main reason for this capability is to optimize the signal integrity of the high-speed IOD interfaces by maintaining margin between the data and clock paths. Interface training is controlled and monitored by FPGA hosted IP (that is, Training IP or TIP).

Figure 11-1. Clock to Data Training Data Path



The TIP uses the dynamic delay control pins of the dynamic RX_DDRX interface components to optimize the receive relationship between the clock and data. Status flags are used to dynamically monitor the relationship of the clock and data at the IREG and uses dynamic controls to adjust the delay chain by adding or removing delay elements in the data path. The delay setting is adjusted to move the data edges earlier or later relative to the clock edges. This feature monitors the relation of the data edges to both the positive and negative clock edges.

FPGA fabric hosted logic is used to control and monitor IOD signals to perform adaptive tuning functions on a bit- or word-wide basis. Bit alignment is the alignment of the data to be 90 degrees centered from the clock edges. This is a physical layer function that is independent of the data or protocol being used. This step requires the transmitter to send data (with transitions) and has a static alignment with the forwarded clock.

RX_DDRX_DYN macro provides controls to add or remove delay from the data path relative to the clock path. The RX_DDRX_DYN also provides flags using the eye monitor which can identify when the data and clock are too close together and side of the clock in which the violation occurs. Using these controls and flags, bit alignment can be performed by only looking at the physical layer.

Word Alignment is the alignment of the fabric presented word to a specific pattern. The RX_DDRX_DYN provides I/O gearing and supports both a 4-bit and 8-bit fabric width. Byte alignment is data pattern dependent and would require a training pattern. When the transmitter sends the training pattern, a pattern detector in the FPGA fabric would use the Lx_BITSLIP port on the RX_DDRX_DYN to rotate the fabric word till the training pattern is found.

The signal, “DELAY_LINE_LOAD” asynchronously reloads the initial static Flash bit delay settings that are predefined by Libero SoC. The signal, “DELAY_LINE_MOVE” uses a rising edge to change the delay setting by ±1 increment each time it is pulsed according to the “DELAY_LINE_DIRECTION” signal value (a “1” increases up the delay setting by 1 increment and a “0” decreases down the delay setting by 1 increment). When the delay setting reaches the minimum value or the maximum value of the delay chain, the delay chain controller generates an out of range output Flag “DELAY_LINE_OUT_OF_RANGE” to indicate that it has reached the end of the delay chain. The delay setting stops at this minimum or maximum setting, even if the “DELAY_LINE_MOVE” signal is still pulsing. Toggling the HS_IO_PAUSE does not disrupt delay line value settings.

The IOD block has a Data Eye Monitor (DEM) used to optimize the clock and input data relationship. The data eye monitor (DEM) and its associated flags are not available to IOD interfaces using a gearing ratio of 1. The DEM includes EYE_MONITOR_EARLY and EYE_MONITOR_LATE flags used to analyze the clock-to-data relationship. IOD designs can utilize these flags to determine the input data edge relationship to the clock edge. The design can then use the DELAY_LINE control inputs to dynamically adjust this relationship to optimize the clock and data relationships until an optimal setting is found.

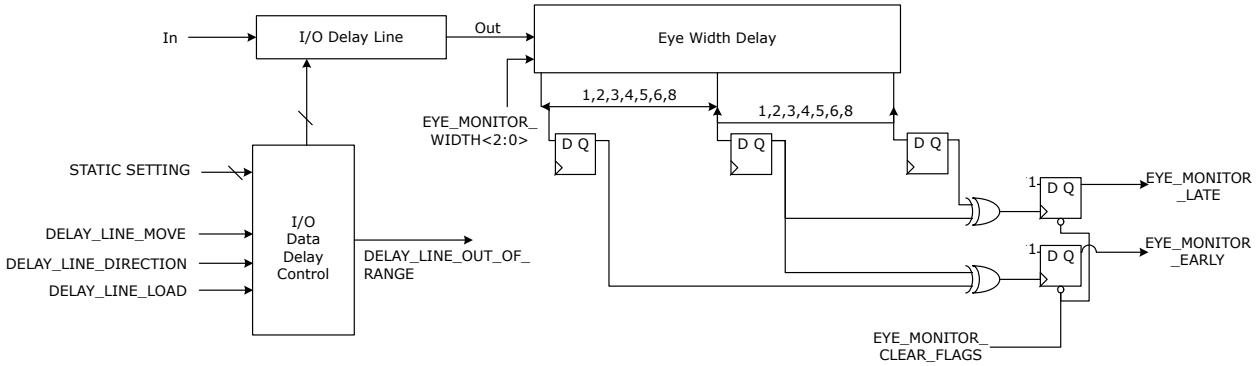
Similarly, output delays are affected when the IOD tri-state enable “E” = 1 or the input delay “E” = 0. The DELAY_LINE_MOVE is applied to the output delay path.

The Data Eye Monitoring (DEM) is accomplished as follows:

- Use the input signals “EYE_MONITOR_WIDTH<2:0>” to set a minimum delay space requirement between the data edges and the clock edges. The programmable delay settings are programmed in delay increments of 1 to 128 taps. This delay setting is then used to generate EYE_MONITOR_EARLY and EYE_MONITOR_LATE flag if the data edges are closer to the clock edges than this minimum setting. By allowing these signals to be dynamically controlled from the FPGA hosted logic, the user can determine the relative size of the eye opening.
- EYE_MONITOR_EARLY is asserted if the data edge is too close to the clock edge on the early side of clock. This Flag indicates that the delay setting must be moved down (decremented).
- EYE_MONITOR_LATE is asserted if the data edge is too close to the clock edge on the late side of clock. This Flag indicates that the delay setting must be moved up (incremented).

- Use the “EYE_MONITOR_CLEAR_FLAGS” input signal, from the fabric, to clear the “EYE_MONITOR_EARLY” and “EYE_MONITOR_LATE” Flags. This signal is from the fabric and indicates that the delay chain setting has been incremented or decremented as a function of the previous Flag settings.

Figure 11-2. IOD Training Block Diagram



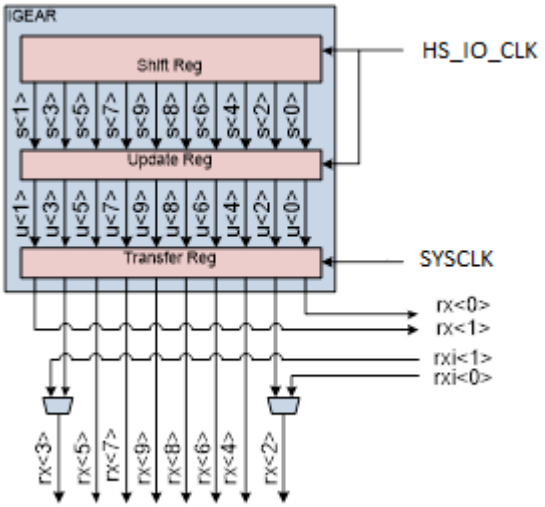
11.1.1. HS_IO_CLK and System Clock Training (Ask a Question)

IOD interfaces implement Input gearing, de-serialization of the high-speed pad signals to lower speed parallel core signals, and the clock domain transfers, as required for the specific interface. The IOD implements a clock domain transfer for the data from the high-speed (HS_IO_CLK) to the low-speed system clock (SYSCLK) which is either GLOBAL or REGIONAL clock of the IOD macro. IOD Rx data is transferred from the Update Register (HS_IO_CLK domain) to the Transfer Register (SYSCLK) domain in the IGEAR logic. HS_IO_CLK and system clock training is implemented with interfaces where the data rate is greater than or equal to 400 Mbps, ratio 2, 3.5, 4. Interfaces using ratio x5 does not require HS_IO_CLK to system clock training because of its higher ratio which already provides adequate margin between them.

