UG0686
User Guide
PolarFire FPGA User I/O

a Microchip company

a Microchip company
Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

50200686 . 4.0 2/20

Contents

1 Revision History 1
- 1.1 Revision 4.0 1
- 1.2 Revision 3.0 1
- 1.3 Revision 2.0 1
- 1.4 Revision 1.0 1

2 I/O Overview 2
- 2.1 GPIO and HSIO Features 2
  - 2.1.1 GPIO Features 2
  - 2.1.2 HSIO Features 2
- 2.2 Supported I/O Standards 3
  - 2.2.1 I/O Standard Descriptions 5
- 2.3 I/O Banks 8
- 2.4 Supply Voltages for PolarFire FPGA I/O Banks 10
- 2.5 PolarFire FPGA I/O Overview 11
  - 2.5.1 Single-Ended Transmitter and Receiver Mode 11
  - 2.5.2 Differential Transmitter Mode 11
  - 2.5.3 Differential Receiver Mode 12
  - 2.5.4 I/O Digital (IOD) 12
- 2.6 I/O Primitive 12

3 I/O Features and Implementation 13
- 3.1 I/O Analog (IOA) Buffer Programmable Features 14
  - 3.1.1 Slew Rate Control 14
  - 3.1.2 Programmable Weak Pull- Up/Down and Bus-Keeper (Hold) Circuits 14
  - 3.1.3 Schmitt Trigger Input Hysteresis 15
  - 3.1.4 Programmable Output Drive Strength 15
  - 3.1.5 Programmable Output Impedance Control 16
  - 3.1.6 Differential Near End Termination 17
  - 3.1.7 On-Die Termination (ODT) 17
  - 3.1.8 Common Mode Voltage (Vcm) Settings 18
  - 3.1.9 Programmable Clamp Diode 19
  - 3.1.10 Compensated Drive Impedance and Terminations 20
  - 3.1.11 SSTL25 and SSTL18 Stub Resistor 20
  - 3.1.12 Shield 21
- 3.2 I/O Implementation Considerations 21
  - 3.2.1 Reference Voltage for I/O Bank 21
  - 3.2.2 Mixed IO in VDDI Banks 22
  - 3.2.3 I/O External Termination 25
  - 3.2.4 Implementing Emulated Standards for Outputs 25
  - 3.2.5 Implementing MIPI D-PHY 27
  - 3.2.6 I/O States During Various Operational Modes 30
  - 3.2.7 Cold Sparing and Hot Socketing 30
  - 3.2.8 IO Glitches 31
  - 3.2.9 IO Calibration 31
  - 3.2.10 Dynamic ODT or Fail-Safe LVDS 32

4 IOD Features and User Modes 33
- 4.1 IOD Block Features 33
- 4.2 IOD Block Overview 33
  - 4.2.1 Programmable I/O Delay 34

4.2.2 I/O Registers . . . 36
4.2.3 I/O Gearing . . . 40
4.2.4 I/O FIFO . . . 40
4.3 PolarFire FPGA I/O Lanes . . . 41
4.3.1 Lane Controller . . . 41
4.3.2 I/O Lanes in Each Bank . . . 42
4.4 I/O Clock Networks . . . 42
4.4.1 Global Clock Resource . . . 43
4.4.2 Regional Clock Networks . . . 43
4.4.3 Lane Clock Resources . . . 43
4.4.4 High-Speed I/O Bank Clock Resource (HS_IO_CLK) . . . 43
4.4.5 Bit Slip . . . 44
4.5 PolarFire FPGA Generic I/O Interfaces . . . 45
4.5.1 RX DDR Interfaces . . . 45
4.5.2 RX_DDR_G_A/ RX_DDR_R_A—Aligned Interfaces with Static Delays . . . 46
4.5.3 RX_DDR_G_C and RX_DDR_R_C—Centered Interfaces with Static Delays . . . 49
4.5.4 RX_DDRX_B_G_C and RX_DDRX_B_G_A/RX_DDRX_B_R_A Interfaces with Static Delays . 51
4.5.5 RX_DDR Fractional Aligned Interfaces . . . 52
4.5.6 RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN . . . 53
4.5.7 TX DDR Interfaces . . . 55
4.6 Latency . . . 57
5 Generic IOD Interface Implementation . . . 58
5.1 Software Primitives . . . 58
5.1.1 Input DELAY . . . 58
5.1.2 Input Register (IREG) . . . 59
5.1.3 Input FIFO . . . 59
5.1.4 Input Gear Box . . . 59
5.2 IO Interface Configurators . . . 59
5.2.1 IOD Templates . . . 60
5.2.2 IOD Generic RX . . . 60
5.2.3 Dynamic Delay Control . . . 63
5.2.4 IOD Generic TX . . . 64
5.3 Basic I/O Configurator . . . 65
6 Protocol-Specific I/O Interfaces . . . 67
6.1 PF_IOD_CDR . . . 67
6.1.1 IOD CDR . . . 68
6.1.2 Receive Interface . . . 69
6.1.3 Transmit Interface . . . 69
6.1.4 Clock Sharing . . . 72
6.2 RGMII to GMII Converter . . . 73
6.3 LVDS 7:1 . . . 78
6.3.1 7:1 LVDS Receive Interface . . . 79
6.3.2 7:1 LVDS Transmit Interface . . . 80
7 Dynamic IOD Interface Training . . . 81
7.1 Clock to Data Margin Training . . . 81
7.1.1 HS_IO_CLK and System Clock Training . . . 82
7.2 CoreRxIODBitAlign . . . 84
7.2.1 CoreRxIODBitAlign Training Algorithm . . . 86
7.2.2 Interface Parameters . . . 87
7.2.3 CoreRxIODBitAlign Ports . . . 88

Figures

Figure 1 MPF300T, MPF300XT, and MPF500T Device I/O Banks . . . 8
Figure 2 MPF200T Device I/O Banks . . . 9
Figure 3 MPF100T Device I/O Banks . . . 9
Figure 4 PolarFire FPGA I/O Pair . . . 11
Figure 5 PolarFire FPGA I/O Pair (Detailed View) . . . 13
Figure 6 SSTL25 and SSTL18 Stub Resistor . . . 21
Figure 7 SLVSE System Diagram . . . 25
Figure 8 Bus-LVDSE System Diagram . . . 26
Figure 9 MLVDSE System Diagram . . . 26
Figure 10 LVPECL System Diagram . . . 27
Figure 11 MIPI D-PHY Receiver . . . 28
Figure 12 MIPI D-PHY Transmit Interface . . . 28
Figure 13 High-Speed Transmit with Bidirectional . . . 29
Figure 14 Dynamic ODT used for Failsafe LVDS . . . 32
Figure 15 IOD Configured for I/O Registers . . . 34
Figure 16 IOD Input Delay Example . . . 35
Figure 17 I/O Registers in IOD . . . 36
Figure 18 Input Register . . . 37
Figure 19 Output Register . . . 37
Figure 20 Enable Register . . . 38
Figure 21 I/O Register Combining from Place and Route Layout Options . . . 39
Figure 22 I/O Digital . . . 40
Figure 23 PolarFire FPGA I/O Lanes . . . 41
Figure 24 Distribution of the Lane Clock . . . 43
Figure 25 Distribution of the HS_IO_CLK . . . 44
Figure 26 Aligned Data and Clock Waveform . . . 46
Figure 27 RX_DDRX1 Aligned Interface Using Global Clock . . . 47
Figure 28 RX_DDRX1 Aligned Interface Using Regional Clock . . . 47
Figure 29 Centered Data and Clock Waveform . . . 49
Figure 30 RX_DDRX1 Centered Interface Using Global Clock . . . 49
Figure 31 RX_DDRX1 Centered Interface Using Regional Clock . . . 50
Figure 32 Rx_geared Waveform . . . 51
Figure 33 Fractional Aligned Data and Clock Waveform . . . 52
Figure 34 RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN Waveform . . . 54
Figure 35 Block Diagram of the RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN Interface . . . 54
Figure 36 TX_DDR_G_A Interface Signal Waveform—TXDDRX1 . . . 56
Figure 37 TX_DDR_G/B_A Interface Block Diagram—TX_DDRX1 . . . 56
Figure 38 IOD Modules used within a Generic DDRX I/O Interface . . . 59
Figure 39 IOD Generic Receive Interfaces—Configuration Tab . . . 61
Figure 40 IOD Generic Receive Interfaces—Advanced Tab . . . 62
Figure 41 IOD Generic Transmit Interfaces—Configuration Tab . . . 64
Figure 42 IOD Generic Transmit Interfaces—Advanced Tab . . . 65
Figure 43 PolarFire IO Configurator . . . 65
Figure 44 IO Configuration Tab . . . 66
Figure 45 PF_IOD_CDR Transmit and Receive Interface Modes . . . 67
Figure 46 SmartDesign of IOD_CDR Topology . . . 68
Figure 47 IOD CDR Configuration . . . 68
Figure 48 Using PF_IOD_CDR Interfaces . . . 71
Figure 49 Multiple PF_IOD_CDR Transmit and Receive Interfaces . . . 72
Figure 50 RGMII to GMII Block Diagram . . . 73
Figure 51 PolarFire RGMII to GMII Configurator . . . 75
Figure 52 RX_DDR_G_A Interface Configuration Tab—Used with RGMII To GMII Configuration . . . 75
Figure 53 RX_DDR_G_A Interface Advanced Tab—Used with RGMII To GMII Configuration . . . 76
Figure 54 RX_DDR_G_A Waveform . . . 76

Figure 55 TX_DDR_G_A Configuration Tab—Used with RGMII To GMII Configuration 77
Figure 56 TX_DDR_G_A Advanced Tab—Used with RGMII To GMII Configuration 77
Figure 57 Example of 7:1 LVDS Interface—Four Data and One Clock 78
Figure 58 RX_DDRX_B_G_FA Interface 79
Figure 59 TX_DDRX_B_A_X3.5 80
Figure 60 IOD Training Block Diagram 82
Figure 61 HS_IO_CLK to SYSCLK Data Transfer 82
Figure 62 SYSCLK to HS_IO_CLK Data Transfer 83
Figure 63 SYSCLK to HS_IO_CLK Before Training 83
Figure 64 SYSCLK to HS_IO_CLK After Training 83
Figure 65 CoreRxIODBitAlign Implementation Diagram 84
Figure 66 CoreRxIODBitAlign Training State Diagram 85
Figure 67 CoreRxIODBitAlign Libero Configurator 87

Tables

Table 1 Supported I/O . . . 3
Table 2 Supply Pin . . . 10
Table 3 Slew Rate Control . . . 14
Table 4 Weak Pull and Bus-Keeper Control . . . 15
Table 5 Schmitt Trigger Control . . . 15
Table 6 Programmable Drive Strength Control . . . 16
Table 7 Programmable Output Impedance Standards . . . 16
Table 8 Source Termination Support . . . 17
Table 9 ODT Support in GPIO and HSIO . . . 18
Table 10 Programmable Differential Termination Control . . . 19
Table 11 Programmable Clamp Diode . . . 20
Table 12 GPIO LVTTL/LVCMOS I/O Compatibility in Receive Mode . . . 23
Table 13 HSIO LVCMOS I/O Compatibility in Receive Mode . . . 23
Table 14 GPIO Mixed Reference Receiver Mode . . . 23
Table 15 HSIO HSUL12/HSTL12/POD I/O Compatibility in Receive Mode . . . 23
Table 16 GPIO Differential I/O Compatibility in Receive Mode . . . 24
Table 17 I/O External Termination with ODT Off . . . 25
Table 18 I/O States during Power-Up and Initialization . . . 30
Table 19 GPIO and HSIO States During Programming Modes . . . 30
Table 20 I/O Input Register Ports . . . 37
Table 21 I/O Output Register Ports . . . 38
Table 22 I/O Register Ports . . . 38
Table 23 I/O Lanes in Each Bank . . . 42
Table 24 PolarFire Device Support for Generic RX DDR interfaces, . . . 46
Table 25 RX_DDR Aligned Interface Mode Ports . . . 48
Table 26 RX_DDR Centered Interface Mode Ports . . . 50
Table 27 RX_DDR_B_C and RX_DDR_B_A Interface Mode Ports . . . 51
Table 28 Fractional Aligned Interface Mode Ports . . . 53
Table 29 RX_DDRX_L_DYN Ports . . . 55
Table 30 TX DDR Interfaces . . . 55
Table 31 TX_DDR_G/B_A Interface Mode Ports . . . 57
Table 32 Latency for the Rx/Tx CLK Interface . . . 57
Table 33 Receive Interface . . . 60
Table 34 IOD Generic Receive Interfaces—Configuration Tab . . . 61
Table 35 IOD Generic Receive Interfaces—Advanced Tab . . . 62
Table 36 Dynamic Delay Control Ports . . . 63
Table 37 Transmit Interface . . . 64
Table 38 IOD Generic Transmit Interfaces—Configuration Tab . . . 64
Table 39 IOD Generic Transmit Interfaces—Advanced Tab . . . 65
Table 40 PF_IOD_CDR Interface Associated Ports . . . 69
Table 41 IOD CDR Configuration . . . 69
Table 42 GMII Ports . . . 74
Table 43 CoreRxIODBitAlign Interface Parameters . . . 87
Table 44 CoreRxIODBitAlign Ports . . . 88

1 Revision History

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

1.1 Revision 4.0

The following is a summary of the changes in revision 4.0 of this document.

- Information about ODT Control was updated. See Table 9, page 18.
- Updated the section IO Calibration, page 31.
- Added the section Dynamic ODT or Fail-Safe LVDS, page 32.
- Updated the section Programmable I/O Delay, page 34.
- Added the section IO Register Combining, page 38.
- Updated the section High-Speed I/O Bank Clock Resource (HS_IO_CLK), page 43.
- Updated the section Interface Selection Rules, page 48.
- Updated the section Generic IOD Interface Implementation, page 58.
- Added new GUI items. See Table 35 on page 62.
- Updated the section Dynamic Delay Control, page 63
- Added the section Basic I/O Configurator, page 65
- Updated the section HS_IO_CLK and System Clock Training, page 82

1.2 Revision 3.0

The following is a summary of the changes in revision 3.0 of this document.

- Information about Static Timing Analysis, page 35 was added.
- Information about LVDS18 Receivers in GPIO, page 24 was added.
- Information about global clock and regional clock network was added. See PolarFire FPGA I/O Lanes, page 41.
- Information about IO lanes in each bank was updated. See Table 23, page 42.
- Information about Bit Slip, page 44 was updated.
- Information about HS_IO_CLK_PAUSE port was updated. See Table 28, page 53.
- Information about Dynamic Delay Control ports was updated. See Table 36, page 63.
- Information about RGMII to GMII Converter, page 73 was added.
- Information about LVDS 7:1, page 78 was added.
- Information about PF_IOD_CDR, page 67 was updated.

1.3 Revision 2.0

The following is a summary of the changes in revision 2.0 of this document.

- Information about PLL and DLL signals in PF_IOD_CDR Interface Associated Ports were added. See Table 41, page 69.
- Information about failsafe logic for differential receivers was added. See Differential Receiver Mode, page 12.
- Information about Supply Voltages for PolarFire FPGA I/O Banks, page 10 was updated.
- Information about Cold Sparing and Hot Socketing, page 30 was updated.
- Information about flexible VDDI was added. See Mixed IO in VDDI Banks, page 22.
- Information about MIPI25 IO standard was added. See Implementing MIPI D-PHY, page 27.
- Information about PolarFire FPGA Generic I/O Interfaces, page 45 was added.
- Information about Generic IOD Interface Implementation, page 58 was added.
- Information about Software Primitives, page 58 was added.
- Information about HSIO data rate was added. See overview section.
- Information about IO lane in each bank was updated. See Table 9, page 18 and Table 23, page 42.

1.4 Revision 1.0

The first publication of this document.

2 I/O Overview

PolarFire® device 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.

PolarFire devices have two types of user I/Os:

- General-purpose I/O (GPIO), which supports a wide range of I/O standards operating with supplies between 1.2 V to 3.3 V nominal. These I/Os operate at speeds of up to 1.066 Gbps for single-ended standards, and 1.6 Gbps using differential standards with -1 SPD graded devices.
- High-speed I/O (HSIO), which supports I/O standards operating with supplies between 1.2 V to 1.8 V. These I/Os are optimized for high-speed and support operations at speeds of up to 1.6 Gbps—single-ended inputs/outputs and differential inputs.

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 grounds to reduce noise leakage. In addition to GPIO and HSIO, a number of I/Os are associated with PolarFire FPGA system controller and with transceiver clocks and data pads. These I/Os are powered up independently of other user I/O banks. For more information, see *UG0677: PolarFire FPGA Transceiver User Guide, UG0714: PolarFire FPGA Programming User Guide*, *UG0722: PolarFire FPGA Packaging and Pin Descriptions User Guide* and *Package Pin Assignment Tables (PPATs)*.

This chapter describes the features and supported standards for each of these user I/O types, providing details about PolarFire FPGA I/O banks and I/O naming conventions.

2.1 GPIO and HSIO Features

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

2.1.1 GPIO Features

- Supports 1.2 V to 3.3 V 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 1.8 V and 1.5 V
- Differential static or dynamic termination of 100 Ω
- Cold-sparing and hot socketing (hot plug-in or hot-swapping) 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

2.1.2 HSIO Features

- Supports 1.2 V to 1.8 V 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 1.8 V, 1.5 V, 1.35 V, and 1.2 V
- 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 supports at up to 1333 Mbps and DDR4 support at up to 1.6 Gbps

2.2 Supported I/O Standards

PolarFire FPGA 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 1 • **Supported I/O**

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

Table 1 • **Supported I/O *(continued)***

I/O Standards	Receiver/Transmitter Modes	V_{DDI} (Nominal) Required	Bank Types	Applications
POD12I, POD12II	Receiver, Transmitter	1.2 V	HSIO	DDR4
Differential Standards				
LVDS33	Receiver	3.3 V	GPIO	General purpose
	Transmitter[3]	3.3 V	GPIO	General purpose
LVDS25	Receiver	2.5V	GPIO	General purpose
	Transmitter[3]	2.5V	GPIO	General purpose
LVDS18[4, 5]	Receiver	1.8 V	HSIO	General purpose
RSDS33	Receiver	3.3 V	GPIO	General purpose
	Transmitter[3]	3.3 V	GPIO	General purpose
RSDS25	Receiver	2.5 V	GPIO	General purpose
	Transmitter[3]	2.5 V	GPIO	General purpose
RSDS18[5]	Receiver	1.8 V	HSIO	General purpose
MINILVDS33	Receiver	3.3 V	GPIO	General purpose
	Transmitter[3]	3.3 V	GPIO	General purpose
MINILVDS25	Receiver	2.5V	GPIO	General purpose
	Transmitter[3]	2.5V	GPIO	General purpose
MINILVDS18[5]	Receiver	1.8 V	HSIO	General purpose
SUBLVDS33	Receiver	3.3 V	GPIO	General purpose
	Transmitter[3]	3.3 V	GPIO	General purpose
SUBLVDS25	Receiver	2.5V	GPIO	General purpose
	Transmitter[3]	2.5V	GPIO	General purpose
SUBLVDS18[5]	Receiver	1.8 V	HSIO	General purpose
PPDS33	Receiver	3.3 V	GPIO	General purpose
	Transmitter[3]	3.3 V	GPIO	General purpose
PPDS25	Receiver	2.5 V	GPIO	General purpose
	Transmitter[3]	2.5 V	GPIO	General purpose
PPDS18[5]	Receiver	1.8 V	HSIO	General purpose
SLVS33	Receiver	3.3 V	GPIO	General purpose
SLVS25	Receiver	2.5 V	GPIO	General purpose
SLVS18	Receiver	1.8 V	HSIO	General purpose
SLVSE15[6]	Transmitter	1.5 V	GPIO, HSIO	General purpose
HCSL33	Receiver	3.3 V	GPIO	General purpose
HCSL25	Receiver	2.5 V	GPIO	General purpose
HCSL18	Receiver	1.8 V	HSIO	General purpose
BUSLVDSE25[6]	Transmitter	2.5 V	GPIO	Multipoint backplane applications
MLVDSE25[6]	Transmitter	2.5 V	GPIO	Multipoint backplane applications
LVPECL33	Receiver	3.3 V	GPIO	Video graphics and clock distribution

Table 1 • **Supported I/O *(continued)***

I/O Standards	Receiver/Transmitter Modes	V_{DDI} (Nominal) Required	Bank Types	Applications
LVPECLE33[6]	Transmitter	3.3 V	GPIO	Video graphics and clock distribution
MIPI25	Receiver	2.5 V	GPIO	Consumer mobile applications
MIPIE25[6]	Transmitter	2.5 V	GPIO	Consumer mobile applications, High–speed Mode

1. Certain I/O standards are designed to support flexible V_{DDI} assignment, see Mixed IO in VDDI Banks, page 22.
2. This application is supported by the I/O Standard, however, the PolarFire 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 100 Ω resistor, and LVDS transmit outputs are not supported in HSIO banks.
5. These standards require an external voltage reference (V_{REF}) and require 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.