The Input IREG gearing logic data path uses three sets of registers to move the data between the domains. The following registers are depicted in Figure 11-3.

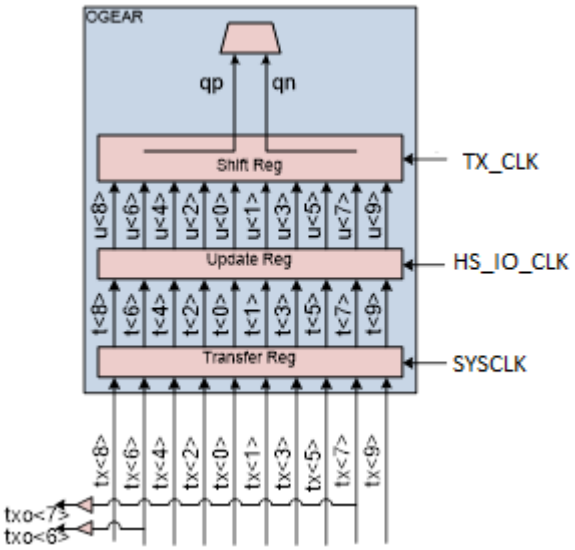
- Shift register
- Update register
- Transfer register

Figure 11-3. HS_IO_CLK to SYSCLK Data Transfer



Similarly, IOD Tx data is transferred from the Transfer Register (SYSCLK domain) to the Update Register (HS_IO_CLK domain) in the OGEAR logic using a same domain transfer topology.

Figure 11-4. SYSCLK to HS_IO_CLK Data Transfer



The HS_IO_CLK and SYSCLKs can have different insertion delays due to dissimilar routing paths within the fabric. This causes the rising clock edges to be misaligned potentially causing timing mismatches when the rising edges of these clocks are not aligned.

Figure 11-5. SYSCLK to HS_IO_CLK Before Training

Figure 11-6. SYSCLK to HS_IO_CLK After Training

In the Figure 11-5 and Figure 11-6, a PLL VCO phase adjustment for the HS_IO_CLK is required to align the rising edges of System clock and HS_IO_CLK for best performance. It requires use of the data EYE_MONITOR of an unused/spare IOD lane to derive the best setting. Upon completion, the CLK_TRAIN_DONE output indicates that the training is successful. CLK_TRAIN_ERROR indicates an error causing the HS_IO_CLK and system clock not to train. This can occur when the clocks are interrupted. CLK_TRAIN_ERROR is not available with fractional interfaces.

Clock training occurs automatically at power-up or the assertion of DEVRST_N for both Tx and Rx IODs in designs that include clock training IP. Tx IOD clock training aligns the fabric global clock and HS_IO_CLK outputs of the PF_IOD_TX_CCC to compensate for CCC/IOD routing variations within the fabric. Retraining can be manually invoked by asserting the CLK_TRAIN_RESTART input of the PF_IOD_TX_CCC.

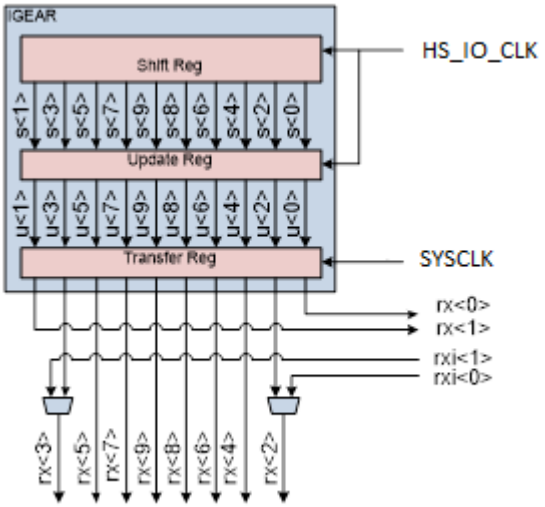
11.2. HS_IO_CLK and System Clock Training (Ask a Question)

IOD interfaces implement Input gearing, de-serialization of the high-speed pad signals to lower speed parallel core signals, and the clock domain transfers, as required for the specific interface. The IOD implements a clock domain transfer for the data from the high-speed (HS_IO_CLK) to the low-speed system clock (SYSCLK), which is either GLOBAL or REGIONAL clock of the IOD macro. IOD Rx data is transferred from the Update Register (HS_IO_CLK domain) to the Transfer Register (SYSCLK) domain in the IGEAR logic. HS_IO_CLK and system clock training are implemented with interfaces where the data rate is greater than or equal to 400 Mbps, ratio 2, 3.5, 4. Interfaces using ratio x5 does not require HS_IO_CLK to system clock training because of its higher ratio, which already provides adequate margin between them.

The Input IREG gearing logic data path uses three sets of registers to move the data between the domains. The following registers are depicted in the following figure.

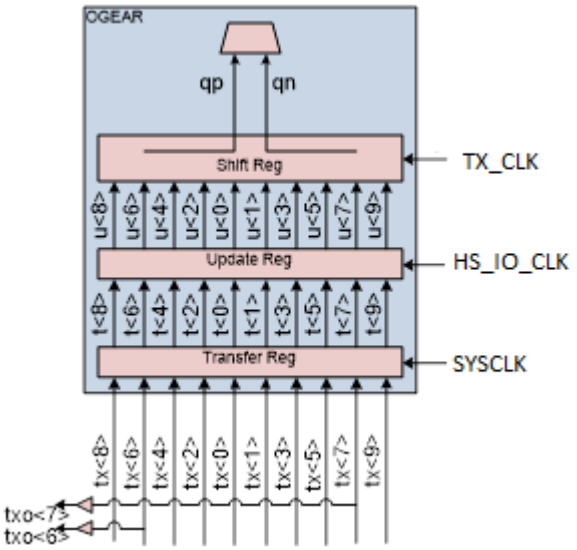
- Shift register
- Update register
- Transfer register

Figure 11-7. HS_IO_CLK to SYSCLK Data Transfer



Similarly, IOD Tx data is transferred from the Transfer Register (SYSCLK domain) to the Update Register (HS_IO_CLK domain) in the OGEAR logic using a same domain transfer topology.

Figure 11-8. SYSCLK to HS_IO_CLK Data Transfer



HS_IO_CLK and SYSCLKs can have different insertion delays due to dissimilar routing paths within the fabric. This causes the rising clock edges to be misaligned, potentially causing timing mismatches when the rising edges of these clocks are not aligned.

Figure 11-9. SYSCLK to HS_IO_CLK Before Training

Figure 11-10. SYSCLK to HS_IO_CLK After Training

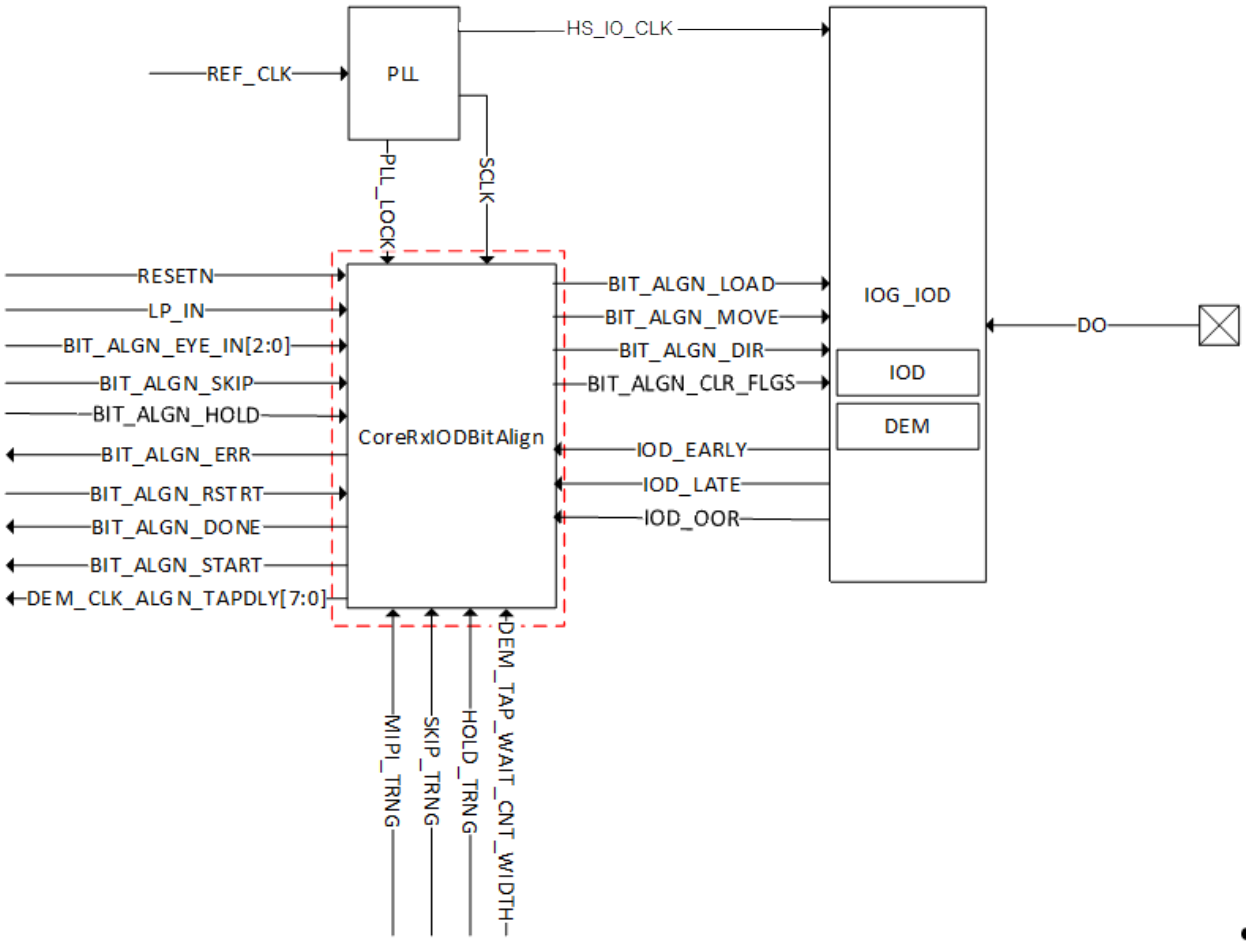
In the Figure 11-9 and Figure 11-10, a PLL VCO phase adjustment for HS_IO_CLK is required to align the rising edges of System clock and HS_IO_CLK for best performance. It requires use of the data EYE_MONITOR of an unused/spare IOD lane to derive the best setting. Upon completion, the CLK_TRAIN_DONE output indicates that the training is successful. CLK_TRAIN_ERROR indicates an error causing the HS_IO_CLK and system clock not to train. This can occur when the clocks are interrupted. CLK_TRAIN_ERROR is not available with fractional interfaces.

Clock training occurs automatically at power-up or the assertion of DEVRST_N for both Tx and Rx IODs in designs that include clock training IP. Tx IOD clock training aligns the fabric global clock and HS_IO_CLK outputs of the PF_IOD_TX_CCC to compensate for CCC/IOD routing variations within the fabric. Retraining can be manually invoked by asserting the CLK_TRAIN_RESTART input of the PF_IOD_TX_CCC.

11.3. CoreRxIODBitAlign (Ask a Question)

CoreRxIODBitAlign IP available from the Libero SoC Catalog performs training when interfacing the IOD macro to support as a dynamic source with adjusting delays to capture the data correctly.

Figure 11-11. CoreRxIODBitAlign Implementation Diagram



Important: HS_IO_CLK is internal connection.

This CoreRxIODBitAlign IP works based on Fabric clock (OUT_FABCLK*) from CCC or PLL component and PF_IOD_GENERIC_RX IOD component works based on OUT*_HS_IO_CLK_* or for bit alignment.

An example application for bit alignment uses the PF_IOD_GENERIC_RX IOD component to receive the serial data with a required data rate of 1000 Mbps in DDRx4 fabric mode. The OUT2_FABCLK_0 know as SCLK is driven from the PLL of the CCC component at 125 MHz and OUT0_HS_IO_CLK_0 to PF_IOD_GENERIC_RX at 500 MHz.

The CoreRxIODBitAlign IP starts the training when the PLL_LOCK is stable and driven high. The LP_IN input is used only in the CoreRxIODBitAlign IP when MIPI_TRNG parameter is set to 1. This LP_IN signaling is active low and level-based, detected as negative edge every time by the IP to indicate the valid start of frame to start the bit alignment training mechanism. If MIPI_TRNG parameter is set to 0, then this input is left unused by the IP.

The CoreRxIODBitAlign IP indicates the start of training by driving BIT_ALGN_START high and BIT_ALGN_DONE as low. It then drives the output BIT_ALGN_LOAD to load the default settings in the PF_IOD_GENERIC_RX component. The BIT_ALGN_CLR_FLGS is used to clear the IOD_EARLY, IOD_LATE and BIT_ALGN_OOR flags.

The CoreRxIODBitAlign IP proceeds with BIT_ALGN_MOVE followed with BIT_ALGN_CLR_FLGS for every TAP and records the IOD_EARLY, IOD_LATE flags. When BIT_ALGN_OOR is set high by the PF_IOD_GENERIC_RX component, then the CoreRxIODBitAlign IP sweeps the recorded EARLY and LATE flags and finds the optimal EARLY and LATE flags to calculate the required TAP delays for clock and data bit alignment. CoreRxIODBitalign IP selects the window size, which has the best case IOD_EARLY and IOD_LATE flags counts.

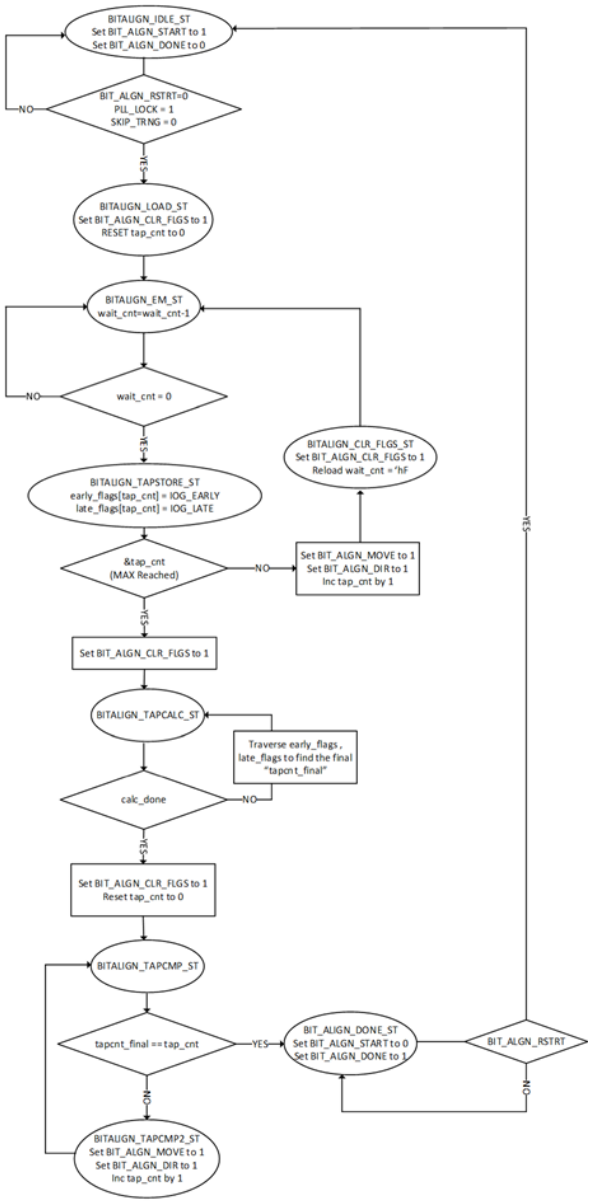
The CoreRxIODBitAlign IP loads the calculated TAP delays and drives BIT_ALGN_START low and BIT_ALGN_DONE high to indicate the completion of the training.