2.2.1 I/O Standard Descriptions

This section provides an overview for each of the I/O standards supported by PolarFire FPGA I/Os.

2.2.1.1 3.3 V Peripheral Component Interface (PCI)

PolarFire FPGA 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.

2.2.1.2 Low-Voltage TTL (LVTTL)

LVTTL is a general-purpose standard (EIA/JESD8-B) for 3.3 V applications. It uses an LVTTL input buffer and a push-pull output buffer. PolarFire FPGA 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 6, page 16.

2.2.1.3 Low-Voltage CMOS (LVCMOS)

LVCMOS is a general-purpose standard implemented in CMOS transistors. PolarFire devices support five different LVCMOS operational modes:

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

2.2.1.4 Stub Series Terminated Logic (SSTL)

Stub series terminated logic (SSTL) is a general-purpose memory bus standard. PolarFire devices support the following SSTL operational modes:

- **SSTL25I—**SSTL Class I-standard with V_{DDI} (nominal) = 2.5 V
- **SSTL25II—**SSTL Class II-standard with V_{DDI} (nominal) = 2.5 V
- **SSTL18I—**SSTL Class I-standard with V_{DDI} (nominal) = 1.8 V
- **SSTL18II—**SSTL Class II-standard with V_{DDI} (nominal) = 1.8 V
- **SSTL15I—**SSTL Class I-standard with V_{DDI} (nominal) = 1.5 V
- **SSTL15II—**SSTL Class II-standard with V_{DDI} (nominal) = 1.5 V

- **SSTL135I—**SSTL Class I-standard with V_{DDI} (nominal) = 1.35 V
- **SSTL135II—**SSTL Class II-standard with V_{DDI} (nominal) = 1.35 V

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 *DS0141: PolarFire FPGA Datasheet*.

2.2.1.5 High-Speed Transceiver Logic (HSTL)

HSTL is a general-purpose, high-speed bus standard (EIA/JESD8-6) with a signaling range between 0 V and 1.5 V, 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.

PolarFire devices support the following HSTL operational modes:

- **HSTL15I—**HSTL Class I-standard with V_{DDI} (nominal) = 1.5 V
- **HSTL15II—**HSTL Class II-standard with V_{DDI} (nominal) = 1.5 V
- **HSTL135I—**HSTL Class I-standard with V_{DDI} (nominal) = 1.35 V
- **HSTL135II—**HSTL Class II-standard with V_{DDI} (nominal) = 1.35 V
- **HSTL12I—**HSTL Class I-standard with V_{DDI} (nominal) = 1.2 V
- **HSTL12I—**HSTL Class II-standard with V_{DDI} (nominal) = 1.2 V

For more information about signal levels for the various HSTL I/O standards, see Table 1, page 3. also, see *DS0141: PolarFire FPGA Datasheet*.

Note: HSTL135 and HSTL12 are not part of the JEDEC specification; they are scaled from HSTL15. For more information about HSTL signal levels, see *DS0141: PolarFire FPGA Datasheet*.

2.2.1.6 High-Speed Unterminated Logic (HSUL)

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

2.2.1.7 Pseudo Open Drain (POD)

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

2.2.1.8 Low-Voltage Differential Signal (LVDS)

Low-voltage differential signaling (ANSI/TIA/EIA-644) is a high-speed, differential I/O standard. The voltage swing between two signal lines is approximately 350 mV. PolarFire FPGA GPIO supports LVDS receive and transmit modes. PolarFire FPGA HSIO supports LVDS receive mode with an external 100 Ω board termination, see I/O External Termination, page 25 for more information.

2.2.1.9 Reduced-Swing Differential Signal (RSDS)

Reduced-swing differential signaling 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 *DS0141: PolarFire FPGA Datasheet*.

While PolarFire devices support RSDS receive and transmit modes with GPIO, PolarFire FPGA HSIO supports RSDS receive mode with an external 100 Ω on-board termination.

2.2.1.10 Mini-LVDS

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. PolarFire FPGA GPIO supports mini-LVDS in both receive and transmit modes. PolarFire FPGA HSIO supports mini-LVDS only in the receive mode and requires an external resistor..

2.2.1.11 Sub-LVDS

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 200 mV compared to 350 mV for LVDS. The nominal common-mode voltage for sub-LVDS is 0.9 V, while it is 1.25 V for LVDS. PolarFire FPGA GPIO supports sub-LVDS in both receive and transmit modes. PolarFire FPGA HSIO supports sub-LVDS only in the receive mode and requires an external resistor.

2.2.1.12 Point-to-Point Differential Signaling (PPDS)

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.

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

2.2.1.13 Scalable Low-Voltage Signaling (SLVS)

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. PolarFire devices support the SLVS I/O standards in GPIO and HSIO banks, but an external resistor is required for transmitter mode. For more information, see Implementing Emulated Standards for Outputs, page 25.

2.2.1.14 High-Speed Current Steering Logic (HCSL)

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

2.2.1.15 Bus-LVDS (B-LVDS)/Multipoint LVDS (M-LVDS)

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.

PolarFire FPGA 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, page 26, and Multipoint Low-Voltage Emulated (MLVDSE25) Output Mode, page 26.

2.2.1.16 Low-Voltage Positive Emitter-Coupled Logic (LVPECL)

LVPECL is a 3.3 V 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 PolarFire FPGA GPIO, external board termination is required for the LVPECL outputs. For more information about LVPECL33, see LVPECL Emulated (LVPECLE33) Output Mode, page 27.

2.2.1.17 Mobile Industry Processor Interface (MIPI) D-PHY

MIPI is a serial communication interface used in camera and display applications. PolarFire devices support implementing the MIPI D-PHY standards in GPIO bank using an external termination. For more information, see Implementing MIPI D-PHY, page 27.

2.3 I/O Banks

Depending upon the device size, each PolarFire device has five, six, or eight user I/O banks. 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 V_{DDI} power supply, and the same V_{REF} 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, V_{REF}. 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, page 14.

Figure 1, page 8 through Figure 3, page 9 show simplified PolarFire device floorplans for each device, including the bank locations. These figures also show the corner block and transceiver block. The corner block includes CCCs and two PLLs and two DLLs each, providing flexible clock management and synthesis for the FPGA fabric, external system, and I/Os. Note that all banks are not available in all devices, see I/O Lanes in Each Bank, page 42 for more information. For more information about CCC and PolarFire FPGA transceivers, see *UG0684: PolarFire FPGA Clocking Resources User Guide* and *UG0677: PolarFire FPGA Transceiver User Guide*.

Figure 1 • **MPF300T, MPF300XT, and MPF500T Device I/O Banks**

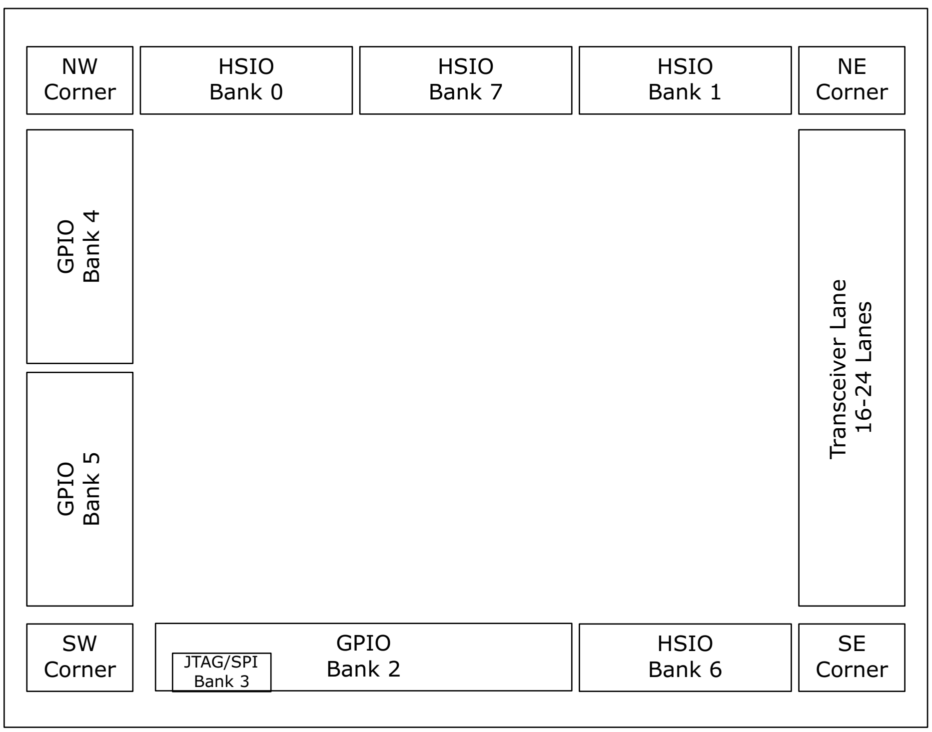


Figure 2 • **MPF200T Device I/O Banks**

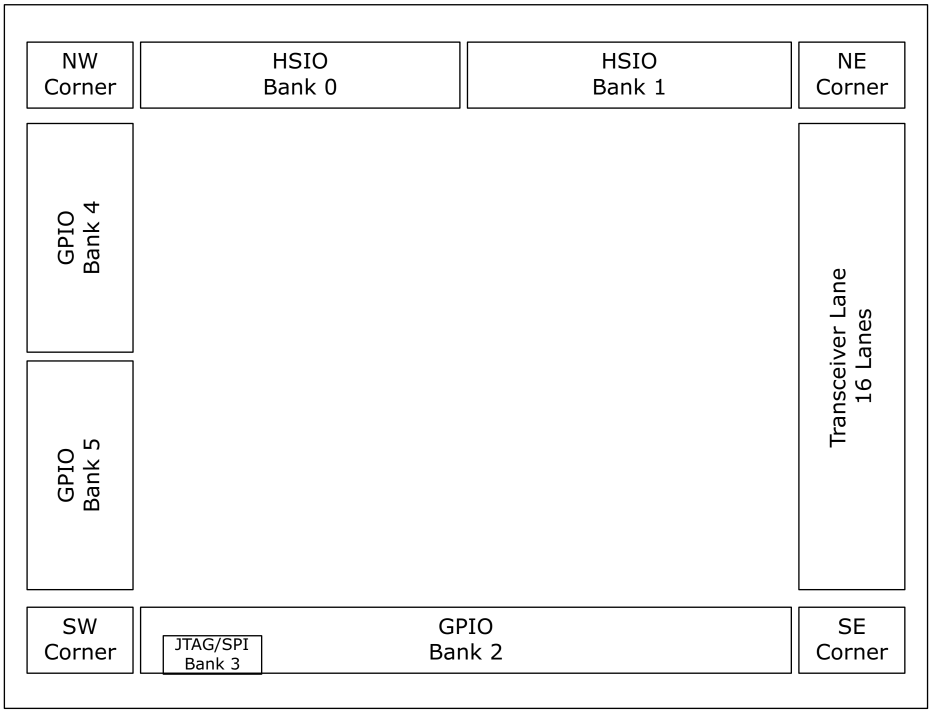
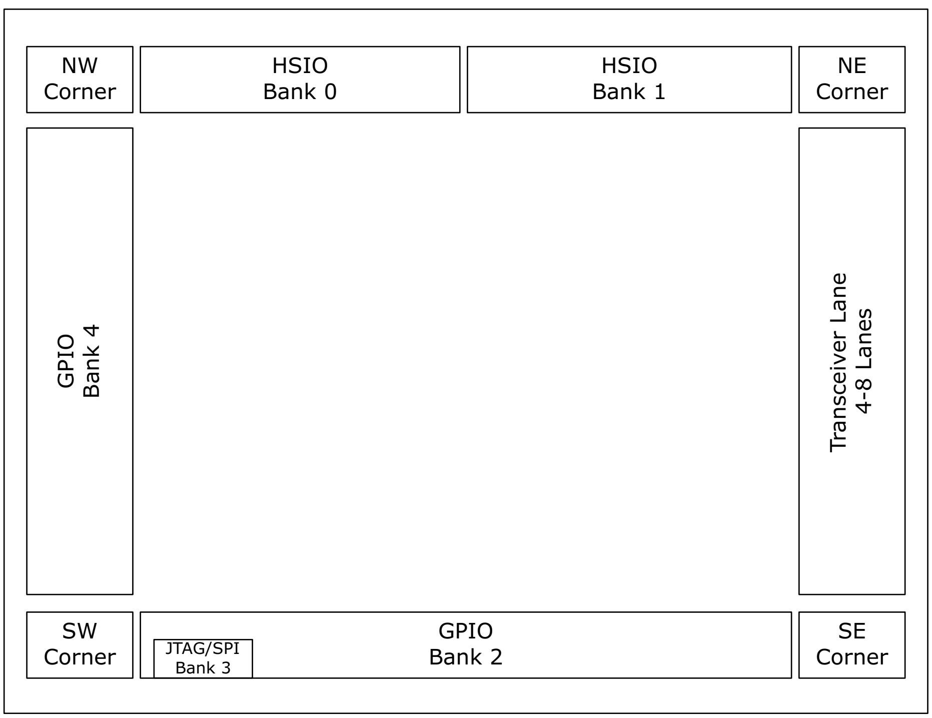


Figure 3 • **MPF100T Device I/O Banks**



2.4 Supply Voltages for PolarFire FPGA I/O Banks

PolarFire devices have multiple I/O banks that require the following bank power supplies listed in the table:

Table 2 • **Supply Pin**

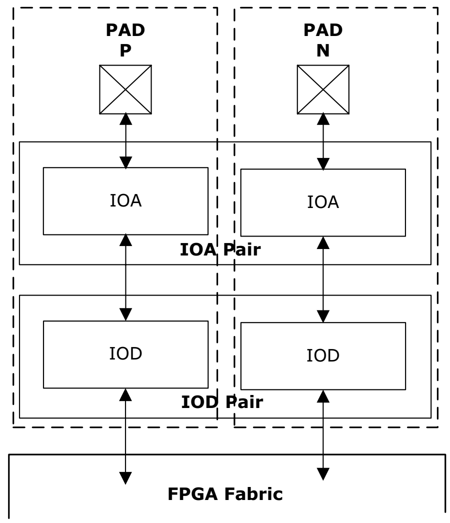
Name	Description	Operating Voltage	Unused Condition
V_{DDIx}	Supply for I/O circuits in a bank	For JTAG bank—1.8 V/2.5 V/3.3 V For GPIO bank—1.2 V/1.5 V/1.8 V/2.5 V/3.3 V For HSIO bank—1.2 V/1.5 V/1.8 V	The bank supply is not allowed to be left floating for any unused bank. It must be powered down to ground by direct connection
V_{DD25}	Power for corner PLLs and PNVM	2.5 V	Must connect to 2.5 V
V_{DD18}	Power for programming and HSIO receiver	1.8 V	Must connect to 1.8 V
V_{DDAUXx}	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 $V_{DDIx.}$ See Table 14, page 23 GPIO Mixed Reference Receiver Mode for legal VDDI and VDDAUX combinations.	Greater than or equal to $V_{DDI.}$ In cases where VDDI and VDDAUX in a given GPIO bank are both 2.5 V or 3.3 V, they should be tied together to same supply.	Must connect to greater than or equal to V_{DDI}
V_{REF}	V_{REF} is the supply reference voltage for reference receivers. Each bank can have only one V_{REF} value. V_{REF} can be externally supplied or internally generated, see Supply Voltages for PolarFire FPGA I/O Banks, page 10 for V_{REF} assignment use model for more information.	Depends on the I/O standards	The regular I/Os are used as V_{REF} supply. Libero® configures unused user I/Os as input buffer disabled and output buffer tri-stated with weak pull-up.

2.5 PolarFire FPGA I/O Overview

Each PolarFire FPGA 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, PolarFire FPGA 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 4 • **PolarFire FPGA 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 5, page 13 shows the overview of IOA buffer.

2.5.1 Single-Ended Transmitter and Receiver Mode

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

2.5.2 Differential Transmitter Mode

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 PolarFire FPGA GPIO bank supports true differential output modes using a differential current driver. Both, PolarFire FPGA GPIO and HSIO banks support complementary output modes.

2.5.3 Differential Receiver Mode

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 13. Differential receivers do not contain any failsafe logic and if an IO is programmed as a differential input (that is, LVDS25), it should not be left floating.

2.5.4 I/O Digital (IOD)

The IOD block interfaces with the FPGA fabric on one side and the IOA buffers on the other side, and 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 PolarFire FPGA I/O digital block works in conjunction with fast and low-skew clock networks. It also includes special clock dividers and other support 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 33.

2.6 I/O Primitive

The PolarFire FPGA 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 tri-state buffer
- **TRIBUFF_DIFF—**represents differential tri-state buffer

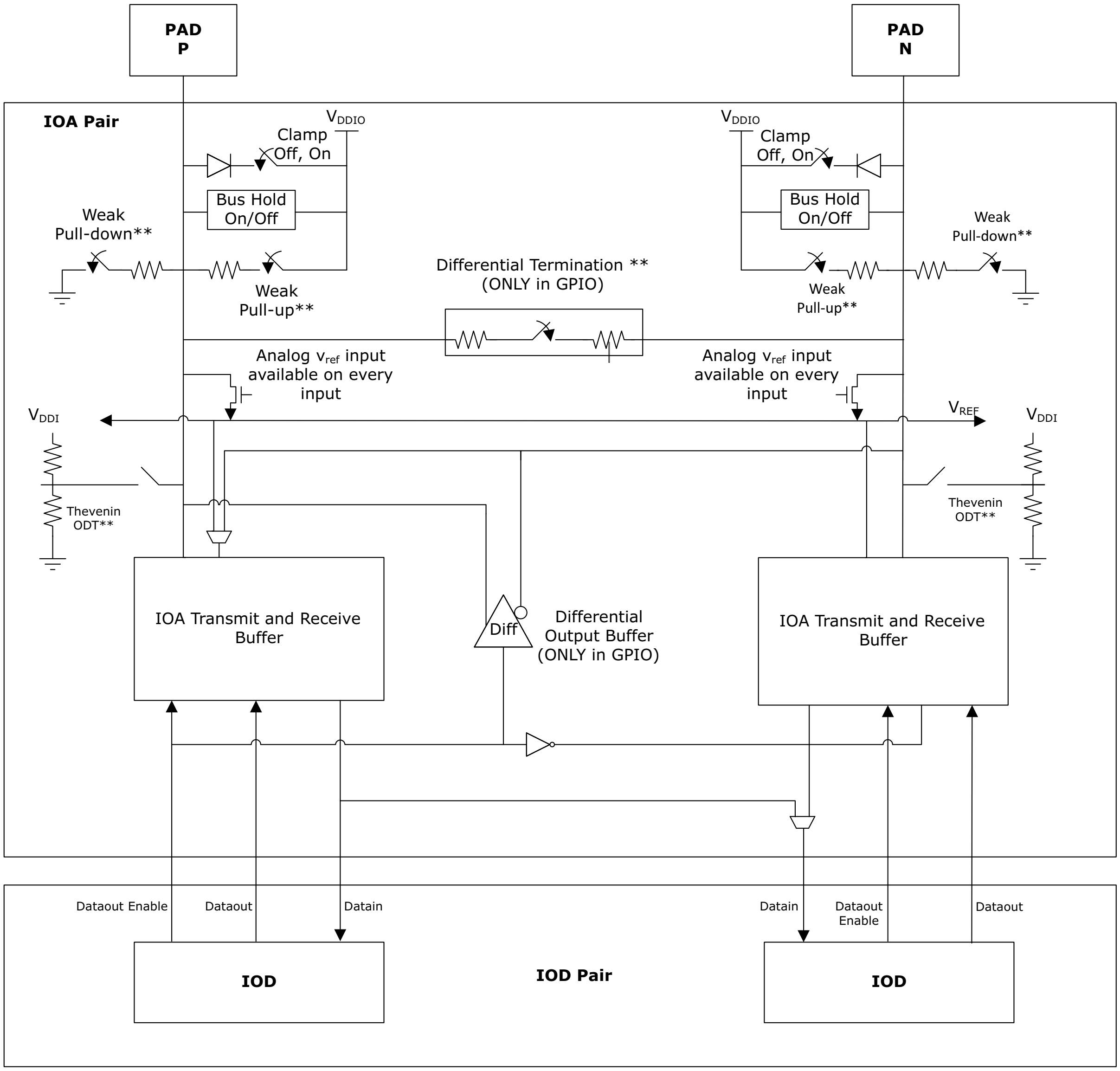
For more information about macro library, see *PolarFire FPGA Macro Library User Guide*.

3 I/O Features and Implementation

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

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

Figure 5 • **PolarFire FPGA I/O Pair (Detailed View)**



Note: The weak pull-up, pull-down, and on-die termination (ODT) ranges are listed in *DS0141: PolarFire FPGA Datasheet*.

3.1 I/O Analog (IOA) Buffer Programmable Features

PolarFire FPGA 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 on PDC constraints, see *UG0715: PDC commands User Guide*.

3.1.1 Slew Rate Control

PolarFire FPGA GPIO supports slew rate control in non-differential output mode. Turning the slew rate on results in faster slew rate, which improves the available timing margin, see *DS0141: PolarFire FPGA Datasheet* for the timing data. When slew rate is turned off, the device uses the default slew rate to reduce the impact of simultaneous switching noise (SSN).

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

Table 3 • **Slew Rate Control**

I/O Standards	Supported I/O Type	Slew Rate Control 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 PolarFire FPGA HSIO buffers. However, these buffers have built-in PVT-compensated slew rate controllers for optimized signal integrity.

3.1.2 Programmable Weak Pull- Up/Down and Bus-Keeper (Hold) Circuits

PolarFire devices have a programmable weak pull-down (20 KΩ typical) typical, pull-up (20 KΩ typical), and bus-keeper circuit on every I/O pad when in input mode. Weak pull-up and pull-down circuits create a default setting for an input when it is not 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. The programmable weak pull-down, pull-up, and bus-keeper circuits are disabled when the output driver is enabled. Differential inputs do not support the programmable weak pull-down, pull-up, and bus-keeper modes. An erased or blank device enables the weak pull-up resistor by default and tristate the output buffer. Unused I/O are programmed by default to enable the weak pull-up resistor and tristate the output buffer.

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

Table 4 • **Weak Pull and Bus-Keeper Control**

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

3.1.3 Schmitt Trigger Input Hysteresis

PolarFire FPGA 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 *DS0141: PolarFire FPGA Datasheet*.

Table 5 • **Schmitt Trigger Control**

I/O Standards	Supported I/O Types	Schmitt Trigger Control Options
LVTTL LVCMOS33 LVCMOS25 PCI	GPIO (input only)	On Off
LVCMOSI15 V LVCMOSI18 V	HSIO (input only)	On 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.

3.1.4 Programmable Output Drive Strength

For LVCMOS, LVTTL, LVDS, and PPDS I/O standards, the PolarFire FPGA 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 in PolarFire devices.

Table 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
LVCMOS33	GPIO (output only)	2, 4, 8, 12, 16, 20
LVCMOS25	GPIO (output only)	2, 4, 6, 8, 12, 16
LVDS25 and LVDS33	GPIO (output only)	3, 3.5, 4, 6[1]
RSDS33 and RSDS25	GPIO (output only)	1.5, 2, 3
MINILVDS33 and MINILVDS25	GPIO (output only)	3, 3.5, 4, 6
SUBLVDS33 and SUBLVDS25	GPIO (output only)	1, 1.5, 2
PPDS33 and PPDS25	GPIO (output only)	1.5, 2, 3
LVCMOS18	GPIO and HSIO (output only)	2, 4, 6, 8, 10, 12
LVCMOS15	GPIO and HSIO (output only)	2, 4, 6, 8, 10
LVCMOS12[2]	GPIO and HSIO (output only)	2, 4, 6, 8, 10

1. Recommendation to use 100 Ω 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 6, page 16.

3.1.5 Programmable Output Impedance Control

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

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

Table 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

Table 7 • **Programmable Output Impedance Standards** ***(continued)***

I/O Standards	Supported I/O Types	Impedance (Ω)
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, page 16.

3.1.6 Differential Near End Termination

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 8 • **Source Termination Support**

I/O Standard	Values
LVDS25, LVDS33, MINILVDS25, MINILVDS33, LCMDS33, LCMDS25, PPDS25, PPDS33, RSDS25, RSDS33, SUBLVDS25, SUBLVDS331	OFF, 100. The default is 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:

```
[-SOURCE_TERM <value>]
```

3.1.7 On-Die Termination (ODT)

ODT is used to terminate input signals, helping to maintain signal quality, saving board space, and reducing external component costs. In PolarFire FPGAs, 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 PolarFire FPGA I/O standards may require external termination for better signal integrity. For more information, see I/O External Termination, page 25.

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 PolarFire devices.

Table 9 • **ODT Support in GPIO and HSIO**

I/O Standards	I/O Types (Input Only)	ODT Control	ODT Type	ODT (Ω)
LVDS33, LVDS25 RSDS33, RSDS25, MINILVDS33, MINILVDS25, SUBLVDS33, SUBLVDS25, LVPECL33,LVPECL25	GPIO, HSIO	Off On Dynamic	Off Differential	100
SSTL18I, SSTL18II	GPIO, HSIO	Off On Dynamic	Off Thévenin	50, 75, 150
SSTL15I, SST15II	GPIO, HSIO	Off On Dynamic	Off Thévenin	20, 30, 40, 60, 120
SSTL135I, SSTL135II	HSIO	Off On Dynamic	Off Thévenin	20, 30, 40, 60, 120
POD12I, POD12II	HSIO	Off On Dynamic	Off Up	34, 40, 48, 60, 120, 240
HSUL12I, HSUL12II	HSIO	Off On Dynamic	Off Up	120, 240
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	Off Up Down Thévenin	60, 120, 240

Note: HSIO banks can support 2.5 V and 3.3 V inputs with VDDI = 1.8 V 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 9, page 18.

3.1.8 Common Mode Voltage (Vcm) Settings

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

The following table lists the programmable differential termination control support and settings in PolarFire devices. For more information about common mode voltage levels for various I/O standards, see *DS0141: PolarFire FPGA Datasheet*.

Table 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	
SLVS25	GPIO, HSIO	MID (HSIO) Low, Mid (GPIO)
HCSL25	GPIO, HSIO	MID (HSIO) Low, Mid (GPIO)
SLVSE	GPIO, HSIO	Off, Mid (HSIO) Off, Low, Mid (GPIO)
PPDS25	GPIO, HSIO	Mid (HSIO) Off, Low, Mid (GPIO)
MLVDSE	GPIO	Off, Low, Mid
BUSLVDS	GPIO	Off, Low, Mid
LVPECL	GPIO	Low, Mid
LVDS	GPIO, HSIO	Mid (HSIO) Off, Low, Mid (GPIO)
RSDS	GPIO, HSIO	Mid (HSIO) Off, Low, Mid (GPIO)
MINILVDS	GPIO, HSIO	Mid (HSIO) Off, Low, Mid (GPIO)

1. For more information about low and mid differential termination types, see *DS0141: PolarFire FPGA Datasheet*.

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 10, page 19.

3.1.9 Programmable Clamp Diode

PolarFire devices have internal PCI clamp diodes for both HSIO and GPIO. PCI 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 V_{DDIx} of the bank, the clamp diode must be off to support hot-socketing insertion, see Cold Sparing and Hot Socketing, page 30 for more information.

For GPIO, 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 clamp diodes that are programmable in PolarFire I/O devices.

Table 11 • **Programmable Clamp Diode**

I/O Standards	Supported I/O Type	Clamp Diode Control
LVTTL, LVCMOS33, LVCMOS25, LVCMOS18, LCMOS15, LVCMOS12, SSTL25, SSTL18I, SSTL18II, SSTL15I, SSTL15II, HSTL15I, HSTL15II	GPIO	Off, On

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

```
set_io – -clamp_diode <value>
```

value can be set as listed in Table 11, page 20.

Note: The clamp diode is always on for HSUL18I, HSUL18II, SLVSE15, MIPI25, PCI, SLVS33, HCSL33, MIPIE25, LVPECL33, LVPECL25, LVPECLE33, LVDS25, LVDS33, RSDS25, RSDS33, MINILVDS25, MINILVDS33, SUBLVDS25, SUBLVDS33, PPDS25, PPDS33, MLVDSE25, and BUSLVDSE25 I/O standards implemented in GPIO bank.

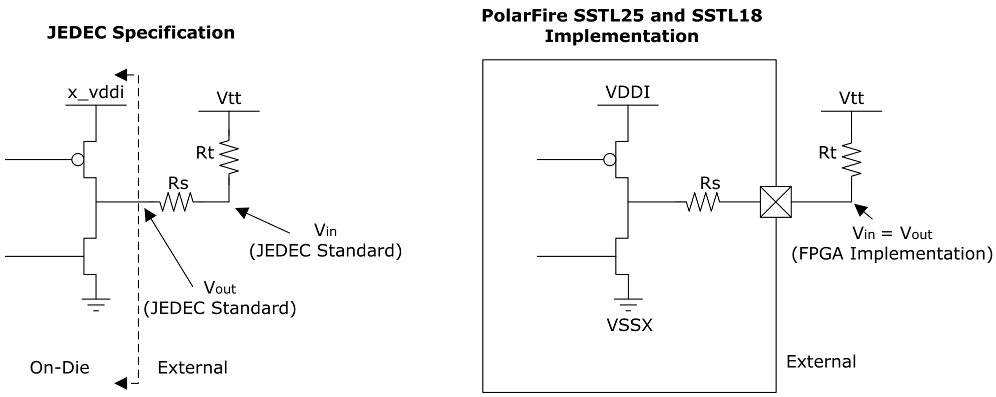
3.1.10 Compensated Drive Impedance and Terminations

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, PolarFire devices have a reference controller between the VDDI power supply and pad signal 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 compensate for Temperature variation and Supply Voltage fluctuations.

3.1.11 SSTL25 and SSTL18 Stub Resistor

For stub-series interface standard SSTL, the output drive also includes the stub resistor. PolarFire FPGA I/Os support this stub resistor for SSTL25 and SSTL18 I/O standards (Figure 6, page 21). This feature reduces both cost and board complexity.

Figure 6 • **SSTL25 and SSTL18 Stub Resistor**



3.1.12 Shield

Shield IOTYPE are provided for "soft ground" pins to improve the localized references. These are actual IO pins that are re-purposed to isolate switching noises around high-speed IO interfaces. Shields are implemented on memory interfaces on the unused DQ bits. 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.

3.2 I/O Implementation Considerations

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

3.2.1 Reference Voltage for I/O Bank

Each voltage-referenced I/O standard needs a reference voltage (V_{REF}) for inputs while in operation. Each bank in a PolarFire device 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.

3.2.1.1 External V_{REF} Input

Any PolarFire FPGA GPIO or HSIO pad on the device can be programmed to act as an external V_{REF} 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 uses the internal VREF.