The CoreRxIODBitAlign IP continues the Re-training dynamically if it detects noisy IOD_EARLY or IOD_LATE feedback assertion from PF_IOD_GENERIC_RX component. The BIT_ALGN_DONE is reset and driven low and BIT_ALGN_START is driven high again by the CoreRxIODBitAlign IP to indicate the restart of the training. The timeout counter when reaches the timeout condition asserts the BIT_ALGN_ERR at the end of the training.

The CoreRxIODBitAlign IP also provides restart mechanism for the user to restart the training whenever required. The BIT_ALGN_RSTRT input is active high level must be driven high (for example, 8 clocks). The BIT_ALGN_DONE is reset and driven low. BIT_ALGN_START is driven high again by the CoreRxIODBitAlign IP to indicate the fresh start of the training.

The CoreRxIODBitAlign IP also provides hold mechanism to hold the training in the middle. In this use case, the HOLD_TRNG parameter must be set to 1 then the CoreRxIODBitAlign IP uses the BIT_ALGN_HOLD input and asserts active high level-based until it requires the CoreRxIODBitAlign IP to hold the training and then continues the training when the input BIT_ALGN_HOLD is driven low.

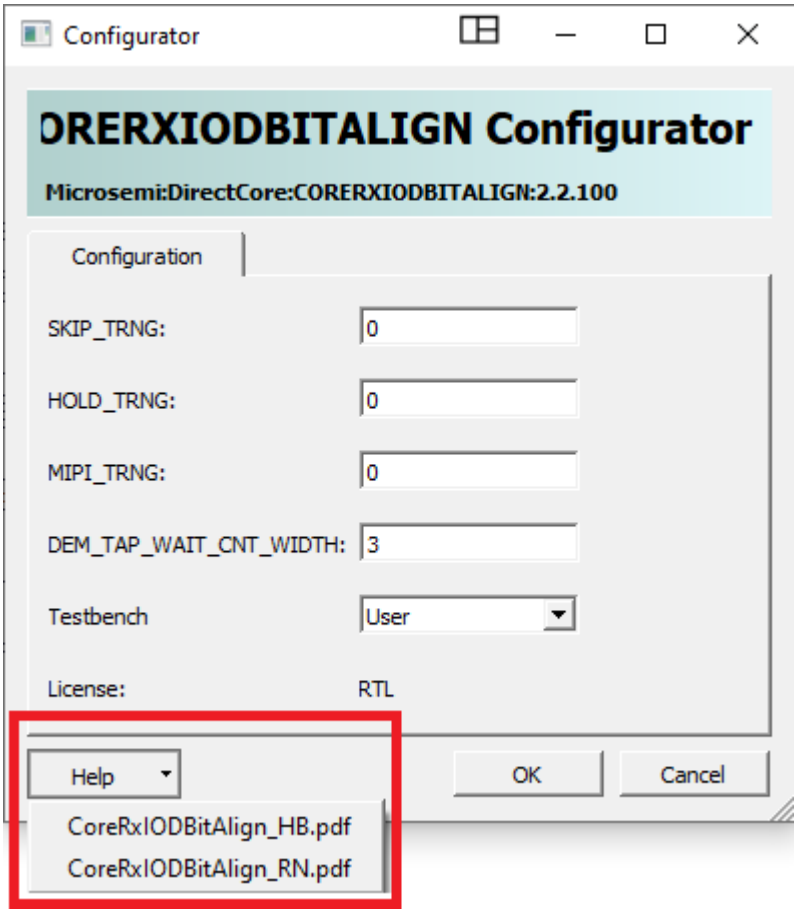
Figure 11-12. CoreRxIODBitAlign Training State Diagram



For more information about IP module usage, see HB0861: CoreRxIODBitAlign Handbook. This handbook can be downloaded from the Libero SoC Catalog.

The following figure shows the CoreRxIODBitAlign Libero SoC Configurator.

Figure 11-13. CoreRxIODBitAlign Libero SoC Configurator



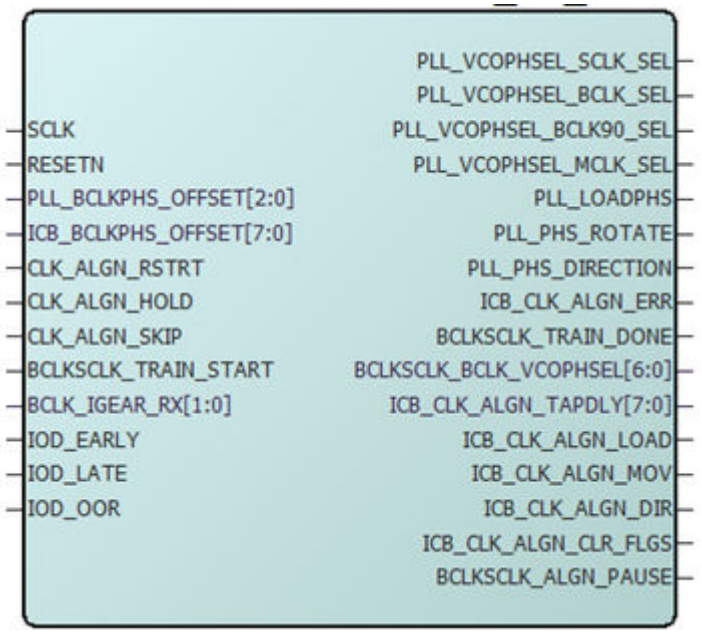
IOD analysis is available for generic RX IOD interfaces that use the CoreRXIODBITALIGN IP. SmartDebug includes an utility that can analyze the clock and data margins of an interface. The utility provides feedback on the amount of delay taps between lanes to depict any disparity in PCB routing deviations. See SmartDebug User Guide for more information.

11.4. CoreBclkSclkAlign Training IP (Ask a Question)

The CoreBclkSclkAlign training IP is provided with IOD blocks to optimize the performance of the interfaces. This must be instantiated for all Generic IOD Rx and Tx interfaces above 400 Mbps with x2, x3.5, and x4 ratios. The IP is specifically intended for clock alignment by using the ICB or PLL VCOPHSEL controls to move the BCLK (HS_IO_CLK) in the PLL in reference to SCLK (system or fabric clock).

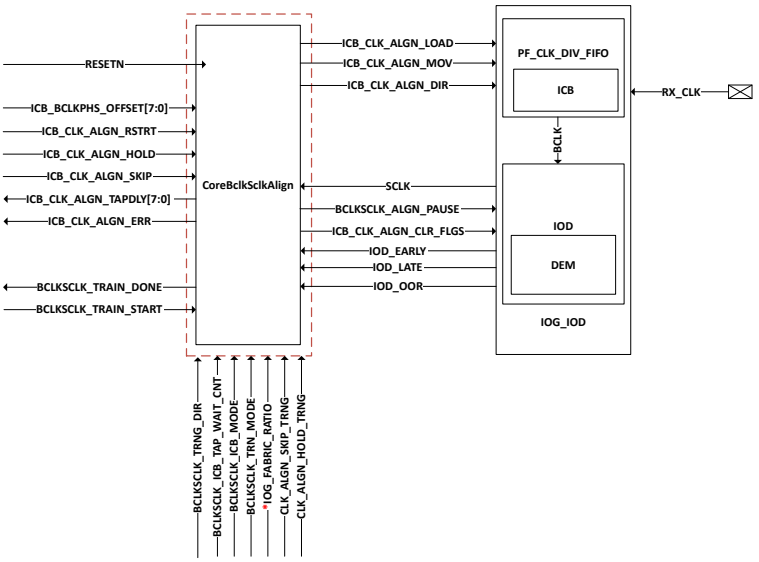
When creating generic IOD interfaces, the CoreBclkSclkAlign IP component is included in the generated core of the IOD Generic Transmit Interface Clocking IP (PF_IOD_TX_CCC) as well as the IOD Generic Receive Interface IP (PF_IOD_GENERIC_RX). Both of these fabric-hosted soft IP components include the CoreBclkSclkAlign, which is instantiated by Libero SoC during IOD configuration process for the purpose of calibrating and training the I/O gearing elements for the use of high-speed interfaces.

Figure 11-14. CoreBclkSclkAlign Component



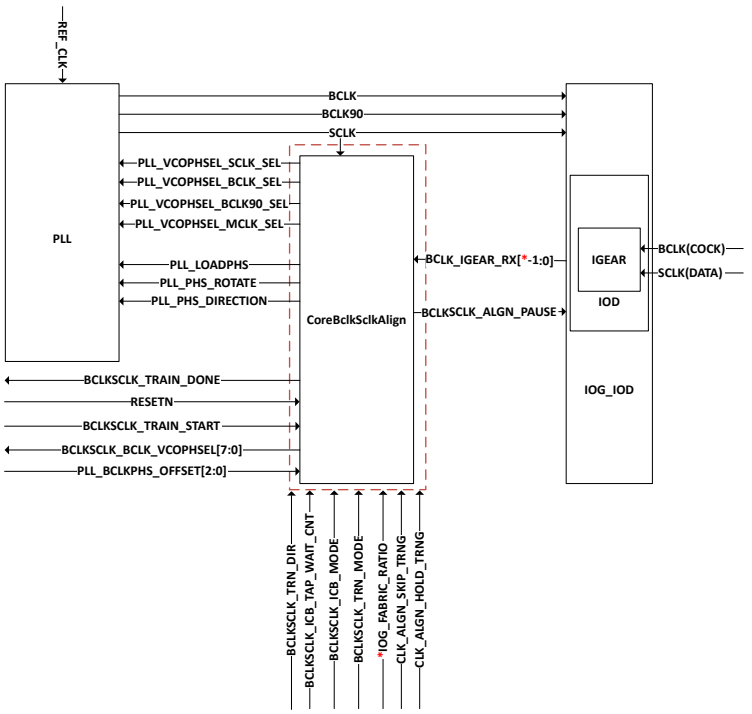
The following figure shows a high-level block diagram of CoreBclkSclkAlign used for clock training in RX modes.

Figure 11-15. RX Mode Clock Training Block Diagram



The following figure shows a high-level block diagram of CoreBclkSclkAlign used for clock training in TX modes.

Figure 11-16. TX Mode Clock Training Block Diagram



CoreBclkSclkAlign performs clock training and is also responsible for interfacing IOD to capture the data.

11.4.1. ICB-Based Fine Training Method (Ask a Question)

The programmable tap delay line is used to modify the delay values by using the fabric interface signals provided in the ICB interface to perform the fine clock training. This training uses the MOV, LOAD, and DIR of the ICB block to move the BCLK in reference to SCLK. This solution uses one spare IOD RX gearbox that uses SCLK as the data pin and BCLK90 as the clock pin. This training is used to control the timing relationships between SCLK and BCLK90 for both RX and TX domains. This is required to reliably and deterministically transfer data across the IO Gearing to the fabric interface. The objective is to set BCLK90 to follow SCLK by 90 degrees to define a known clock domain relationship. The retraining is done based on early late assertion in the IOD. HS_IO_CLK_PAUSE to the IOD is asserted at the end of the clock training routine.

11.4.2. PLL-Based Coarse Training Method (Ask a Question)

The coarse training uses the PLL VCOPHSEL controls to move the BCLK in the PLL in reference to SCLK. This trains both BCLK and BCLK90 by tying the phase select signals for both BCLK and BCLK90 together. For the training feedback, it uses one spare IOD RX gearbox that uses SCLK as the data pin and BCLK90 as the clock pin. This training is used to control the timing relationships between SCLK and BCLK90 for both RX and TX domains. This is required to reliably and deterministically transfer data across the I/O Gearing to the fabric interface. The objective is to set BCLK90 to follow SCLK by 90 degrees to define a known clock domain relationship. HS_IO_CLK_PAUSE to the IOD is asserted at the end of the clock training routine.

11.4.3. Example Training Algorithm (Ask a Question)

To train the algorithm, perform the following steps:

1. A spare IOD is used for this clock-to-clock training using the spare IOG data-eye feature. In this example, the SCLK is the data, and the BCLK90 is the clock. The PLL reference clock is using 100 MHz and the PLL generates the clocks for MAX data rate of 1600 Mbps.

Table 11-1. Example of PLL Output Clock

PLL Clock Output	Clock Description	Clock Frequency
PLL_OUT[0]	MEMCLK	800 MHz
PLL_OUT[1]	SCLK	200 MHz
PLL_OUT[2]	BCLK	800 MHz
PLL_OUT[3]	BCLK90	800 MHz[1]

Note:

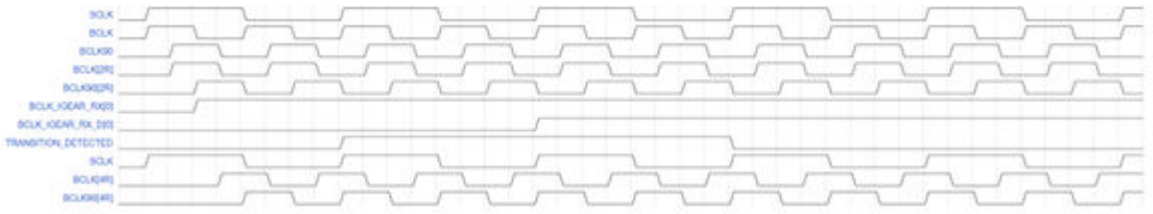
a. BCLK90 is always set at the start after the PLL Lock by moving the PLL VCOPHSEL in 45° through PLL ROTATE increment by 2, so as to follow BCLK by 90° shift.

2. Align BCLK90 to SCLK. The control mechanism is by moving the PLL VCOPHSEL in 45° increments (smallest available) to phase shift BCLK/BCLK90 relative to the SCLK.

   Transition detection mechanism is used to find positive edge to positive edge crossing between BCLK . DDRx2 mode requires 2 bits for the transition detection. The BCLK_IGEAR_RX[1] transition from 1 to 0 and BCLK_IGEAR_RX[0] transition from 0 to 1 or vice-versa is used for the transition detection.

   The following figure shows an example waveform which describes the transition detection and the relationship achieved through PLL_BCLKPHS_OFFSET[2:0]

Figure 11-17. BCLK90(Clk) SCLK(Dat)—Transition-Detection and SCLK_BCLK_BCLK90 Clk Relationship



3. Offset BCLK90 relative to SCLK is achieved by changing the PLL VCOPHSEL controls, appropriately. The BCLK to BCLK90 relationship is previously locked as defined in the preceding step and in the preceding figure.

   The IOD BCLK input pin always uses the BCLK90. The SCLK is sourced from the PLL and BCLKs are sourced from the PLL. Although, the SCLK phase is locked by the PLL and never is shifted, there may be significant PVT variations producing a need for re-training. Maximizing BCLK90 versus SCLK timing margin reduces the need for re-training.