Use PDC or the IO Attribute editor to choose any regular IO 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
```

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

Any available package pin can be selected and set it as a V_{REF}. This requires placement of at least one IO type requiring a V_{REF} in IOeditor or pdc.

For more information about external reference inputs, see *UG0726: PolarFire FPGA Board Design User Guide*.

3.2.1.2 Internally-Generated V_{REF}

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

3.2.2 Mixed IO in VDDI Banks

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 provide independence for mixing I/O standards in the same bank.

PolarFire 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-plug design. For example, when VDDI is set to 3.3 V, a input receiver of 3.3 V, 2.5 V, 1.8 V, and 1.2 V. LVCMOS can be placed in the same I/O bank.

The mixing of different IO within a bank is supported by the Libero software. Before placing any mixed IO voltage, the user should first set the bank to the desired VDDI voltage followed by setting the attributes of the IO that allows for mixed mode. Placing the IO should be the last step. When implementing mixed IO mode restrictions on ODT, CLAMP and RES_PULL must be followed. The HSIO receivers have a reduced set of compatible I/O standards when 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.

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

Table 12 • **GPIO LVTTL/LVCMOS I/O Compatibility in Receive Mode**[1]

V_{DDI}	LVTTL/LVCMOS33	LVCMOS25	LVCMOS18	LVCMOS15	LVCMOS12
3.3 V	Yes	Yes[2]	Yes[2]	No	Yes[2]
2.5 V	Yes	Yes	Yes	Yes	Yes
1.8 V	Yes	Yes	Yes	Yes	Yes
1.5 V	Yes	Yes	Yes	Yes	Yes
1.2 V	Yes	Yes	No	Yes	Yes

1. RES_PULL must be DOWN or NONE. All mixed modes above require CLAMP = OFF.
2. ODT must be OFF.

Table 12, page 23 shows the compatible IO types when mixing within the VDDI banks. Using the table for example, a VDDI low voltage of 1.2 V in GPIO can include LVCMOS33 inputs. Similarly, a VDDI low voltage of 1.2 V cannot include LVCMOS18 inputs.

Table 13 • **HSIO LVCMOS I/O Compatibility in Receive Mode**[1]

V_{DDI}	LVCMOS18	LVCMOS15	LVCMOS12
1.8 V	Yes	Yes	Yes
1.5 V	No	Yes	Yes
1.35 V	No	No	Yes
1.2 V	No	No	Yes

1. RES_PULL must be DOWN or NONE. All mixed modes above require CLAMP = ON.

The following table lists GPIO mixed reference receiver mode data.

Table 14 • **GPIO Mixed Reference Receiver Mode**[1]

V_{DDI}	V_{DDAUX}	SSTL25	SSTL18, HSUL18	SSTL15, HSTL15
3.3 V	3.3 V	No	No	No
2.5 V	2.5 V	Yes (mid-range Vcm)	Yes (mid-range Vcm)	Yes (Low-range Vcm)
1.8 V	2.5 V	Yes (mid-range Vcm and clamp diode off)	Yes (mid-range Vcm)	Yes (Low-range Vcm)
1.5 V	2.5 V	Yes (mid-range Vcm and clamp diode off)	Yes (mid-range Vcm and clamp diode off)	Yes (Low-range Vcm)
1.2 V	2.5 V	No	No	No

1. ODT must be OFF for all cases.

Table 15 • **HSIO HSUL12/HSTL12/POD I/O Compatibility in Receive Mode**[1]

V_{DDI}	SSTL15 HSUL15	SSTL18 HSTL18	SSTL135 HSTL135	HSUL12 HSTL12 POD
1.8 V	Yes (mid-range Vcm)	Yes (mid-range Vcm)	Yes (mid-range Vcm)	Yes (mid-range Vcm)

Table 15 • **HSIO HSUL12/HSTL12/POD I/O Compatibility in Receive Mode[1] *(continued)***

V_{DDI}	SSTL15 HSUL15	SSTL18 HSTL18	SSTL135 HSTL135	HSUL12 HSTL12 POD
1.5 V	Yes (mid-range Vcm)	No	Yes (mid-range Vcm)	Yes (mid-range Vcm)
1.35 V	No	No	Yes (mid-range Vcm)	Yes (mid-range Vcm)
1.2 V	No	No	No	Yes (mid-range Vcm)

1. ODT must be OFF for all cases.

Table 16 • **GPIO Differential I/O Compatibility in Receive Mode**

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

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

HSIO differential receivers do not support mixed IO voltage combinations.

3.2.2.1 LVDS

GPIO and HSIO banks can receive LVDS input signals. For GPIO, these inputs have an internal 100 Ω differential termination resistor that can be enabled by the Libero software. HSIO does not have this internal resistor capability. HSIO requires a 100 Ω 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.

LVDS outputs a natively available in GPIO banks. Use either VDDI=2.5V or 3.3V (LVDS25 or LVDS33). These are true LVDS transmitters. For information about DC specification, see the *DS0141: PolarFire FPGA Datasheet*. LVDS outputs are not available in HSIO banks.

3.2.2.2 LVDS18 Receivers in GPIO

The LVDS18 in the GPIO banks are indirectly supported from Libero. There is no explicit LVDS18 IOTYPE selection for GPIO in Libero, however, the silicon device does support it. A user can have a GPIO bank configuration with the board VDDI = 1.8 V and VDDAUX = 2.5 V (see *DS0141: PolarFire FPGA Datasheet*). It requires the selection of LVDS25 as IO TYPE and VDDI=2.5V in IOEditor or pdc. In this case, the Clamp Diode need to be OFF first before placing an LVDS25 on a bank with VDDI=1.8.

3.2.3 I/O External Termination

If ODT is not used or not available, PolarFire FPGA 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 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	No external termination required
LVDS	100 Ω, parallel termination (HSIO only)
MLVDS	100 Ω, parallel termination (HSIO only)
BLVDS	100 Ω, parallel termination (HSIO only)
RLVDS	100 Ω, parallel termination (HSIO only)
Mini-LVDS	100 Ω, parallel termination (HSIO only)
LVPECL	100 Ω, parallel termination (HSIO only)

3.2.4 Implementing Emulated Standards for Outputs

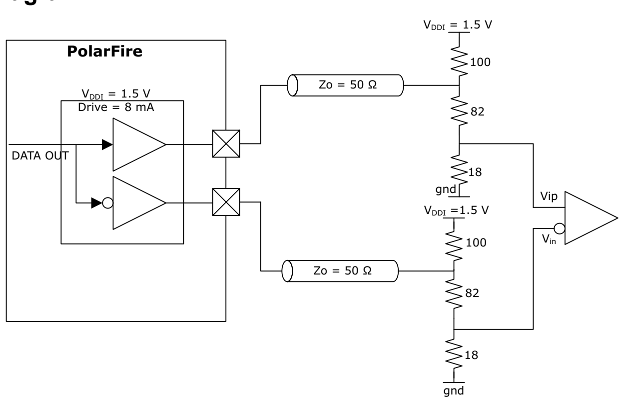
PolarFire devices require external terminations to implement SLVSE, BLVDSE, MLVDSE, and LVPECLE output modes. These outputs, referred to as emulated differential outputs, are noted in Table 5, page 15.

Emulated differential standards use compensated push-pull drivers in complementary output mode, and require external terminations on the board to match the comm-on-mode and voltage swing to meet the IO signal standards. This section provides example implementations for the emulated standards.

3.2.4.1 Scalable Low-Voltage Signaling Emulated (SLVSE15) Output Mode

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

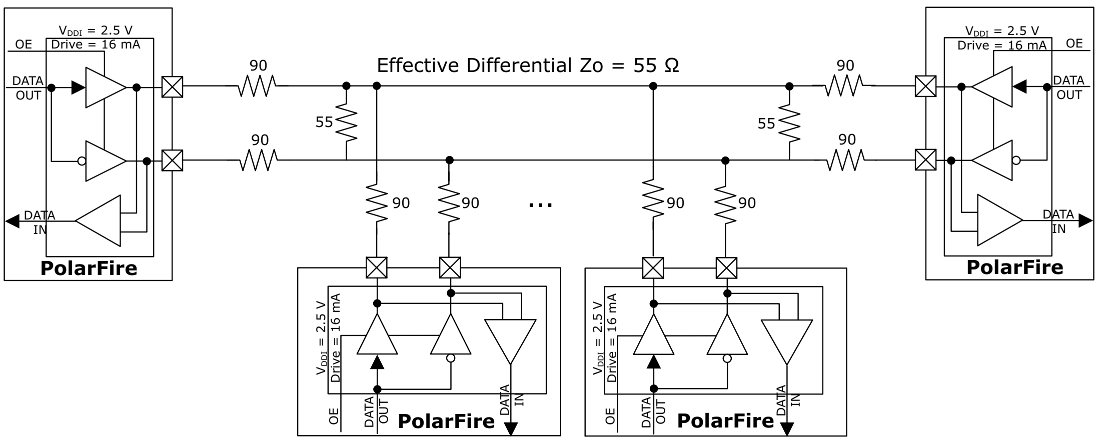
Figure 7 • **SLVSE System Diagram**



3.2.4.2 Bus-LVDS Emulated (BLVDSE25) Output Mode

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 PolarFire FPGA BLVDS implementation using 90 Ω stub resistors at every drop and 55 Ω stub resistors on either side of the bus. The termination values at the ends of the bus, which can range anywhere between 45 Ω and 90 Ω, must be optimized to match the effective differential impedance of the bus. In this example, the two parallel 55 Ω stub resistors yield an effective 27 Ω differential termination.

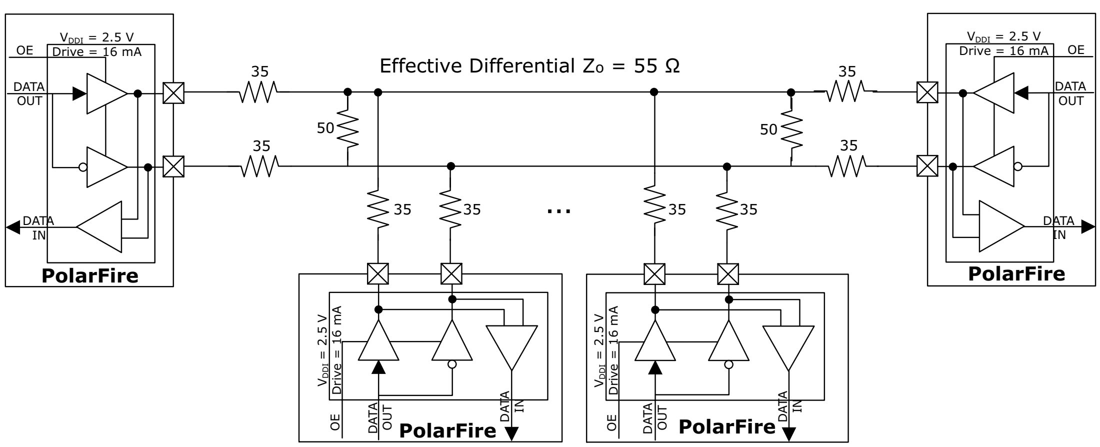
Figure 8 • Bus-LVDSE System Diagram



3.2.4.3 Multipoint Low-Voltage Emulated (MLVDSE25) Output Mode

MLVDS has larger signaling amplitude when compared to BLVDS, and therefore, 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 35 Ω stub resistors at every drop and 50 Ω stub resistors on either side of the bus. The termination values at the ends of the bus, which can range anywhere between 50 Ω and 70 Ω, must be optimized to match the effective differential impedance of the bus.

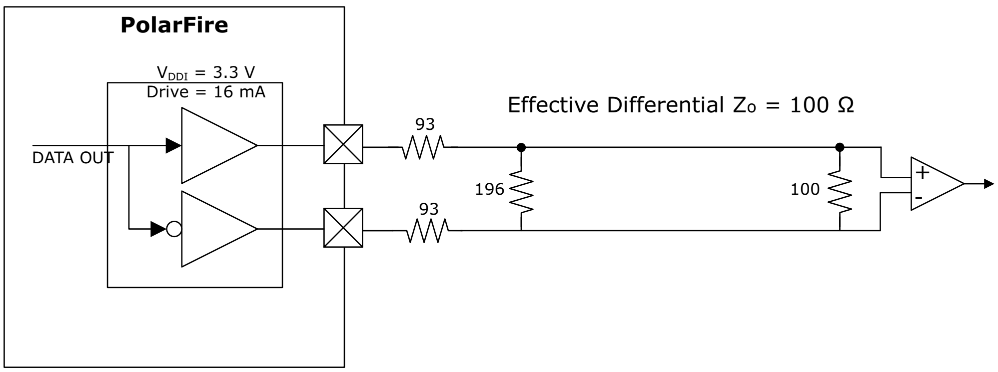
Figure 9 • MLVDSE System Diagram



3.2.4.4 LVPECL Emulated (LVPECLE33) Output Mode

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

Figure 10 • **LVPECL System Diagram**



3.2.5 Implementing MIPI D-PHY

PolarFire devices support 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. PolarFire supports MIPI D-PHY with MIPI25 and MIPIE25 IO types dependent on the interface.

3.2.5.1 MIPI D-PHY Receive Interface

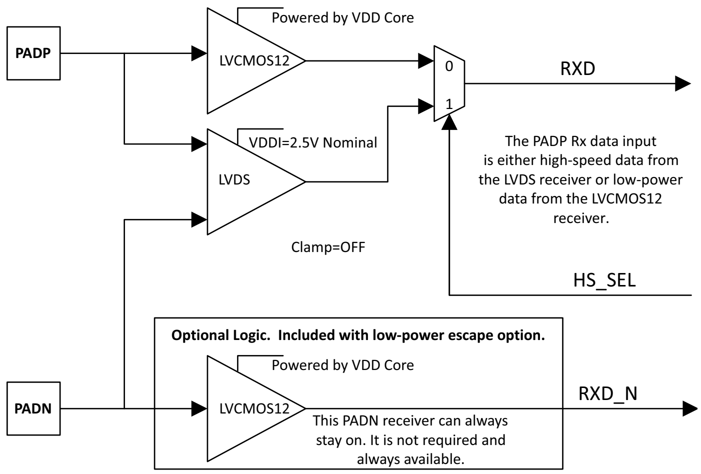
PolarFire FPGA GPIO supports unidirectional MIPI D-PHY I/O in the receive direction, as shown in the following illustration. The MIPI D-PHY receiver supports high-speed (HS) signaling mode for data traffic and low-power (LP) signaling mode used for control. Each HS lane using MIPI25 is terminated and 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 mode. These modes are selectable via an enable (HS_SEL) from the IOD component when MIPI low-power escape support is selected in the PolarFire IOD Generic Receive Interfaces configurator (See Figure 39, page 61).

When the MIPI25 low-power escape support is used, the IO 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 100 Ω differential termination resistor when HS_SEL=1. This is generated by Libero 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 100 Ω 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 both the PADP and PADN pins. This MIPI interface is implemented by configuring the PADP as a MIPI receiver, PADN pin and LVCMOS12 receiver. FPGA hosted logic is required to control this feature.

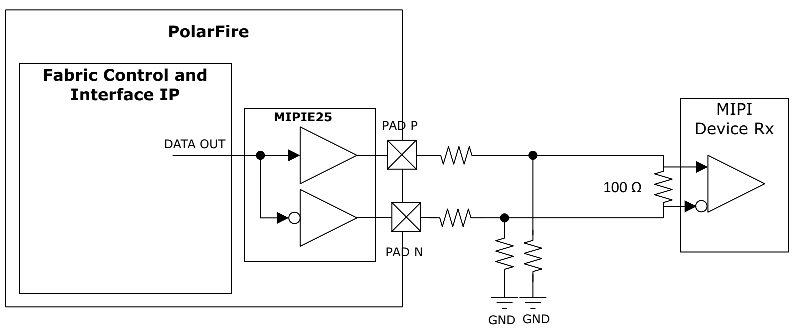
Figure 11 • **MIPI D-PHY Receiver**



3.2.5.2 MIPI D-PHY Transmitting Interface (High-speed Only)

PolarFire FPGA 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.

Figure 12 • **MIPI D-PHY Transmit Interface**

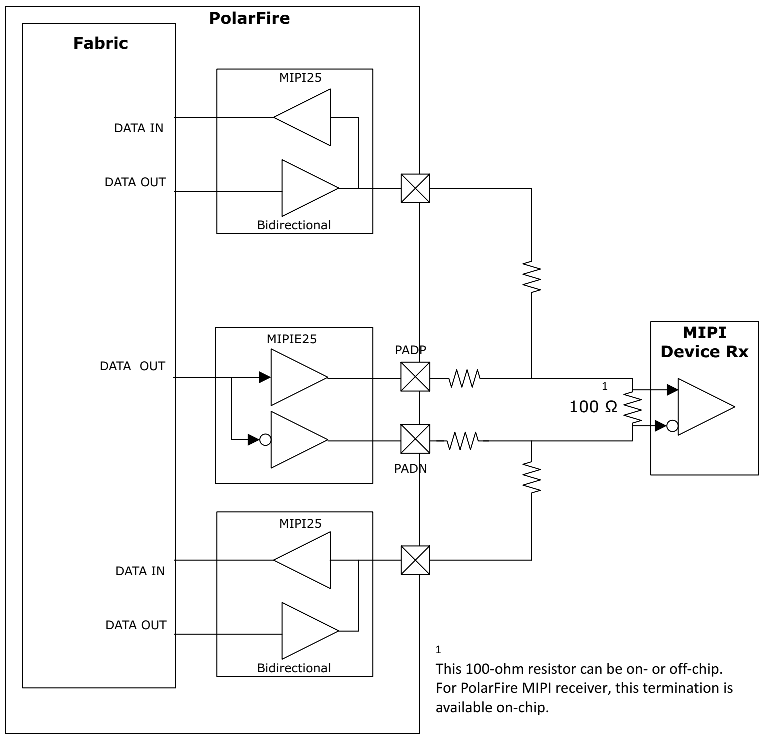


Note: Resistor value vary based on optimal performance. See *UG0726: PolarFire FPGA Board Design User Guide* for resistor specifications.

3.2.5.3 MIPI D-PHY Transmit Interface (High-speed Only) with Bidirectional Low-Power Mode

PolarFire FPGA GPIO also supports a bidirectional MIPI D-PHY lane with external resistors, as shown in the following illustration. Microsemi 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 PolarFire FPGA Generic I/O Interfaces, page 45 for more information.

Figure 13 • High-Speed Transmit with Bidirectional