   This training procedure defines the BCLK90 to SCLK phase relationship by moving BCLK90. When the BCLK90s phase is adjusted, BCLK90 can never be changed by any subsequent training. BCLK becomes fixed after this point.

11.4.4. CoreBclkSclkAlign Coarse Training Timing Diagram (Ask a Question)

The following timing diagram shows the training procedure result of coarse training with BCLK90 as clock and SCLK as data pin and defines the BCLK90 to SCLK phase relationship by moving BCLK90.

Figure 11-18. BCLK90(Clk) SCLK(Dat)—Transition Detection

11.4.4.1. CoreBclkSclkAlign Training Parameters (Ask a Question)

The following table lists the parameter and its description.

Table 11-2. CoreBclkSclkAlign Training Parameters

Parameter	Description
BCLKSCLK_TRN_DIR	• 1: BCLKSCLK training performed in the forward direction. • 0: BCLKSCLK training performed in the backward direction. **Note:** This parameter is used only for PLL-based clock training. The change in direction can mitigate potential divider pulse glitches as the backward direction lengthens the pulse width, thus reducing the likelihood of glitch occurrences.
BCLKSCLK_TRN_MODE	0 – PLL-based training (default) 1 – ICB Delay-based training
IOG_FABRIC_RATIO	2 – DDR by 2 mode 4 – DDR by 4 mode
CLK_ALGN_SKIP_TRNG	0 – ICB Delay-based training 1 – SKIP ICB Delay-based training This parameter is used in common for PLL/ICB Mode and used internally in the DUT-based on BCLKSCLK_TRN_MODE to connect either PLL or ICB block.
CLK_ALGN_HOLD_TRNG	0 – ICB Delay-based training 1 – HOLD ICB Delay-based training This parameter is used in common for PLL/ICB Mode and used internally in the DUT-based on BCLKSCLK_TRN_MODE to connect either PLL or ICB block.
BCLKSCLK_ICB_MODE	0 – 128 Tap delay-based training 1 – 256 Tap delay-based training (default) This parameter is used only for ICB-based clock training.
BCLKSCLK_ICB_TAP_WAIT_CNT	3 – Width of the tap delay counter (default) This parameter is used only for ICB-based clock training.

11.4.4.2. CoreBclkSclkAlign Training Ports (Ask a Question)

The following table lists the clock alignment common signals required for CoreBclkSclkAlign.

Table 11-3. CoreBclkSclkAlign Ports

Port Name	Type	Description
Clocks and Reset		
SCLK	Input	Fabric clock

Table 11-3. CoreBclkSclkAlign Ports (continued)

Port Name	Type	Description
RESETN	Input	Active low asynchronous reset
Data Bus and Control Signals – PLL and ICB Mode		
BCLKSCLK_TRAIN_START	Input	Set to 1 to start the clock training It is level-based; it must be driven high to start either PLL or ICB clock training mode .
BCLKSCLK_TRAIN_DONE	Output	Set to 1 at the end of the clock training and Reset to 0 when BCLKSCLK_TRAIN_START is 0 It is level-based; the DUT drives high at the end of clock training in PLL or ICB clock training mode.
BCLKSCLK_ALGN_PAUSE	Output	Set to 1 to reset the lane controller during start or rotate reached eight times (retrain) or at the completion of the clock training in both PLL and ICB mode.
CLK_ALGN_RSTRT	Input	0 – Disable HOLD-based training 1 – RESTART PLL-based or ICB-based training It is level-based; it must be driven high to restart the PLL or ICB clock training mode. This parameter is used in common for PLL/ICB mode and used internally in the DUT-based on BCLKSCLK_TRN_MODE to connect either PLL or ICB Block
CLK_ALGN_SKIP	Input	0 – Disable HOLD-based training 1 – SKIP PLL / ICB-based training It is level-based; it must be driven high to SKIP the PLL/ICB clock training mode. CLK_ALGN_SKIP_TRNG - parameter must be set to enable this feature. This parameter is used in common for PLL / ICB Mode and used internally in the DUT-based on BCLKSCLK_TRN_MODE to connect either PLL or ICB Block
CLK_ALGN_HOLD	Input	0 – Disable HOLD-based training 1 –HOLD PLL-based / ICB-based training It is level-based; it must be driven high to HOLD the PLL/ICB clock training mode. CLK_ALGN_HOLD_TRNG - parameter must be set to enable this feature. This parameter is used in common for PLL/ICB mode and used internally in the DUT-based on BCLKSCLK_TRN_MODE to connect either PLL or ICB Block.
Data Bus and Control Signals – PLL Mode Alone		
BCLK_IGEAR_RX[*-1:0]	Input	IOG_IOD RX_N for BCLK transition detection The parameter width must be set to either DDR2 or DDR4 fabric ratio to parametrize the width this input as required
PLL_BCLKPHS_OFFSET[2:0]	Input	BCLK VCO phase required for SCLK , BCLK , and BCLK90 clock relationship.
BCLKSCLK_BCLK_VCOPHSEL[6:0]	Output	The final phase used by the DUT is driven out and is mainly used in DEBUG mode for the PLL Clock Training mode.
PLL_VCOPHSEL_SCLK_SEL	Output	PLL VCO Phase select for SCLK
PLL_VCOPHSEL_BCLK_SEL	Output	PLL VCO Phase select for BCLK
PLL_VCOPHSEL_BCLK90_SEL	Output	PLL VCO Phase select for BCLK90
PLL_VCOPHSEL_MCLK_SEL	Output	PLL VCO Phase select for memory clock
PLL_LOADPHS	Output	PLL Phase load default values
PLL_PHS_ROTATE	Output	PLL Phase rotate Based on PLL_VCOPHSEL_X, the corresponding clock rotates in 45° shift for VCO/8 phase delays in forward or backward direction. BCLKSCLK_TRN_DIR parameter is used to select the direction. 0 - Backward Direction 1 - Forward Direction

Table 11-3. CoreBclkSclkAlign Ports (continued)

Port Name	Type	Description
PLL_PHS_DIRECTION	Output	• 1: Rotate phase forward. This is the default value. • 0: Rotate phase backward. **Note:** The algorithm uses the rotate phase in backward direction.
Data Bus and Control Signals –used for ICB Mode Alone		
IOD_EARLY	Input	Data eye monitor early flag
IOD_LATE	Input	Data eye monitor late flag
IOD_OOR	Input	Out of range flag for delay line
ICB_BCLKPHS_OFFSET[7:0]	Input	Used for BCLK Phase Alignment The value is used as added TAP delays in the IP for alignment
ICB_CLK_ALGN_CLR_FLGS	Output	Clear Early/Late flags
ICB_CLK_ALGN_LOAD	Output	PLL Load default values
ICB_CLK_ALGN_DIR	Output	Delay line up/down direction 1 – up (increment 1 tap) (default)
ICB_CLK_ALGN_MOV	Output	Increment the delay on move pulse
ICB_CLK_ALGN_ERR	Output	0 – No Error ICB Delay-based training 1 – Timeout Error in ICB Delay-based training
ICB_CLK_ALGN_TAPDLY[7:0]	Output	The final calculated TAP delays and ICB_BCLKPHS_OFFSET[7:0] is driven out by the DUT and is mainly used in DEBUG mode for ICB clock training mode.

The following table lists all the valid and invalid training use cases of the COREBCLKSCLKALGN IP.