Note: See *UG0726: PolarFire FPGA Board Design User Guide* for resistor specifications.

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

3.2.6 I/O States During Various Operational Modes

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 users can design their boards accordingly.

3.2.6.1 Power-Up and Initialization

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

Table 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 I/O settings. Data and output enable signals are based on user settings.

For more information about I/O states, see *UG0714: PolarFire FPGA Programming User Guide*.

3.2.6.2 Device Programming Modes

The following table lists PolarFire FPGA user I/O states during various programming modes. For more information about programming modes, see *UG0714: PolarFire FPGA Programming User Guide*.

Table 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

3.2.7 Cold Sparing and Hot Socketing

This section describes cold sparing and hot socketing capabilities of PolarFire FPGA user I/Os. For more information about cold sparing and hot socketing, see *UG0726: PolarFire FPGA Board Design User Guide*.

3.2.7.1 Cold Sparing

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 by both GPIO and HSIO—any I/O of an unpowered PolarFire FPGA can be safely driven with very minimal leakage current. When the device is powered off, both V_{DD} and the V_{DDI} are clamped to ground, preventing these supplies from powering up when a voltage is applied to the inputs. It is a good design practice to not rely on the outputs of an unpowered or partially powered PolarFire device to drive other components in the system. Note that both GPIO and HSIO support the cold sparing feature.

3.2.7.1.1 Hot Socketing

Hot socketing allows a voltage to be applied to the inputs of PolarFire devices before power is present on the V_{DDI} pins. PolarFire FPGA GPIO supports hot socketing, but HSIO does not support hot socketing.

When the 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 $V_{REF,}$ the amount of current flowing into or out should be minimized for the GPIO pin so that the external V_{REF} signal is not affected.

3.2.8 IO Glitches

IO 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 IO interfaces are discussed in *UG0726: PolarFire FPGA Board Design User Guide*.

3.2.9 IO Calibration

HSIO and GPIO have a built in I/O calibration feature per bank excluding bank 3. The IO 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 IO drivers. The calibration is initially completed at power up. It is initiated by power-on detectors on VDDI and VDDAUX power supplies. For more information about calibration requirements for proper start-up and initialization, see *UG0725: PolarFire FPGA Device 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 IO's in order 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 IO's within the bank. The calibration values are used for PVT compensation. GPIO and HSIO uses 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.

GPIO initially powers-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 utilizing the ODT control capabilities in the IO (see On-Die Termination (ODT), page 17). Training IP (TIP) normally associated with high-speed DDR interfaces can be used to disable the IO termination until calibration is complete. For untrained termination interfaces, the ODT_DYN interface can be used to disable this pre-calibrated termination.

3.2.10 Dynamic ODT or Fail-Safe LVDS

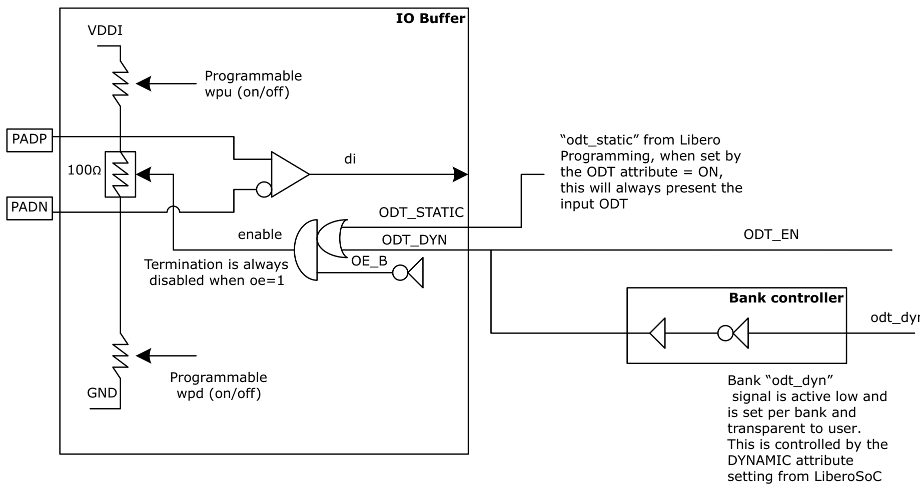
PolarFire can support an internal LVDS fail-safe solution. This configuration uses a combination of the following device features:

- 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.

As per bank, 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 the ODT_EN to switch in or 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 should be present for the LVDS receiver. During fail-safe, drive ODT_EN = '0' to disable ODT.

Figure 14 • **Dynamic ODT used for Failsafe LVDS**



I/O configurators that use LVDS input have the "Enable ODT_EN pin for LVDS Failsafe" option. In the I/O Editor, the ODT attribute for differential I/O's has the "Dynamic" option for differential I/Os.

The set_io PDC command supports the "-dynamic" attribute for differential I/Os.

4 IOD Features and User Modes

Each PolarFire FPGA 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 (Figure 15, page 34). 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 PolarFire 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.

4.1 IOD Block Features

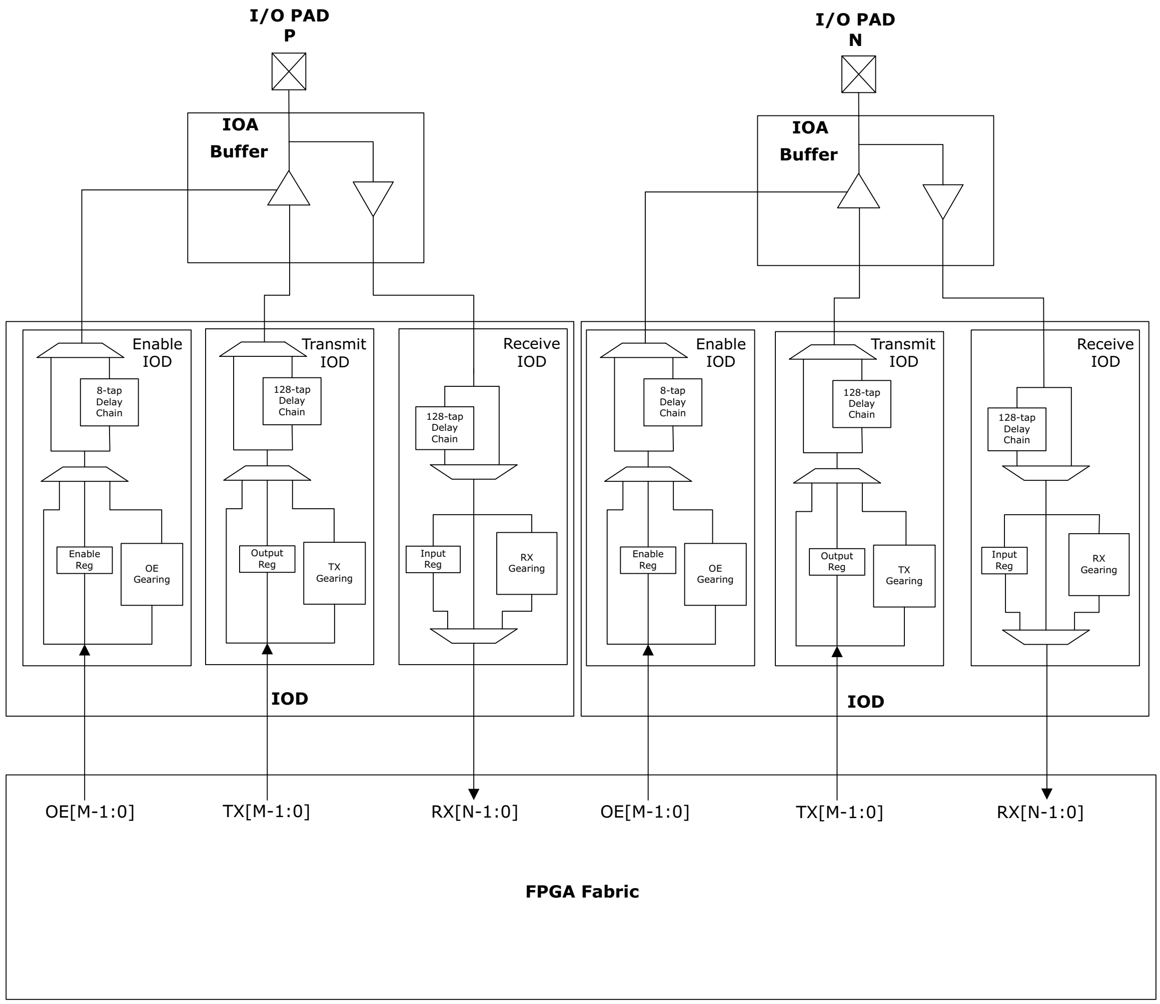
- Programmable input and/or output delay chain
- I/O register for data-in, data-out, and output enable signals
- Up to 1:10 input deserialization (input digital)
- Up to 10:1 output serialization (output digital)
- Support for DDR and SDR interfaces
- Word alignment with a slip control
- High-speed and 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

4.2 IOD Block Overview

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 15, page 34 shows an overview of the IOD block. Various I/O features are set mainly by the protocol configurator or the Libero configurator within Libero SoC PolarFire. However, some of the I/O features such as I/O register and programmable delay can be controlled automatically or manually by Libero SoC PolarFire.

The following illustration shows an overview of the IOD block.

Figure 15 • **IOD Configured for I/O Registers**



Note: The values of M and N depend on the digital ratio.

4.2.1 Programmable I/O Delay

The IOD block includes process, voltage, programmable delay chains for both input and output data paths. These programmable delay chains on input data paths allow tap delay of approximately 25 ps. 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 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 of 25 ps. The enable path also includes a 8-tap programmable delay chain. The programmable delay chain can be set statically by using the I/O attribute editor or by using a PDC command in Libero SoC PolarFire. The value per tap delay is 30 ps typical (-1). For information about delays, see *DS0141: PolarFire 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 via 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 PolarFire I/O interfaces, see PolarFire FPGA Generic I/O Interfaces, page 45 for more information. Static delay values can be controlled by PDC command constraint via IOEditor 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 via IOEditor or manual constraint file input. In the PDC constraint file, IN_DELAY allows settings from OFF, 1 - 128.

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

4.2.1.1 Static Timing Analysis

Static delays are automatically prescribed by the IOD configurator. The values that are added based on the IOD configuration can be found in a 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. The user can adjust the delay values by adding or subtracting from the initial value applied in the Libero configurations. The per tap incremental delay value is found in the *DS0141: PolarFire FPGA Datasheet*.

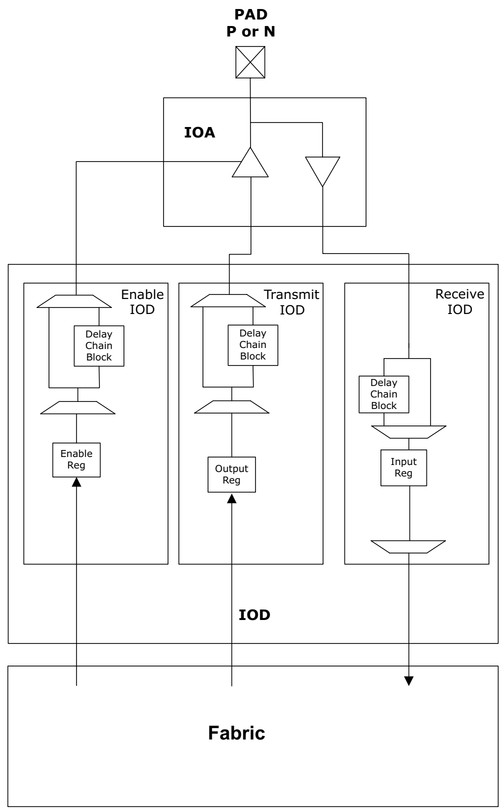
Figure 16 • **IOD Input Delay Example**

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]

4.2.2 I/O Registers

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. Figure 17, page 36 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 & route software.

Figure 17 • **I/O Registers in IOD**



The I/O register is used for:

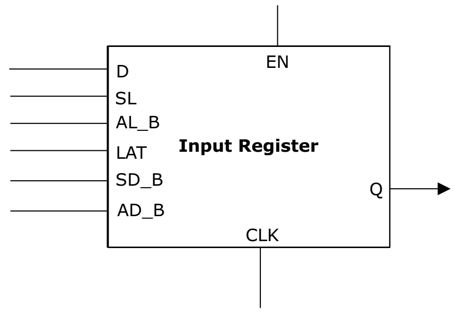
- Better I/O interface performance, as the registers are placed close to the I/O pads.
- Synchronizing the transmit and receive bus signals. For example, the I/O registers ensure that all the bits of the bus are synchronized to the clock signal when they are transmitted or received.

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

4.2.2.1 Input Register

The following illustration shows the input register.

Figure 18 • **Input Register**



The following table lists the input register pins and descriptions.

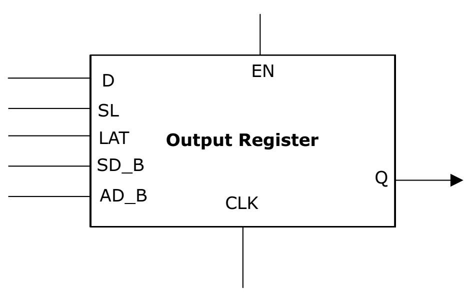
Table 20 • **I/O Input Register Ports**

Ports	Types	Descriptions
CLK	Input	Clock input
D	Input	Data input
Q	Output	Data output
EN	Input	Clock enable (active high)
SL	Input	Synchronous load (active high)
AL_B	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)

4.2.2.2 Output Register

The following illustration shows the output register.

Figure 19 • **Output Register**



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

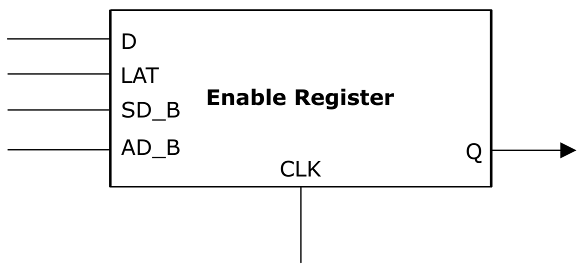
Table 21 • **I/O Output Register Ports**

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

4.2.2.3 Enable Register

The following illustration shows enable register.

Figure 20 • **Enable Register**



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

Table 22 • **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)

4.2.2.4 IO Register Combining

IO register combining is supported on enable, input, and output of any IO. 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 {<portname>} \
[-in_reg yes|no] \
[-out_reg yes|no] \
[-en_reg yes|no]

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: specifes whether the output register is combined into the port

-en_reg: specifes whether the enable register is combined into the port <portname>.

Note: Valid values are “yes” or “no”.

IO register combining is only permitted with one FF with an IO. The FF needs to be connected to the IO 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 discussed above for more tightly controlling the use of I/O register combining.