Table 11-4. COREBCLKSCLKALGN IP Training Use Cases

COREBCLKSCLKALGN IP Skip/Hold/Restart Feature						
Parameter		Ports			Use Case	Notes
CLK_ALGN_SKIP_TRNG	CLK_ALGN_HOLD_TRNG	CLK_ALGN_SKIP	CLK_ALGN_HOLD	CLK_ALGN_RSTRT		
1	1	1	0	0	Valid	Skip training
1	1	1	0	1	Invalid	Training is skipped completely, then Restart training is not a valid use case.
1	1	1	1	0	Invalid	Training is skipped completely, then Hold training is not a valid use case.
1	1	1	1	1	Invalid	Training is skipped completely, then Hold/Restart training is not a valid use case.
1	1	0	1	0	Valid	Hold training whenever CLK_TRAIN_DONE is “0”.
1	1	0	1	1	Invalid	Both Hold/Restart training cannot occur at the same time.
1	1	0	0	1	Valid	Restart training whenever required.

12. Revision History (Ask a Question)

The revision history table describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

Revision	Date	Description
J	05/2025	The following is a summary of the changes made in this revision: • Added support for RT PolarFire and RT PolarFire SoC throughout the document. • Renamed the document title to "PolarFire Family User I/O User Guide". • Added 3 mA as one of the programmable output drive strengths for SUBLVDS33 and SUBLVDS25 in Table 7-6. • Updated Table 7-15 to reflect that since internal Vref is derived from the bank VDDI, single-ended I/O with Vref can only be supported when I/O standard is set to the correct VDDI. Added the relevant note. • Added information related to LVDS25 in LVDS section. • Updated I/O Calibration. • Updated the note in MIPI D-PHY Transmit Only (High-Speed and Low-Power) section as follows: – Added information about a `Vsim` command that can be used to stop the toggling of L0_LP_DATA and L0_LP_DATA_N signals of the PF_IOD_GENERIC_RX IP. – Added a testbench example to demonstrate the use of `defparam` for controlling individual IOD for MIPI. • Updated Figure 9-6 and Figure 9-7. • Added information related to the **Clock Selection** options of Figure 9-7 in the Basic I/O Configurator - PF_IO section.

Revision History (continued)

Revision	Date	Description
H	02/2025	The following is a summary of the changes made in this revision: • Updated the information regarding clamp diodes for GPIOs in Programmable Clamp Diode. • Updated Implementing Emulated Standards for Outputs by including details about Bus-LVDS Emulated (BLVDSE25) Output Modes. • Updated MIPI D-PHY Transmitting Interface (High-speed Only). • Updated the syntax of the `set_ioff` command in I/O Register Combining. • Updated Figure 7-5 by specifying that Zo represents effective PCB trace differential. • Updated the fail-safe related information, see Dynamic ODT or Fail-Safe LVDS. • Added a note regarding the differential ODT resistor tolerance when used within a GPIO bank containing true differential input pairs that use a mixture of Low and Mid VCM_RANGE settings. See Common Mode Voltage (Vcm) Settings. • Updated the CoreBclkSclkAlign Training IP related information, see CoreBclkSclkAlign Training IP.
G	07/2024	The following is a summary of the changes made in this revision: • Added LVDS18G to Table 7-8 and Table 7-9. • Changed the following attributes of `set_io` command from lower case to upper case in the mentioned sections: – `-SLEW` – `-RES_PULL` – `-SCHMITT_TRIGGER` – `-IMPEDANCE` – `-VCM_RANGE` – `-CLAMP_DIODE` Slew Rate Control, Programmable Weak Pull- Up/Down and Bus-Keeper (Hold) Circuits, Schmitt Trigger Input Hysteresis, Programmable Output Impedance Control, Common Mode Voltage (Vcm) Settings, and Programmable Clamp Diode.

Revision History (continued)

Revision	Date	Description
		• Added default values for the following attributes of the `set_io` command: – `-RES_PULL` – `-SCHMITT_TRIGGER` – `-OUT_DRIVE` – `-SOURCE_TERM` – `-ODT` – `-VCM_RANGE` – `-CLAMP_DIODE` • Added Table 7-3, Table 7-4, and a note about the default value of both input and output buffer for the switch `-RES PULL` in Programmable Weak Pull- Up/Down and Bus-Keeper (Hold) Circuits. • Added SLVS I/O Standard to Table 7-17 in I/O External Termination. • Added a note regarding the DC Input Levels (VIH and VIL) in Mixed I/O in VDDI Banks. • Added a note regarding a false scenario that occurs only during simulation, see MIPI D-PHY Transmit Only (High-Speed and Low-Power). • Added footnotes under Table 7-20. • Added a footnote referencing the AN4623 web link, to the STREAM_START port in Table 10-2 in the Transmit Interface. • Updated Figure 9-3. • Added the “Training Direction for BclkSclk” parameter in Table 9-3 • Updated Figure 10-12. • Updated Figure 11-15. • Updated Figure 11-16. • Added a new parameter “BCLKSCLK_TRN_DIR” in Table 11-2. • Updated the description of “PLL_PHS_DIRECTION” in Table 11-3. • Added Table 11-4.

Revision History (continued)

Revision	Date	Description
F	02/2024	The following is a summary of the changes made in this revision: • Updated Slew Rate Control as follows: – Changed the following sentence "Turning the slew rate on results to faster slew rate improves the available timing margin." to "In I/O Editor or the PDC file, when SLEW set to ON, the device uses a limited slew rate for the I/O standard in the device." – Changed the following sentence "When slew rate is turned OFF, the device uses the default slew rate to reduce the impact of simultaneous switching noise (SSN)." to "In I/O Editor or the PDC file, when SLEW set to OFF, the device uses the fastest slew rate for the I/O standard. The impacts to Simultaneous Switching Noise (SSN) can be reduced by using the SLEW set to ON. However, this reduces the maximum rate of change of the output signal that can influence switching performance." – Removed the reference to the datasheet. • Added a sentence in Open Drain GPIO to describe that the actual voltage at the PAD Output will be lower than 3.3V depending on the value of the pull-up resistor when the GPIO pin is not driving LOW. • Removed 3.3V VDDAUX supported value from LVDS in GPIO Banks with VDDI = 1.8V. • Added a new configuration option **Eye window override** in Table 10-3. • Updated information about Programmable Drive Strength PDC command. See Programmable Output Drive Strength.

Revision History (continued)

Revision	Date	Description
E	08/2023	The following is a summary of the changes made in this revision: • Information about MPFS devices were added. See Table 8-5 and PolarFire Family I/O Banks. • Information about fabric global clock for external source was updated. See Table 9-3. • Information regarding the ARST_N input to the generic TX block was added. See Figure 8-23 and Table 8-13. • Added a note about post layout simulation of the IOD. See IOD Generic TX. • Added information about using a TX_CCC for multiple IOD generic TX interfaces. See Table 9-7. • Information about I/O lanes was updated. See Table 8-5. • Information about supported I/O standards was updated. See Supported I/O Standards. • Added information about LVCMOS33. See Table 7-17. • Added information about I/O Initialization. • Updated information about RX_DDRX_B_G_C and RX_DDRX_B_G_A/RX_DDRX_B_R_A Interfaces with Static Delays. • Updated information about RX_DDR Fractional Aligned/Fractional Dynamic Interfaces. • Updated information about RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN. • Updated information about SELA_LANE[10:0] and CLR_FLAGS_N. See Table 10-4. • Added information about CoreBclkSclkAlign Training IP.

Revision History (continued)

Revision	Date	Description
D	07/2022	The following is a summary of the changes made in this revision. • Information about weak pull-up and pull-down recommendation was added. See Programmable Weak Pull- Up/Down and Bus-Keeper (Hold) Circuits and a footnote under Table 7-22. • Information about PolarFire Family I/O Banks was updated. • Information in the I/O Calibration section was updated. • Information about SGMII I/O was added. • Information about DDR I/O was added. • Information about ARST_N port was updated. See Table 8-10 and Table 8-11. • Information about HS_IO_CLK and System Clock Training was updated. • Information about Shield pins was updated. See Shield. • Information about low power detection of MIPI and low power escape mode was added. See Figure 7-8. • Information about protocol supported by MSSIO bank was added. See Table 7-21. • Information about training control ports was added. See Figure 9-3. • Information about transmit spread, transmit independent, and data/ control split was added. See Figure 9-5. • Information about Table 10-4 was updated. • Information about PF_IO macro was added. See I/O Registers. • Information about port names of Input registers and Output registers were updated. See Input Register and Output Register. • Information about fabric global clock for external source was added. See Figure 9-3. • Information about enable user control of clock pattern was added. See Table 9-7.