Note: This feature is off by default. Users must turn it on to enable combining.

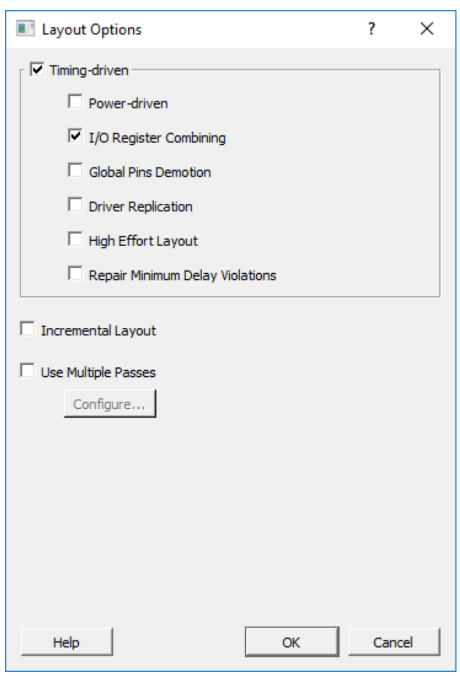
Every I/O has several embedded registers that you can use for faster clock-to-out timing, and external hold and setup. When combining these registers at the I/O buffer, some design rules must be met.

This feature is supported by all I/O standards.

Following are the rules to combining the IO 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 21 • **I/O Register Combining from Place and Route Layout Options**

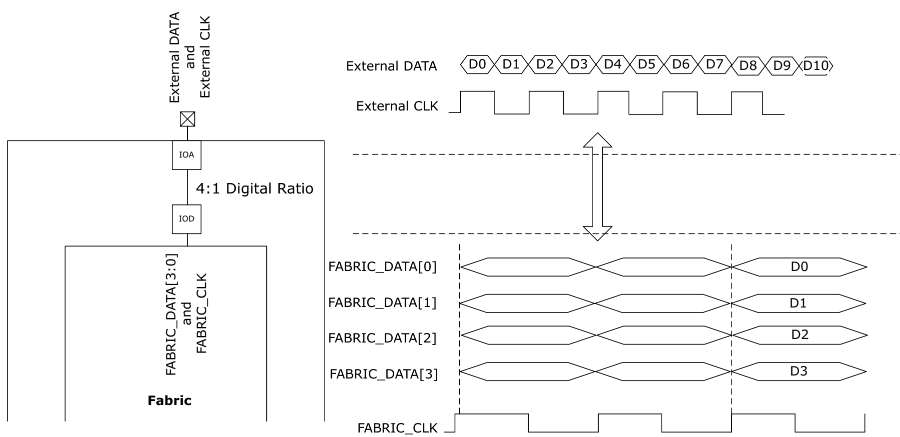


4.2.3 I/O Gearing

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 PolarFire automatically configures these gearboxes based on the application settings. Generic IOD interfaces provide a complete solution from the IO pins to the fabric. Generic IOD is supported by construction using Libero configurators and limited to the defined list of use cases. See PolarFire FPGA Generic I/O Interfaces, page 45 for available support.

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

Figure 22 • **I/O Digital**



4.2.4 I/O FIFO

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 PolarFire and Microsemi memory controller cores configure the I/O FIFO based on the application settings.

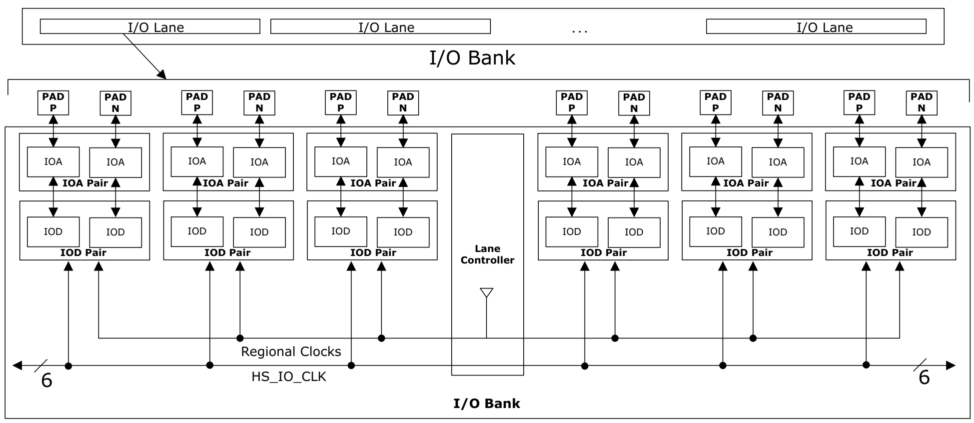
4.3 PolarFire FPGA I/O Lanes

To support memory interfaces, PolarFire FPGA 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 *UG0684: PolarFire FPGA Clocking Resources User Guide* for more information.

The PolarFire FPGA 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 PolarFire FPGA 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 IO interfaces, which requires high-speed and low-skew clocking.

The following illustration shows the PolarFire FPGA I/O lanes diagram.

Figure 23 • **PolarFire FPGA 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 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.

4.3.1 Lane Controller

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 IO 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 PolarFire FPGA Generic I/O Interfaces, page 45.

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, page 67 for more information. A divided-down version of the recovered clock is provided to the fabric (DIVCLK).

4.3.2 I/O Lanes in Each Bank

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

Table 23 • **I/O Lanes in Each Bank**

		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**	
Devices	**Packages**	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
MPF100	FCSG536	60	5	0	0	60	5	13	96	8	0	0	68	5	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

Note: There is a lane migration restriction for IOCDR and any IOD generic Rx interfaces using regional clock. This implies a design cannot migrate from the MPF300 to the other devices if the impacted lanes are being used.

The lanes that cannot migrate from the MPF300 include (as documented in the associated Package Pin Assignment Table (PPAT)):

MPF100, MPF200: DDR_S_3 (South bank Lane 3)
MPF500: DDR_S_6 (South bank Lane 6), DDR_N_9 (North bank Lane 9)

4.4 I/O Clock Networks

Each PolarFire FPGA I/O contains a fabric, a high-speed IO 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. For more information about global clock network, see *UG0684: PolarFire FPGA Clocking Resources User Guide*.

4.4.1 Global Clock Resource

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 PolarFire 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 *UG0684: PolarFire FPGA Clocking Resources User Guide*.

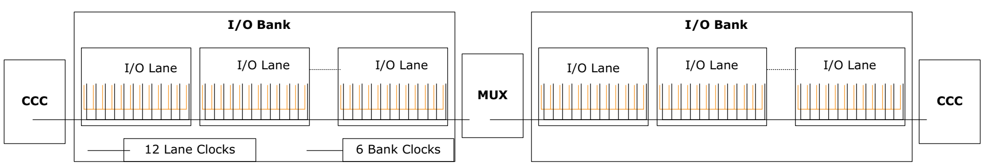
4.4.2 Regional Clock Networks

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 offer one regional clock buffer per I/O lane on the northern, southern, and western edges. Note that 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 *UG0684: PolarFire FPGA Clocking Resources User Guide*.

4.4.3 Lane Clock Resources

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 PolarFire automatically uses the lane clock resource based on the I/O configuration.

Figure 24 • **Distribution of the Lane Clock**



4.4.4 High-Speed I/O Bank Clock Resource (HS_IO_CLK)

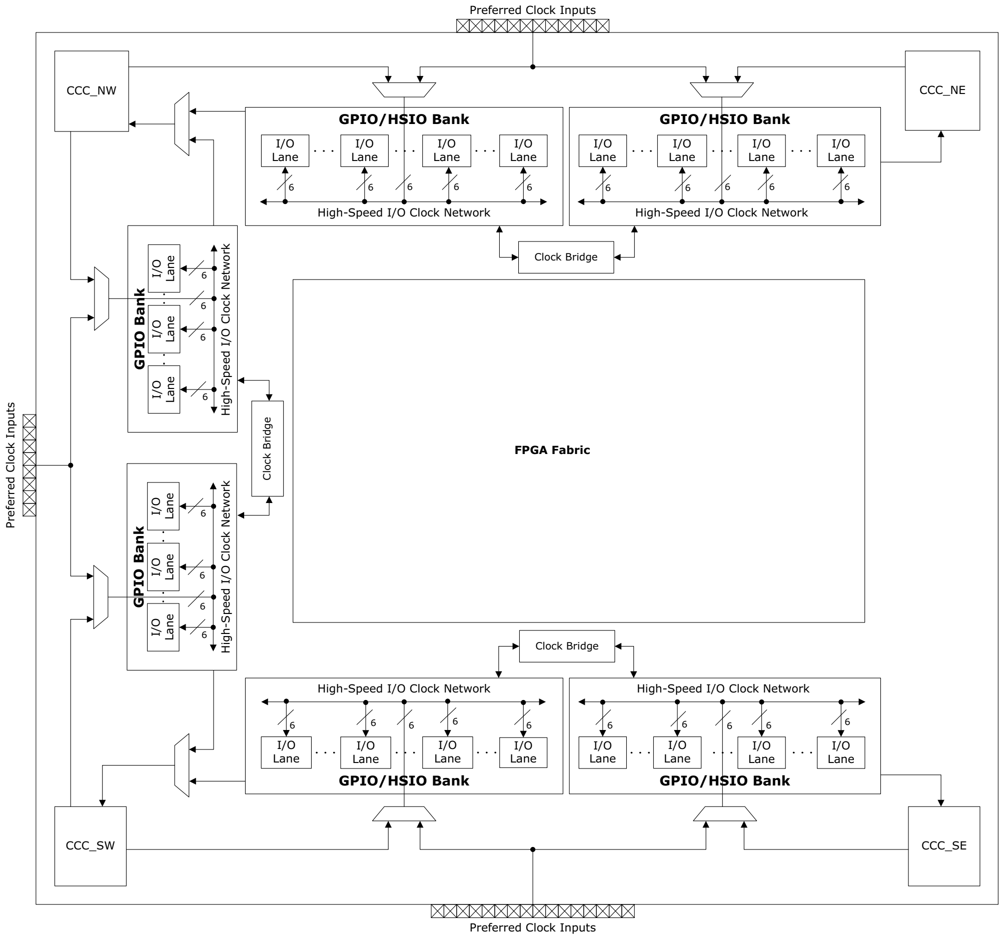
High-speed I/O bank clock networks are integrated into I/O banks and distribute clocks along the entire I/O bank with low-skew. They are used to clock data in and out of the I/O logic when implementing the high-speed interfaces. The high-speed I/O clock networks are located on the east corner of the 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 Interfaces, page 52). Some HS_IO_CLK bridging topologies are not fully migrative between the MPF devices due to architectural limitations.

There are noted differences in the clocking architecture between devices. One notable difference is in the bank2 structure between MPF100T and MPF300T. In MPF300T, all I/Os in bank2 can be directly driven by south west (SW) PLL. Whereas, in the MPF100, I/Os are split into two groups: Lane0/1/2 are driven by SW PLL and other lanes are driven by south east (SE) PLL.

For MPF100T and MPF200T devices, the placer will error as a bridging error for clock and data if the pin out originates from the MPF300T. Users must always target potential devices in Libero before committing the final pin out to the PCB to assure no clocking issues exist when migrating.

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 to the user as they are setup by Libero based on configuration.

Figure 25 • Distribution of the HS_IO_CLK



4.4.5 Bit Slip

BITSLIP is used to align the de-serialized input data burst into the fabric (this is 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 RX_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 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 RX_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 RX_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 Generic IO Interface** configurators allow optional use of the BITSLIP function.

4.5 PolarFire FPGA Generic I/O Interfaces

Many pre-defined interfaces are available from the Libero IO configurator. The user selects an interface from the list that closest matches their needs. See Generic IOD Interface Implementation, page 58 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 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, page 81.

When building generic high-speed DDR interfaces in PolarFire devices, it is required to follow the Interface Rules described for each type of interface. The PolarFire FPGA 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 each 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. Users must also be aware of performance specifications for IOA types when building their particular IO interface. Users must select an IO type that matches the desired maximum performance rate by referencing the *DS0141: PolarFire FPGA Datasheet*.

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

4.5.1 RX DDR Interfaces

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). The IO makes direct use of clock topologies around the perimeter of the IO ring to build synchronous IO 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 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 inputs 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 generated data and clock delay tunings. Using constraints, the user 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 where-as 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. See *UG0684: PolarFire FPGA Clocking Resources User Guide* for more information about clocking topologies of the PolarFire devices.

The PolarFire Generic IO 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 Aligned)

The PolarFire family includes the following generic RX DDR interface types.

Table 24 • **PolarFire Device Support for Generic RX DDR interfaces[1], [2]**

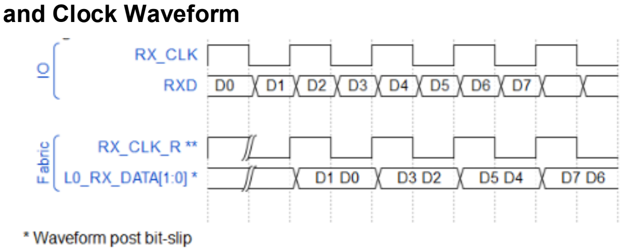
Name	Clock Type	Gearing Ratio	Description
RX_DDR_G_A	Continuous	1	Rx DDR w/Global aligned
RX_DDR_R_A	Continuous	1	Rx DDR w/Regional clock aligned
RX_DDR_G_C	Continuous	1	Rx DDR w/Global centered
RX_DDR_R_C	Continuous	1	Rx DDR w/Regional clock centered
RX_DDRX_B_G_A	Continuous	2, 3.5, 4, 5	Rx DDR geared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

1. For more information about maximum operating frequency, see *DS0141: PolarFire FPGA Datasheet*.
2. Regional clock interfaces uses the generated HS_IO_CLK to capture the data and then transferred to the regional clock within the FPGA fabric.

4.5.2 RX_DDR_G_A/ RX_DDR_R_A—Aligned Interfaces with Static Delays

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 of the PolarFire device 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 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 *DS0141: PolarFire FPGA Datasheet*. There are two interface configurations based on clock source topology being either global or lane-based.

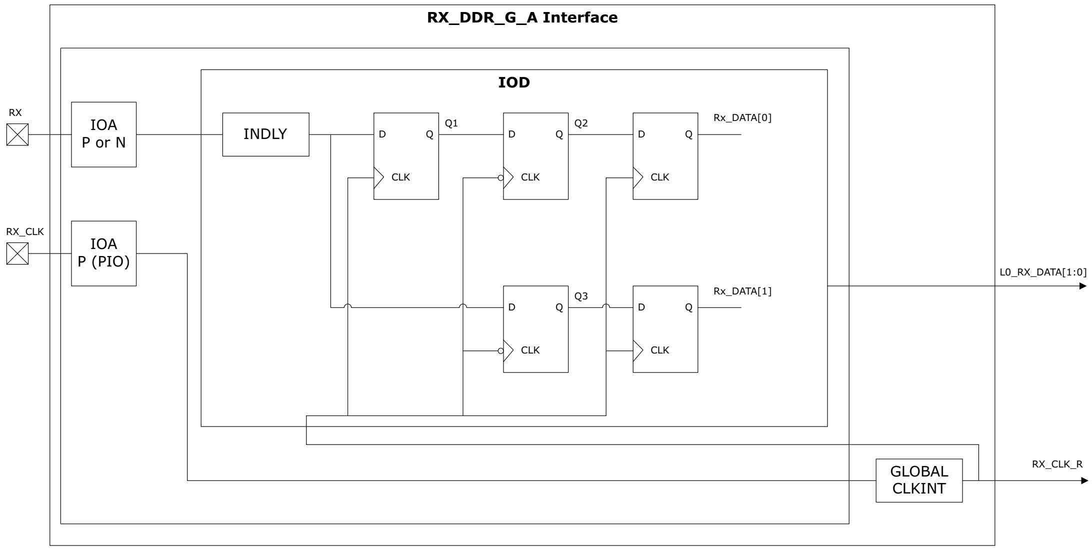
Figure 26 • **Aligned Data and Clock Waveform**



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 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.

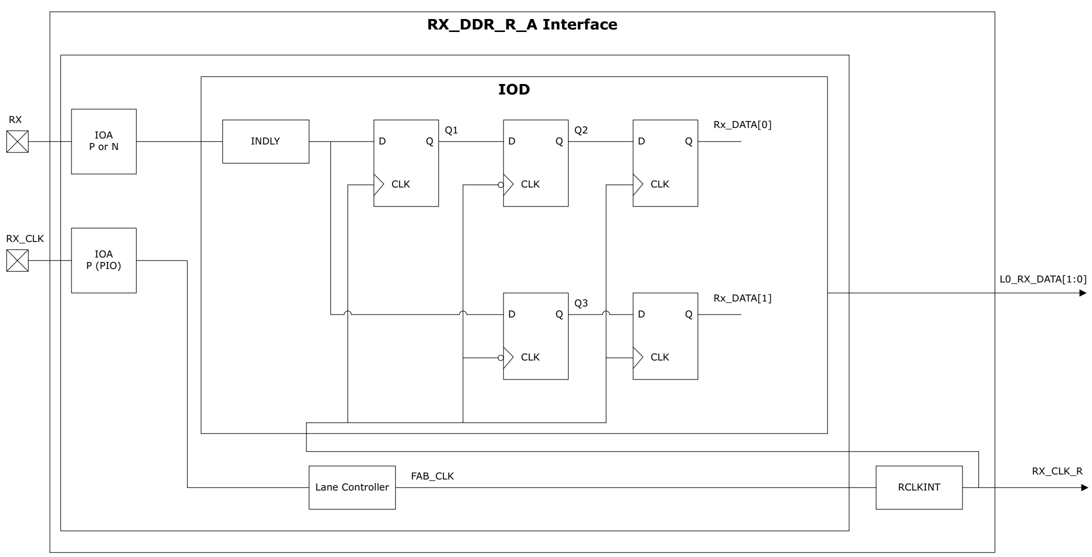
Figure 27 • **RX_DDRX1 Aligned Interface Using Global Clock**



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 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 28 • **RX_DDRX1 Aligned Interface Using Regional Clock**



4.5.2.1 Interface Ports

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

Table 25 • **RX_DDR Aligned Interface Mode Ports[1]**

Port	I/O	Description
RX	Input	Input DDR data. Supports up to 32-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 32 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, page 58.

4.5.2.2 Interface Selection Rules

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.

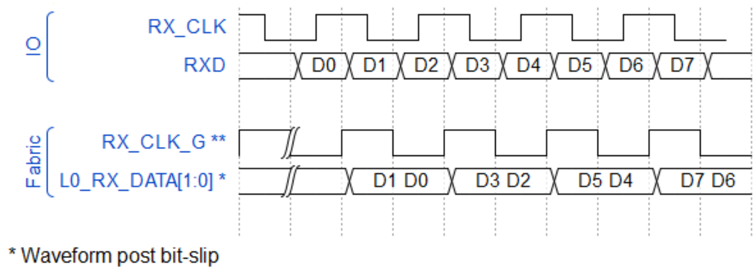
The following rules apply when assigning a pin to the RX_DDR_R_A aligned interface:

- 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.

4.5.3 RX_DDR_G_C and RX_DDR_R_C—Centered Interfaces with Static Delays

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 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.

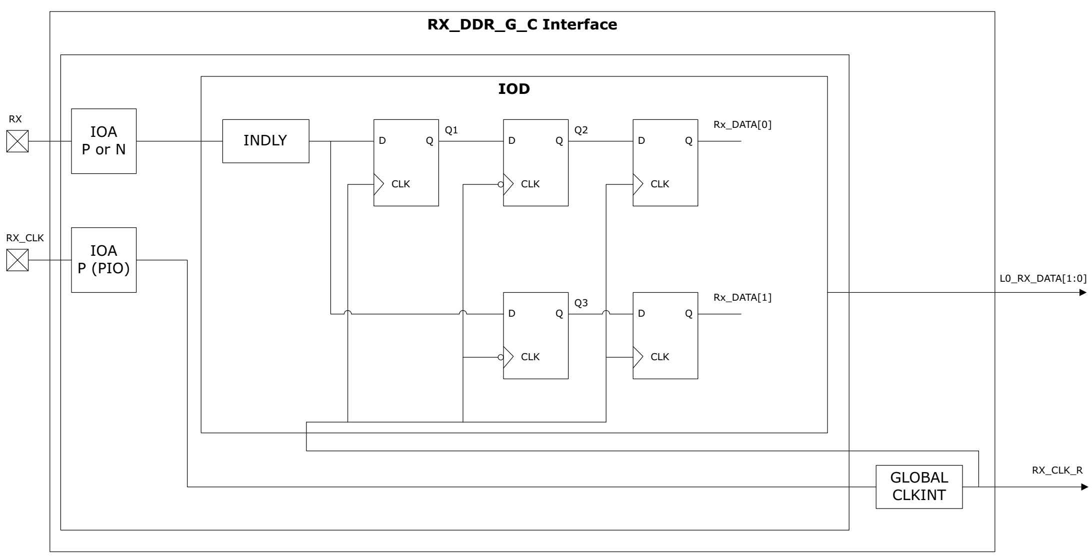
Figure 29 • **Centered Data and Clock Waveform**



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.

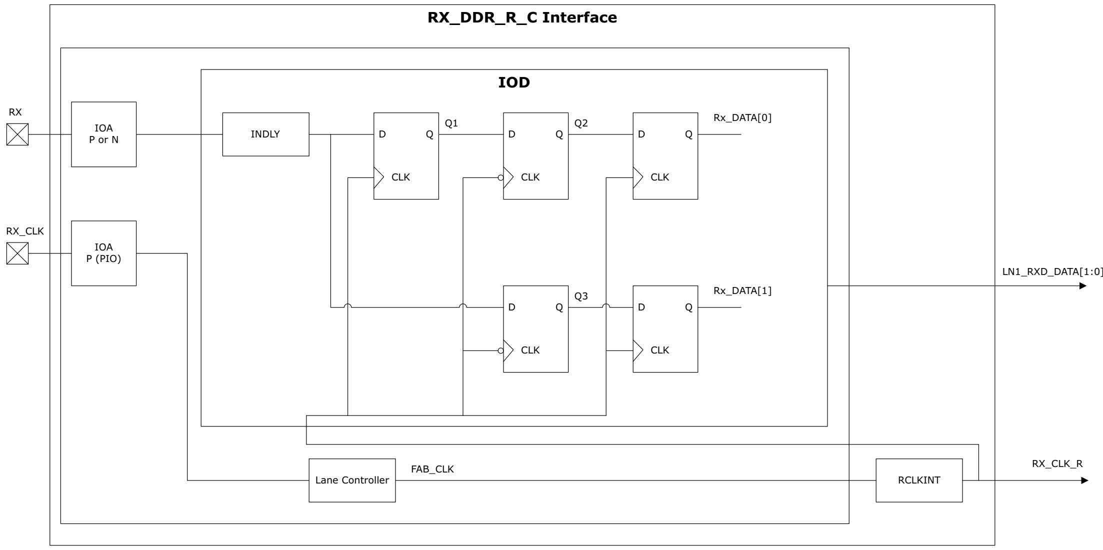
Figure 30 • **RX_DDRX1 Centered Interface Using Global Clock**



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 31 • **RX_DDRX1 Centered Interface Using Regional Clock**



4.5.3.1 Interface Ports

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

Table 26 • **RX_DDR Centered Interface Mode Ports[1]**

Port	I/O	Description
RX	Input	Input DDR data. Supports up to 32-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 32 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, page 58.

4.5.3.2 Interface Selection Rules

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.

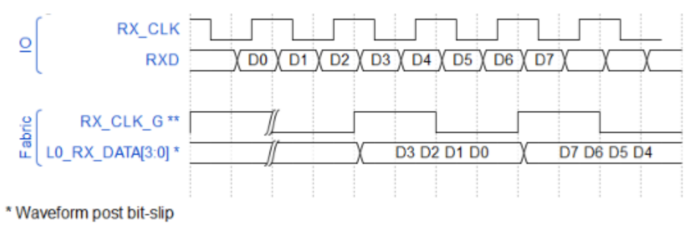
- RX and RX_CLK I/Os must be placed in the same I/O lane.
- RX_CLK input must be placed in the P side I/O with the DQS function in the lane.
- RX and RX_CLK I/Os must be placed in the same bank (exception on device with bank7, I/Os can be either in both bank0 and bank7).
- One IOD per data I/Os.
- One IOA per data and clock I/Os.

4.5.4 RX_DDRX_B_G_C and RX_DDRX_B_G_A/RX_DDRX_B_R_A Interfaces with Static Delays

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 PolarFire device. Similar to the RX_DDRX1 interfaces, the Libero 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.

Figure 32 • **Rx_geared Waveform**



4.5.4.1 Interface Ports

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

Table 27 • **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 32-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 IO Gearing state machine to detect the phase difference between fabric clock and clock coming out of PLL.

Table 27 • **RX_DDR_B_C and RX_DDR_B_A Interface Mode Ports[1]**

Port	I/O	Description
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 32 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, page 58.

4.5.4.2 Interface Selection Rules

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:

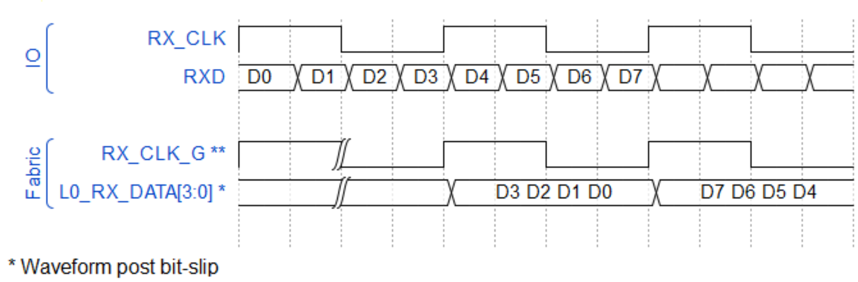
- Interface use two ICB_CLKDIVDELAY and three HS_IO_CLK.
- RX_CLK input must be placed in an I/O with the CLKIN_z_w function in the same bank as other I/Os.
- RX and RX_CLK I/Os must be placed in the same bank (exception on device with bank7, I/Os can be either in both bank0 and bank7).
- One IOD per 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.

4.5.5 RX_DDR Fractional Aligned Interfaces

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 into the clock path with a multiplier of 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.

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

Figure 33 • **Fractional Aligned Data and Clock Waveform**



4.5.5.1 Interface Ports

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

Table 28 • **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.
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 powerdown.
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.
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 32 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.

4.5.5.2 Interface Selection Rules

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. See *UG0684: PolarFire FPGA Clocking Resources User Guide* for information about the available pins for each device. 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.

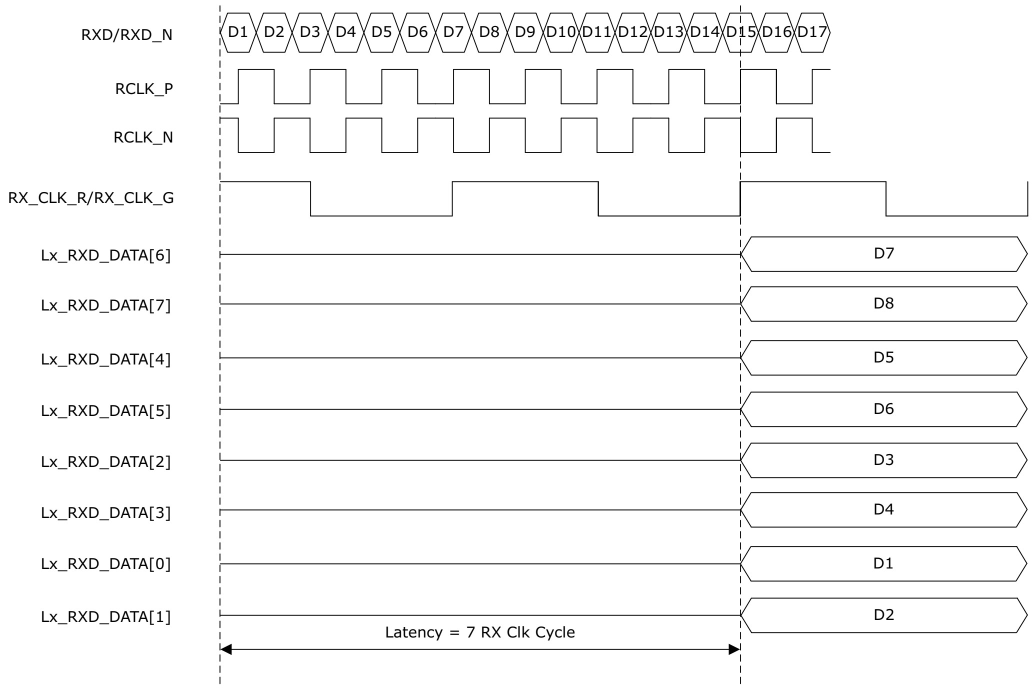
4.5.6 RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN

The RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN interface is used to capture differential DDR data using dynamic control. The clock and data relationship can be adjusted dynamically when the device receives the differential DDR data. The RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN interface is used for the maximum data rate of 1600 Mbps and uses the digital ratio of 2, 3.5, 4, and 5.

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

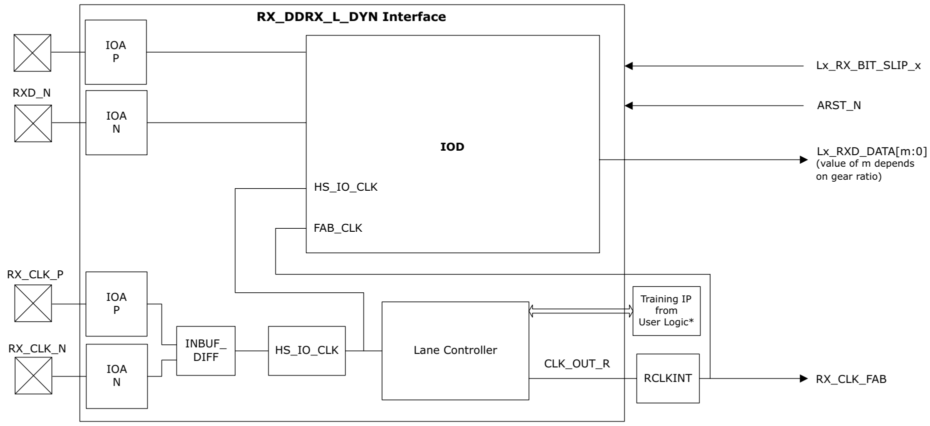
The following illustration shows the signal waveform of RX_DDRX_B_G_DYN/ RX_DDRX_B_R_DYN interface when slip input is not used.

Figure 34 • **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 35 • **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, page 63.

4.5.6.1 Interface Ports

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

Table 29 • **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
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_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.

1. For more information, see Dynamic Delay Control, page 63.

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 one of two ways:

- 3.5 Digital Mode—slips 2 bits at a time
- 2, 4, and 5 Digital Modes—slip 1 bit at a time

4.5.6.2 Interface Selection Rules

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

- Interface use 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.

4.5.7 TX DDR Interfaces

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

Table 30 • **TX DDR Interfaces**

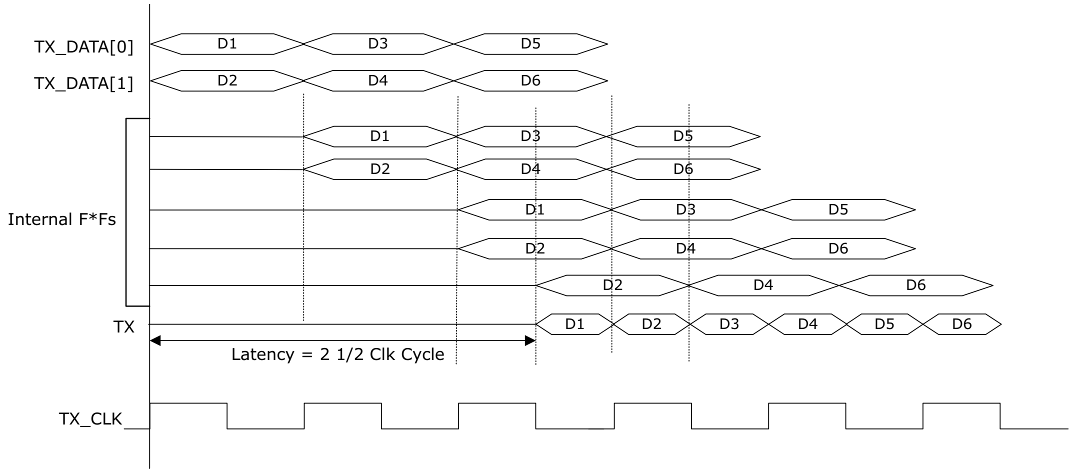
Interface Name	Topology	Gearing Ratio	Clock-to-Data Condition
TX_DDR_G/B_A	TX DDR	1	From a global clock source, aligned clock, and data

4.5.7.1 TX_DDR_G/B_A

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 via the TX pin.

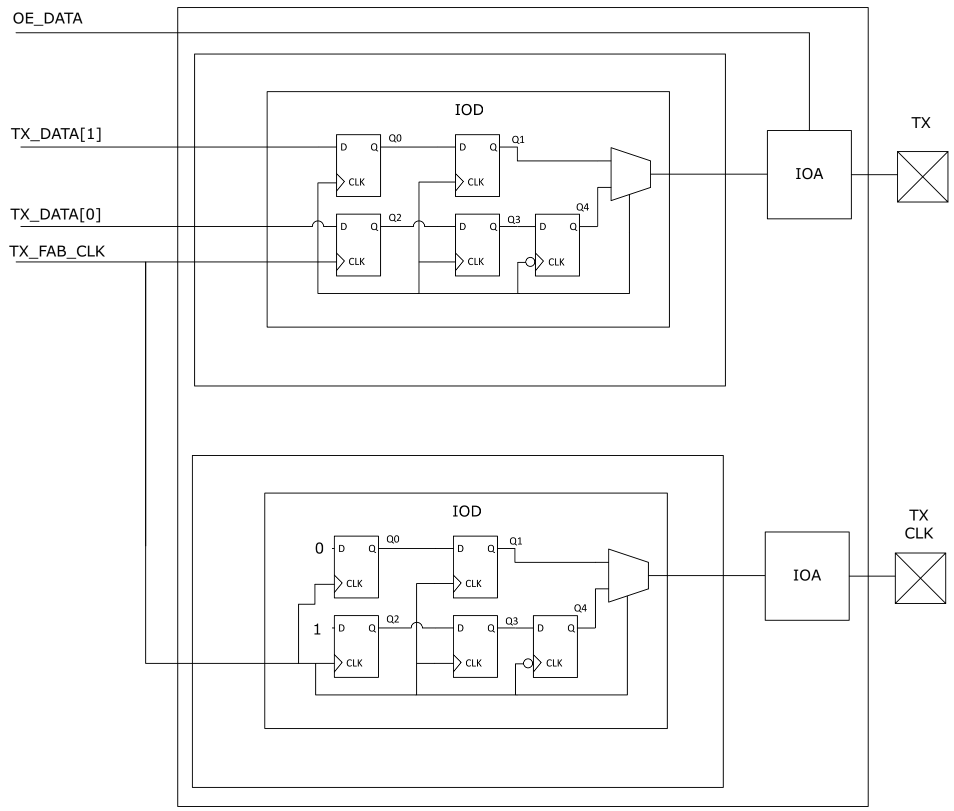
The following figure shows the TX_DDR_G_A interface signal waveform.

Figure 36 • **TX_DDR_G_A Interface Signal Waveform—TXDDRX1**



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

Figure 37 • **TX_DDR_G/B_A Interface Block Diagram—TX_DDRX1**



4.5.7.2 Interface Ports

The following table lists the TX_DDR_G_A interface mode ports.

Table 31 • TX_DDR_G/B_A Interface Mode Ports[1]

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

1. Other pins are visible when advanced options are used. See Generic IOD Interface Implementation, page 58.

4.5.7.3 Interface Selection Rules

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.

4.6 Latency

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 32 • 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

5 Generic IOD Interface Implementation

The PolarFire IO architecture includes many functional features to support source synchronous IO 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 IO clock (HSIO_CLK) to the global clock of the interface.

The Libero SoC PolarFire 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 built components are correct by construction containing the complete data path from the IO pins to the fabric. Libero SoC PolarFire software includes the following IO interface configurators in the DirectICore 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 software.

1. Determine the type of interface to implement for list of defined interfaces.
2. Use the PolarFire IO Interface configurators to build the interface.
3. Use available pre-sets within the PolarFire IO 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 review the Consolidated IOD Rules spreadsheet for pre-place and route guidance.
6. Confirm timing and clock constraints.
7. Define desired IO standard for single-ended or differential IO.

5.1 Software Primitives

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

5.1.1 Input DELAY

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 software
- DYNAMIC—uses the calibrated codes from DLL of the CCC to maintain correct timing across the system.
- 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, page 63.

The DELAY can be adjusted by the user through Libero. 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 defaults. For more information about Input DELAY, see Programmable I/O Delay, page 34.

5.1.2 Input Register (IREG)

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.

5.1.3 Input FIFO

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.

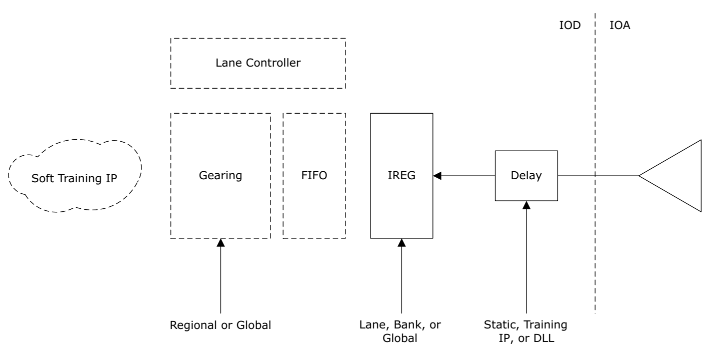
The input FIFO for each IO is composed of two 8 flip-flop deep registers. One register is used for the input data associated with positive edge of clock and the other register is used for the input data associated with the negative edge of the clock. Both registers run on the negative clock edge, by 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, page 40.

5.1.4 Input Gear Box

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 38 • **IOD Modules used within a Generic DDRX I/O Interface**



5.2 IO Interface Configurators

IO 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 user 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 users from crating modules that are not allowed by the architecture.

5.2.1 IOD Templates

Many IOD templates are available for ease entry of interface settings. The interfaces (see PolarFire FPGA Generic I/O Interfaces, page 45) 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.