Revision History (continued)

Revision	Date	Description
C	12/2021	The following is a summary of the changes made in this revision. • Updated I/O Interface Timing Constraints. • The revision history tables of both the user guides are retained here for the future reference. For information, see Table 12-1 and Table 12-2.
B	10/2021	The following is a summary of the changes made in this revision. • Updated Shield. • Updated Open Drain GPIO. • Added the 3.3V Tolerant Input section. • Updated MIPI D-PHY Receive Interface. • Updated MSSIO. • Updated IOD CDR.
A	08/2021	The first publication of this document. This user guide was created by merging the following documents: • UG0686: PolarFire FPGA User I/O User Guide • UG0916: PolarFire SoC FPGA I/O User Guide

The following revision history table describes the changes that were implemented in the *UG0686: PolarFire FPGA User I/O User Guide* document. The changes are listed by revision.

Note: UG0686: PolarFire FPGA User I/O User Guide document is now obsolete and the information in the document has been migrated to PolarFire® FPGA and PolarFire SoC FPGA User I/O User Guide.

Table 12-1. Revision History of UG0686: PolarFire FPGA User I/O User Guide

Revision	Date	Description
Revision 7.0	4/21	The following is a summary of the changes in this revision. • Information about Sub-LVDS was updated. • Information about Programmable Weak Pull- Up/Down and Bus-Keeper (Hold) Circuits was updated. • Information about LVPECL25 IO Standard was removed from the document. • Information about HS_IO_CLK and System Clock Training was updated. • Information about Unused I/O Pins was added. • Information about IO Interface Timing Constraints was added. • Information about Full Duplex 1GbE and SGMII IOCDR was added. • Information about GPIO was updated in footnote of ODT Support in GPIO and HSIO table. • Information about HSIO was corrected in Cold Sparing. • Information about Dynamic ODT or Fail-Safe LVDS was updated. • Information about Transceiver Receivers, Transmitters and Reference Clock Inputs was added. • Information about CoreRxIODBitAlign Ports was updated. • Information about STREAM_START port was updated. See PF_IOD_CDR Interface Associated Ports table. • Information about Clock to Data Margin Training was updated. • Information about MIPI D-PHY Transmit Only (High-Speed and Low-power) was updated. • Information about HS_IO_CLK_PAUSE port was added to TX_DDR_G/B_A Interface Mode Ports table.
Revision 6.0	9/20	The following is a summary of the changes in this revision. • Information about Open Drain GPIO was added. • Information about Bit Slip was updated. • Information about MIPI D-PHY Transmit Only (High-Speed and Low-power) was updated. • Information about Implementing MIPI D-PHY was updated.
Revision 5.0	4/20	The following is a summary of the changes in this revision. • Information about Cold Sparing was updated. • Information about I/O Lanes in Each Bank was updated. • Information about I/O Clock Networks was updated. • Information about MIPI D-PHY Transmit Only (High-Speed and Low-power) was added. • Information about RX DDR Interfaces was updated. • Information about RX port and L#_RX_DATA[n:0] port was updated. • Information about PolarFire IOD CDR Clocking was added.
Revision 4.0	2/20	The following is a summary of the changes in this revision. • Information about ODT Control was updated. • Updated the IO Calibration section. • Added the section Dynamic ODT or Fail-Safe LVDS. • Updated the section Programmable I/O Delay. • Added the section IO Register Combining. • Updated the section High-Speed I/O Bank Clock Resource (HS_IO_CLK). • Updated the section Interface Selection Rules. • Updated the section Generic IOD Interface Implementation. • Updated the section Dynamic Delay Control. • Added the section Basic I/O Configurator. • Updated the section HS_IO_CLK and System Clock Training.

Table 12-1. Revision History of UG0686: PolarFire FPGA User I/O User Guide (continued)

Revision	Date	Description
Revision 3.0	5/19	The following is a summary of the changes in this revision. • Information about Static Timing Analysis was added. • Information about LVDS18 Receivers in GPIO was added. • Information about global clock and regional clock network was added. See PolarFire FPGA I/O Lanes. • Information about IO lanes in each bank was updated. • Information about Bit Slip was updated. • Information about HS_IO_CLK_PAUSE port was updated. • Information about Dynamic Delay Control ports was updated. • Information about RGMII to GMII Converter was added. • Information about LVDS 7:1 was added. • Information about PF_IOD_CDR was updated.
Revision 2.0	11/18	The following is a summary of the changes in this revision. • Information about PLL and DLL signals in PF_IOD_CDR Interface Associated Ports were added. • Information about failsafe logic for differential receivers was added. See Differential Receiver Mode. • Information about Supply Voltages for PolarFire FPGA I/O Banks was updated. • Information about Cold Sparing and Hot Swap was updated. • Information about flexible VDDI was added. See Mixed IO in VDDI Banks. • Information about MIPI25 IO standard was added. See Implementing MIPI D-PHY. • Information about PolarFire FPGA Generic I/O Interfaces was added. • Information about Generic IOD Interface Implementation was added. • Information about Software Primitives was added. • Information about HSIO data rate was added. • Information about IO lane in each bank was updated.
Revision 1.0	2/17	The first publication of UG0686: PolarFire FPGA User I/O User Guide.

The following revision history table describes the changes that were implemented in the *UG0916: PolarFire SoC FPGA User I/O User Guide* document. The changes are listed by revision.

Note: UG0916: PolarFire SoC FPGA User I/O User Guide document is now obsolete and the information in the document has been migrated to PolarFire® FPGA and PolarFire SoC FPGA User I/O User Guide.

Table 12-2. Revision History of UG0916: PolarFire SoC FPGA User I/O User Guide

Revision	Date	Description
Revision 3.0	4/21	The following is a summary of the changes in this revision. • Information about MSS DDR VREF was added. • Information about Sub-LVDS was updated. • Information about Programmable Weak Pull- Up/Down and Bus-Keeper (Hold) Circuits was updated. • Information about LVPECL25 IO Standard was removed from the document. • Information about HS_IO_CLK and System Clock Training was updated. • Information about Unused I/O Pins was added. • Information about IO Interface Timing Constraints was added. • Information about Full Duplex 1GbE and SGMII IOCDR was added. • Information about GPIO was updated in footnote of ODT Support in GPIO and HSIO table. • Information about HSIO was corrected in Cold Sparing. • Information about Dynamic ODT or Fail-Safe LVDS was updated. • Information about Transceiver Receivers, Transmitters and Reference Clock Inputs was added. • Information about CoreRxIODBitAlign Ports was updated. • Information about STREAM_START port was updated. See PF_IOD_CDR Interface Associated Ports table. • Information about Clock to Data Margin Training was updated. • Information about MIPI D-PHY Transmit Only (High-Speed and Low-power) was updated. • Information about HS_IO_CLK_PAUSE port was added to TX_DDR_G/B_A Interface Mode Ports table.
Revision 2.0	9/20	The following is a summary of the changes in this revision. • Information about Open Drain GPIO was added. • Information about Bit Slip was updated. • Information about MIPI D-PHY Transmit Only (High-Speed and Low-power) was updated. • Information about Implementing MIPI D-PHY was updated.
Revision 1.0	4/20	The first publication of UG0916: PolarFire SoC FPGA User I/O User Guide.

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-1275-6

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP’S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.