5.2.2 IOD Generic RX

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

Table 33 • **Receive Interface**

Software Name	Ratio	Clock to Data Relationship	I/O Clock	Fabric Clock	Max Data Rate (Mbps)	Lane Organization	One Lane Max	Dynamic Bit Training
RX_DDR_G_A	1	Aligned	Global	Global	670	✗	✗	✗
RX_DDR_R_A	1	Aligned	Regional	Regional	500	✓	✓	✗
RX_DDR_G_C	1	Centered	Global	Global	670	✗	✗	✗
RX_DDR_R_C	1	Centered	Regional	Regional	500	✓	✓	✗
RX_DDRX_B_G_A	2, 3.5, 4, 5	Aligned	High-speed I/O Clock	Global	740	✓	✗	✗
RX_DDRX_B_R_A	2, 3.5, 4, 5	Aligned	High-speed I/O Clock	Regional	440	✓	✓	✗
RX_DDRX_B_G_C	2, 3.5, 4, 5	Centered	High-speed I/O Clock	Global	740	✓	✗	✗
RX_DDRX_B_R_C	2, 3.5, 4, 5	Centered	High-speed I/O Clock	Regional	440	✓	✓	✗
RX_DDRX_B_G_FA	2, 3.5, 4, 5	Fractional Aligned	High-speed I/O Clock	Global	740	✓	✗	✗
RX_DDRX_B_G_DYN	2, 3.5, 4, 5	Dynamic	High-speed I/O Clock	Global	1000, 1600, 1600, 1600	✓	✗	✓
RX_DDRX_B_R_DYN	2, 3.5, 4, 5	Dynamic	High-speed I/O Clock	Regional	500	✓	✓	✓

The following figure shows the IOD Generic Receive Interfaces.

Figure 39 • **IOD Generic Receive Interfaces—Configuration Tab**

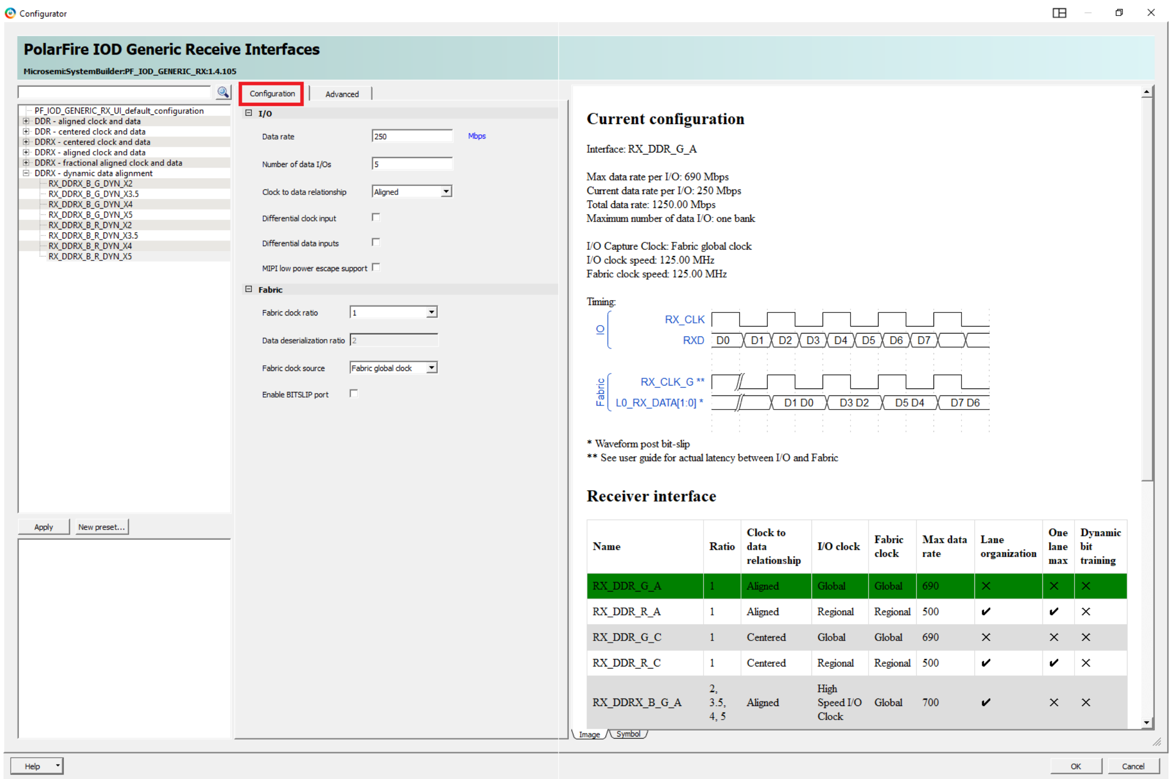


Table 34 • **IOD Generic Receive Interfaces—Configuration Tab**

GUI Option	Selections
Data rate	User Input[1]
Number of data I/Os	User Input – Number of desired RX data inputs (1 to 32)
Clock to data relationship	Aligned, Centered, Dynamic, and Fractional-aligned[2]
Differential clock inputs	Disable (single-ended) and Enabled (differential)
Differential data inputs	Disable (single-ended) and Enabled (differential)
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, page 44 for more information about Bit Slip.

1. See Receiver Interface (right panel) for valid data rates (Figure 39, page 61).

Figure 40 • **IOD Generic Receive Interfaces—Advanced Tab**

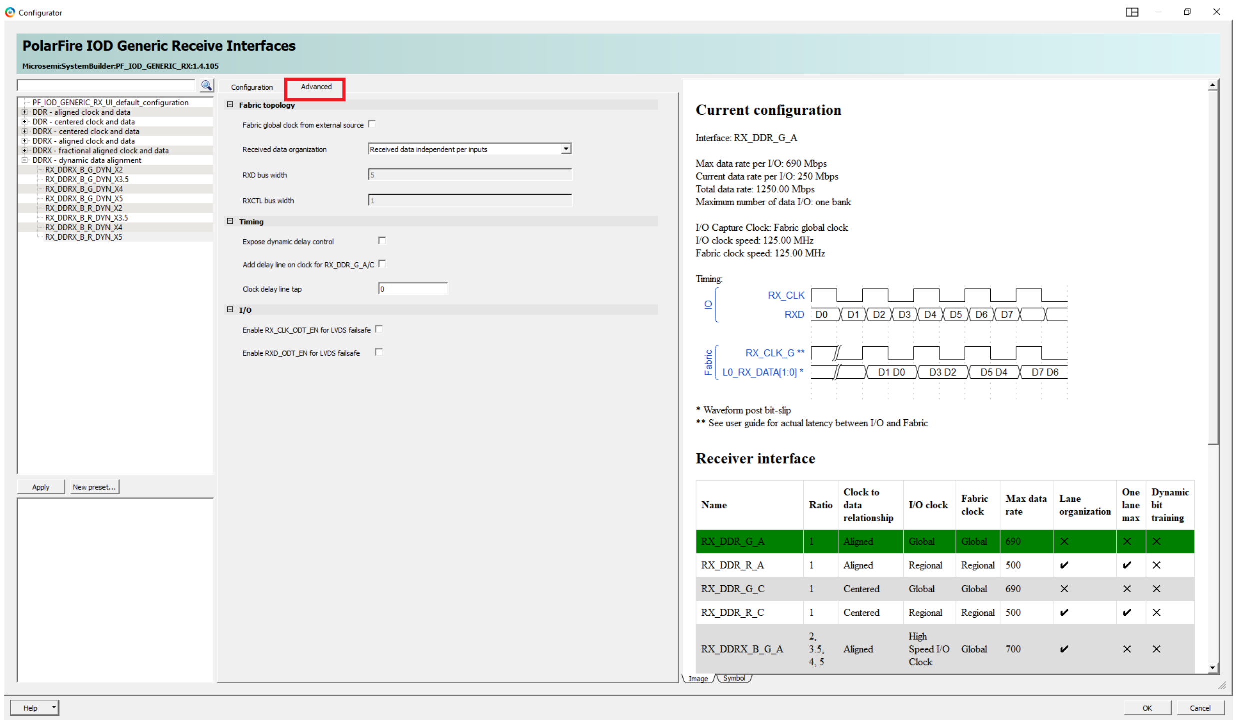


Table 35 • **IOD Generic Receive Interfaces—Advanced Tab**

GUI Option	Selections
Fabric global clock for external source	Enable and Disable
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	
Expose dynamic delay control	Enable and Disable
Add delay line on clock for RX_DDR_G_A/C	Enables static delay chain to be added to clock path
Clock delay line tap	Number of delay taps to be added. See Programmable I/O Delay, page 34 section for information.
Enable RX_CLK_ODT_EN for LVDS failsafe	See Dynamic ODT or Fail-Safe LVDS, page 32 for information.
Enable RXD_ODT_EN for LVDS failsafe	See Dynamic ODT or Fail-Safe LVDS, page 32 for information.

5.2.3 Dynamic Delay Control

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 40, page 62. These ports are automatically exposed when selecting any of the RX_DDRX_DYNAMIC interfaces.

Table 36 • **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
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 setting should 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 should 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 setting 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, the user can determine the relative size of the eye opening.

5.2.4 IOD Generic TX

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

Table 37 • **Transmit Interface**

Software Name	Ratio	Clock to Data Relationship	I/O Clock	Fabric Clock	Max Data Rate (Mbps)	Lane Organization	One Lane Max	Dynamic Bit Training
TX_DDR_G_A	1	Aligned	Global	Global	500	✘	✘	✘
TX_DDR_G_C	1	Centered	Global	Global	500	✘	✘	✘
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 PolarFire IOD Generic Transmit Interfaces configurator.

Figure 41 • **IOD Generic Transmit Interfaces—Configuration Tab**

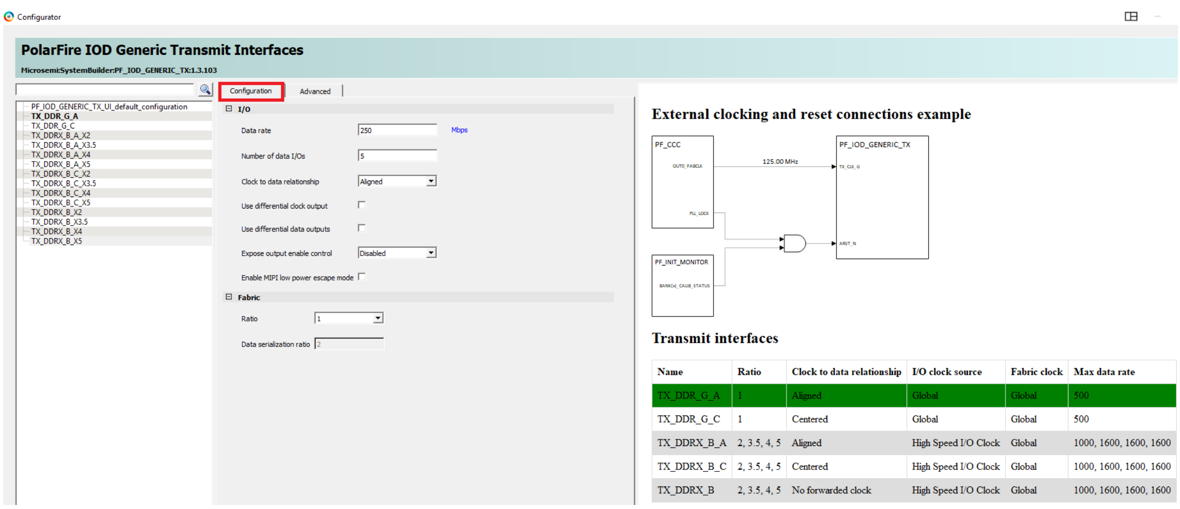


Table 38 • **IOD Generic Transmit Interfaces—Configuration Tab**

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 41, page 64).

Figure 42 • **IOD Generic Transmit Interfaces—Advanced Tab**

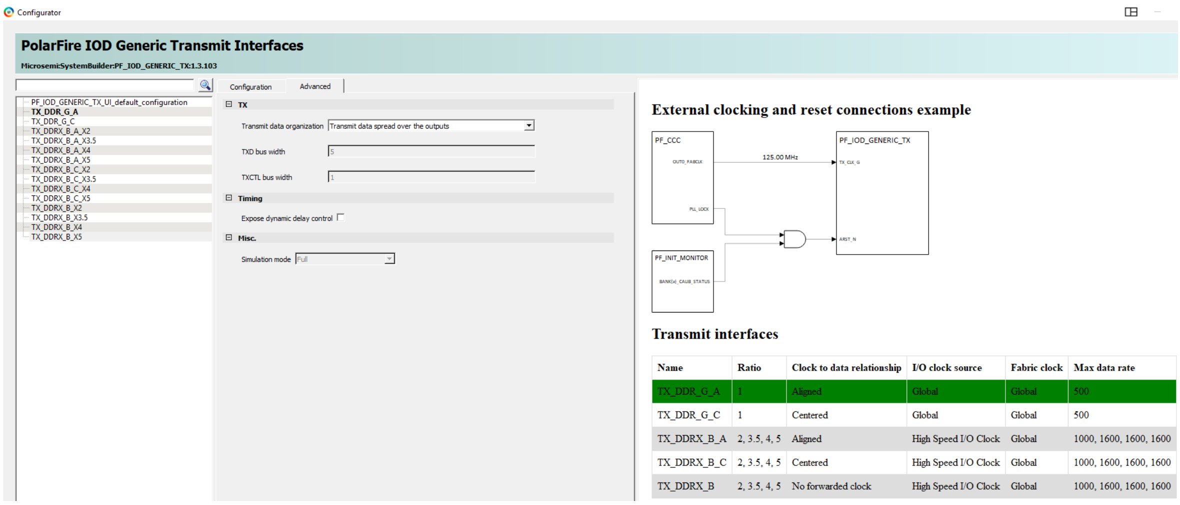


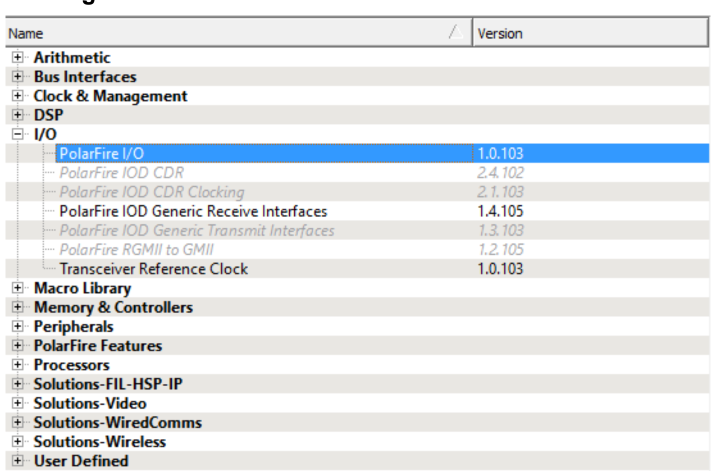
Table 39 • **IOD Generic Transmit Interfaces—Advanced Tab**

GUI Option	Selections
Transmit data organization	Transmit data spread over outputs, Transmit data independent over outputs, Transmit data spread over outputs with data/Control split
TXD bus width	
TXCTL bus width	
Expose dynamic delay control	
Simulation mode	Full

5.3 Basic I/O Configurator

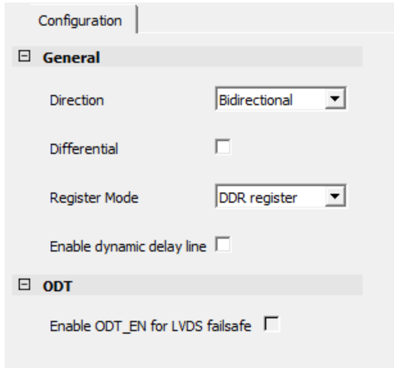
A basic I/O configurator is available in the Libero SoC catalog. It is capable of building simple I/O macros. For information about I/O macros, see *PolarFire Macro Library Guide*.

Figure 43 • **PolarFire IO 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 44 • **IO 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 IO. The configurator does not provide the capability to choose a specific I/O standard. Users must use the IOEditor 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 IO 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. For source-synchronous designs, users should target IOD interfaces, which includes low-skew clock management. See IO Interface Configurators, page 59.

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 data path.

ODT_EN checkbox 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.

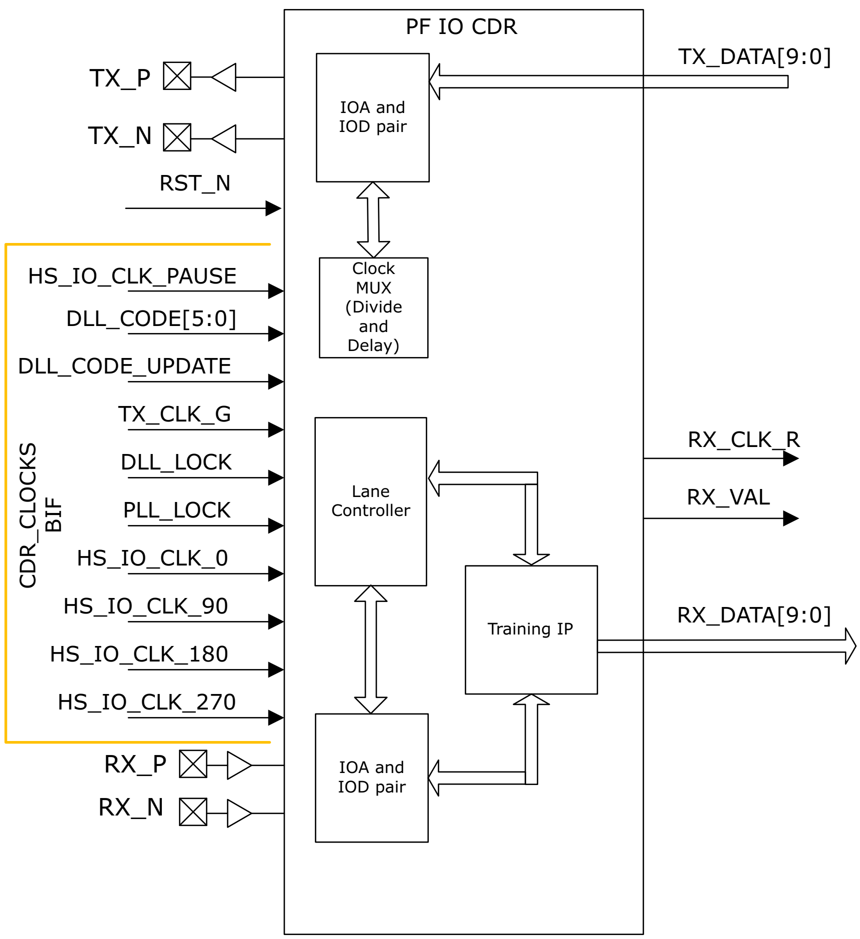
6 Protocol-Specific I/O Interfaces

6.1 PF_IOD_CDR

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 *DG0799: PolarFire FPGA 1G Ethernet Loopback Using IOD CDR Demo Guide*.

The following illustration shows the PF_IOD_CDR transmit and receive interface.

Figure 45 • **PF_IOD_CDR Transmit and Receive Interface Modes**

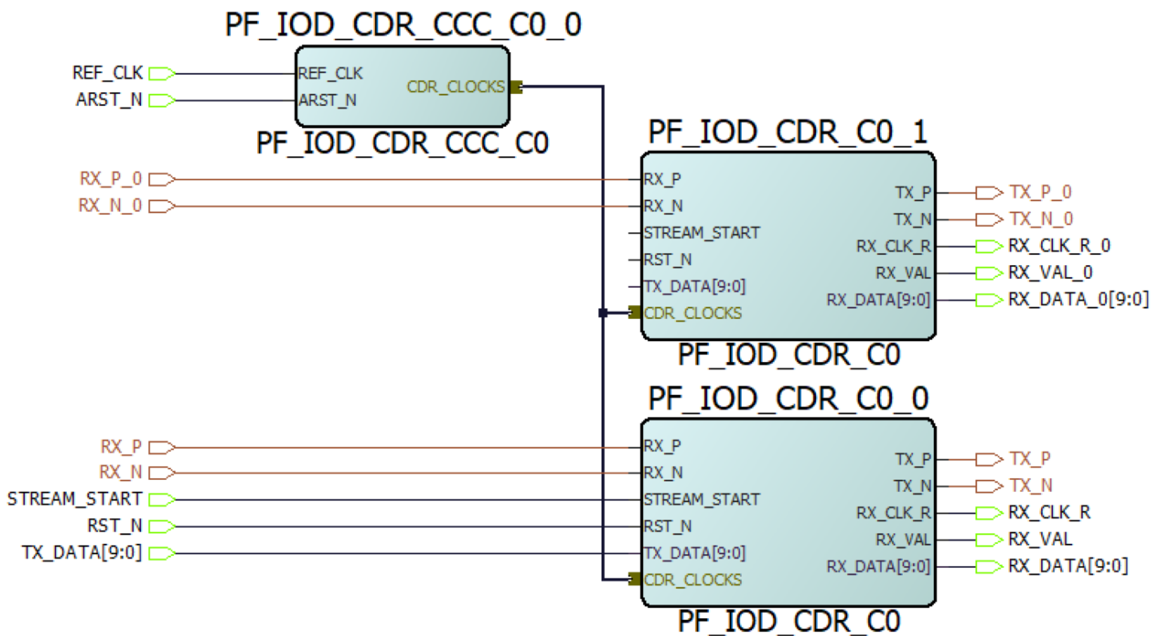


The IOD_CDR solutions requires two purpose built IP cores.

- PF_IOD_CDR
- PF _IOD_CDR_CCC

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

Figure 46 • SmartDesign of IOD_CDR Topology



6.1.1 IOD CDR

The following figure shows the IOD CDR configurator.

Figure 47 • IOD CDR Configuration

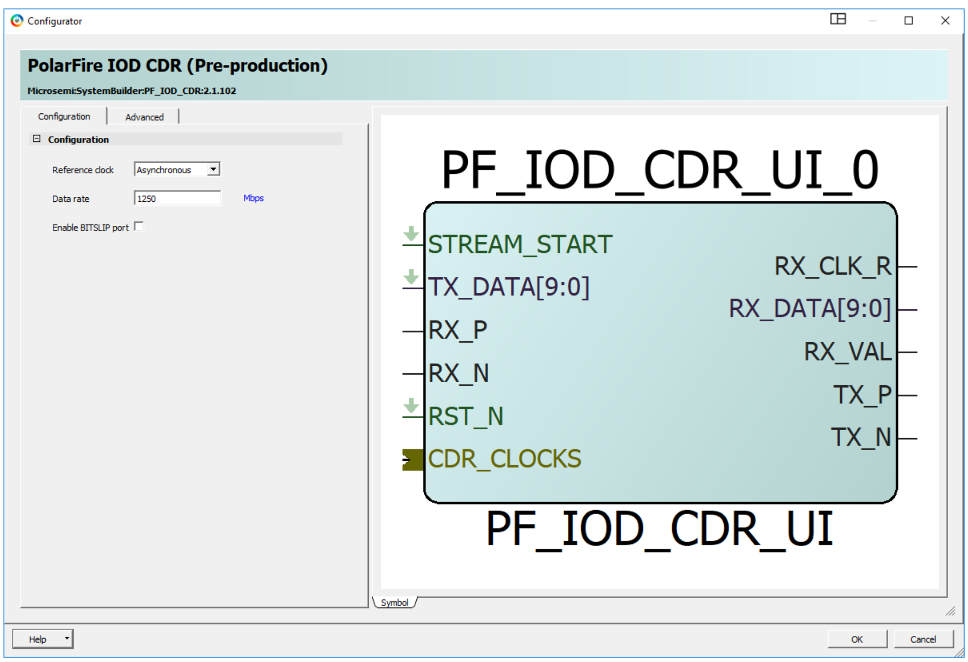


Table 40 • IOD CDR Configuration

GUI Option	Selections
Data rate	User Input – 1250 Mbps maximum
RX enabled	RX Enabled Only
Enable BITSLIP port	Disabled and Enabled
TX enabled	TX Enabled Only
Simulation mode	Full and Fast

6.1.2 Receive Interface

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.

6.1.3 Transmit Interface

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 via 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 41 • 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.
DLL_LOCK	Input	CDR_ CLOCKS	Lock signal from CCC-DLL.
TX_CLK_G	Input	CDR_ CLOCKS	Transmit clock from the Fabric.

Table 41 • PF_IOD_CDR Interface Associated Ports _(continued)_

Port	I/O	BIF	Description
DLL_CODE[5:0][2]	Input	CDR_ CLOCKS	Delay code bus input from DLL-CCC. DLL delay code for 90° phase of the data.
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	Input		High input indicates valid serial input stream
TX_DATA[9:0]	Input		Transmit parallel data.
DLL_CODE_UPDATE[3]	Input		Delay code update.
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.
TX_P	Output Pad		Serial data output (P side).
TX_N	Output Pad		Serial data output (N side).

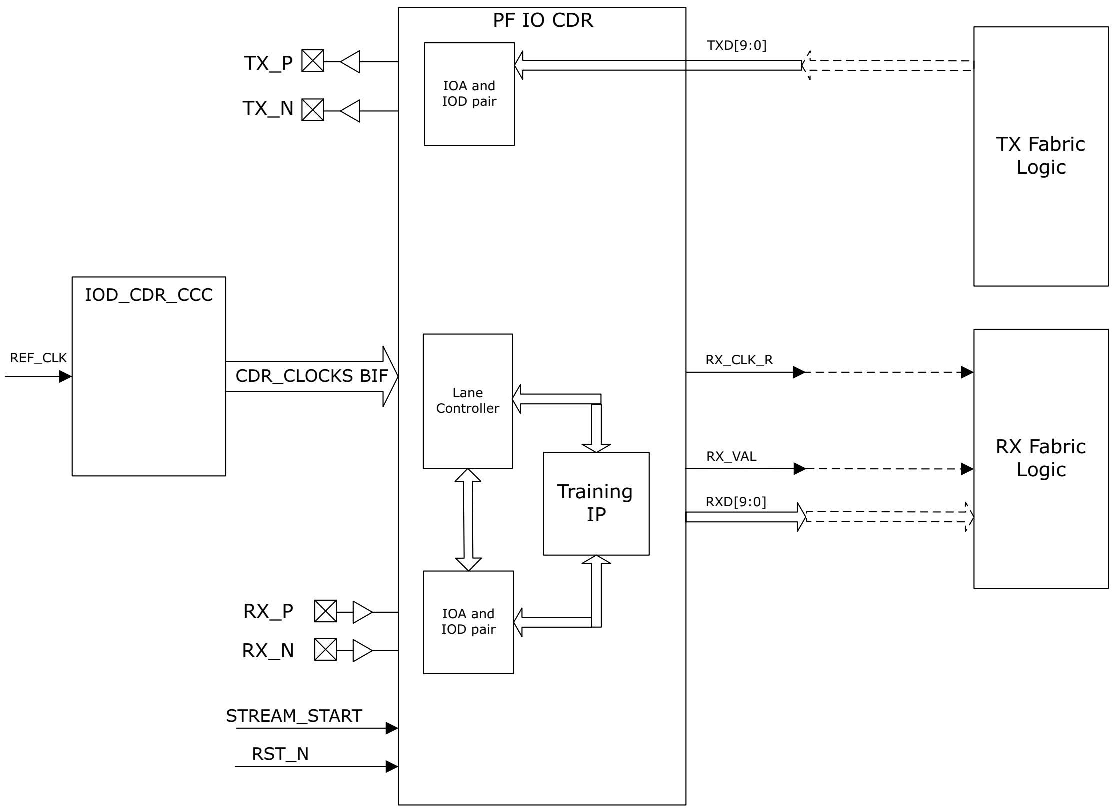
1. PLL takes any reference clock input frequency (default 125 MHz) and outputs 625 MHz clock with 0, 90, 180, and 270 degree shift on four outputs.
2. DLL takes 625 MHz reference clock input from the PLL output in Clock Reference Mode and outputs delay code as quarter of the clock cycle. The delay code is used in calculating of fine tune delay of CDR clock phase.
3. The delay code gets updated by driving high on CODE_UPDATE signal for at least two reference clock cycles. If the CODE_UPDATE is driven high and held in that state, the delay code output is continuously updated.
4. User optional pin enabling the BITSLIP exposes the RX_BIT_SLIP.
5. Resets the IOD block of the IOCDR. Does not reset DLL.

6.1.3.1 Bank Clock Generation Using PF_IOD_CDR_CCC

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 PolarFire 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 48 • **Using PF_IOD_CDR Interfaces**

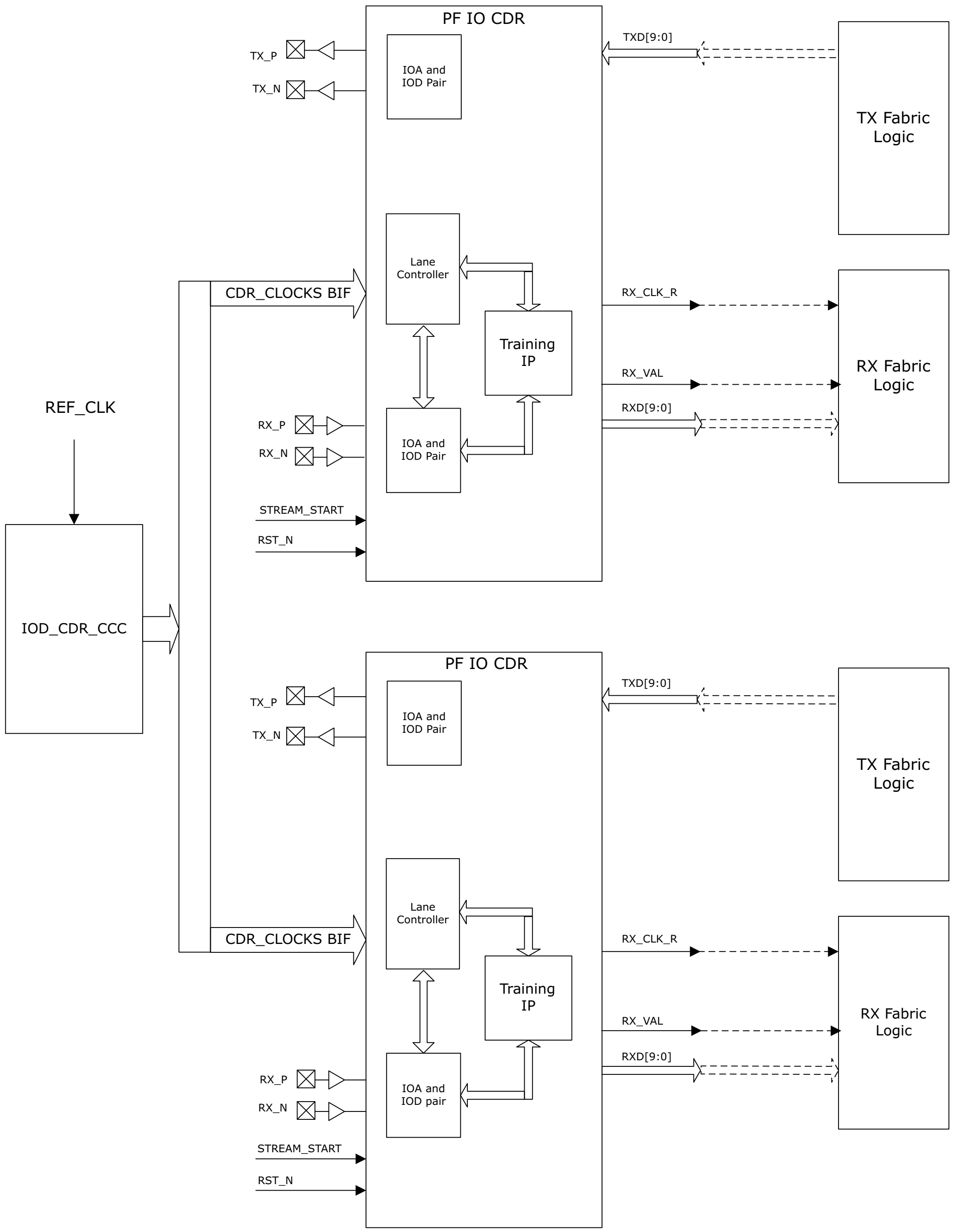


6.1.4 Clock Sharing

The same PLL is shared between the PF_IOD_CDR receive and transmit interfaces, as shown in Figure 49, page 72. 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 I/O Banks, page 8).

The following illustration shows multiple PF_IOD_CDR transmit and receive interfaces.

Figure 49 • **Multiple PF_IOD_CDR Transmit and Receive Interfaces**



6.1.4.1 Interface Selection Rules

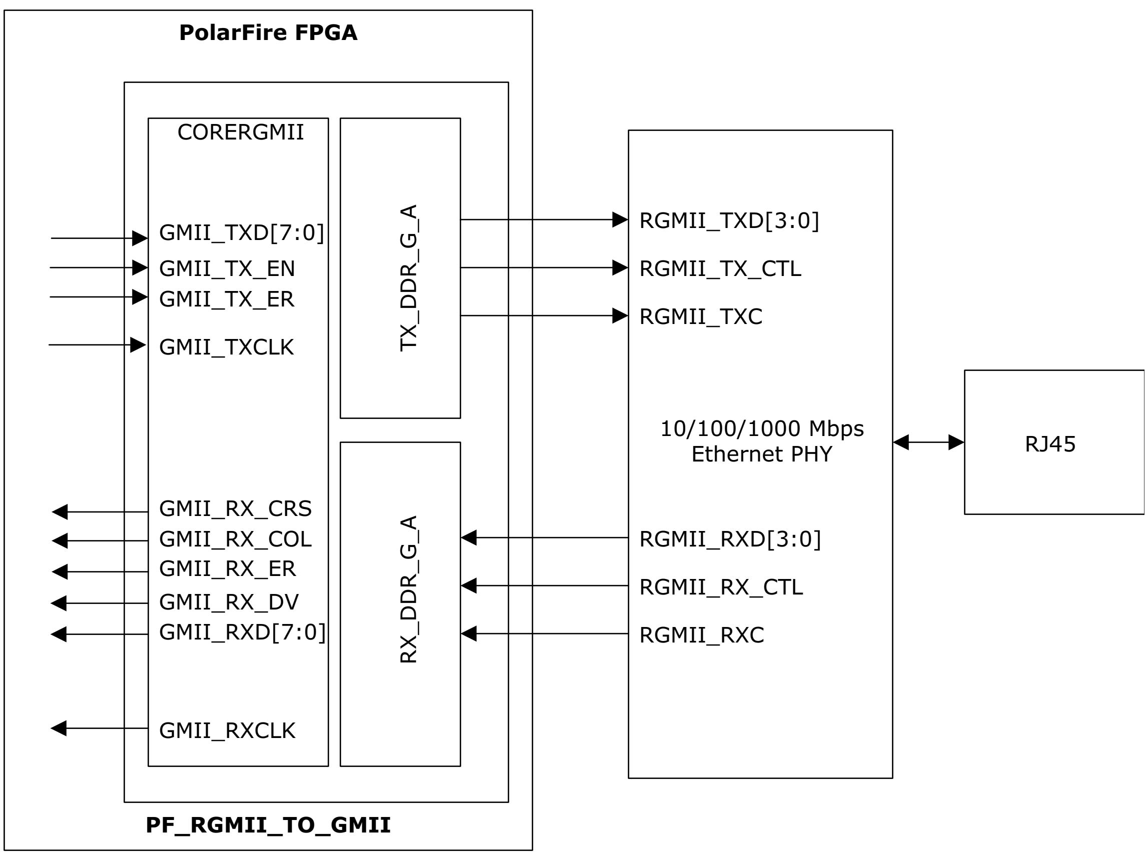
Follow these rules when assigning a pin for the PF_IOD_CDR interface:

- 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. Lanes are grouped per bank with the following two exceptions:
  - Bank7 (MPF300T/MPF500T) I/O are in the same group of lanes as I/O in bank0
  - For devices without Bank6 (MPF100T/MPF200T): Bank2 is split into two groups of lanes. The project requires one IOD_CDR_CCC per group of lanes. The two groups are { DDR_S_0* DDR_S_1 DDR_S_2 } and { DDR_S_3 DDR_S_4 DDR_S_5 DDR_S_6 DDR_S_7 } (* only in MPF200T). See *PolarFire* web page for information about PPAT files definitions.
- 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.

6.2 RGMII to GMII Converter

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 the PolarFire FPGA device family using the IOD blocks used with PolarFire GPIO or HSIO buffers.

Figure 50 • **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.

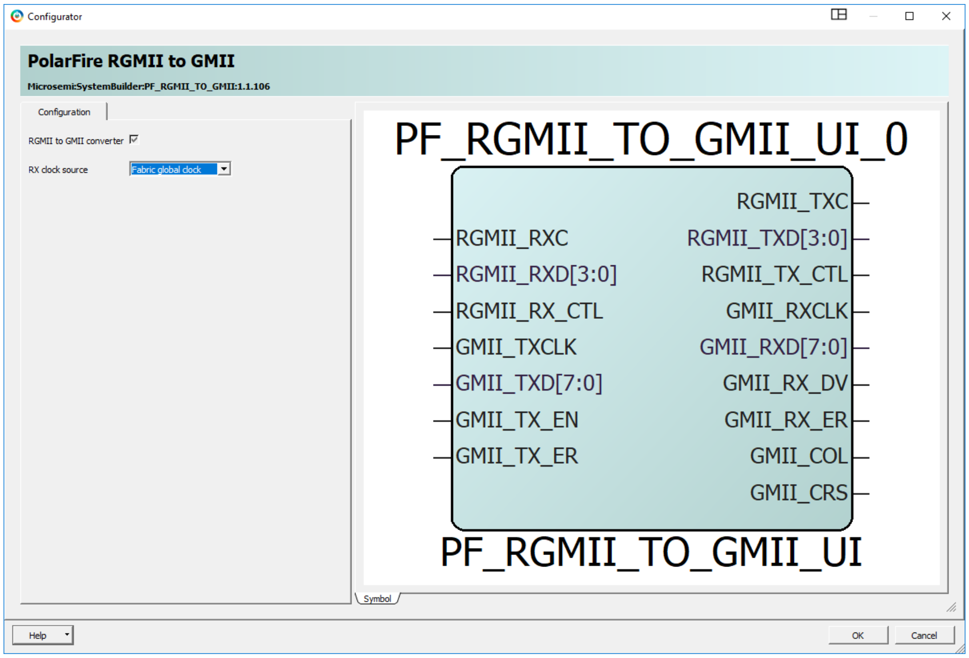
The following table lists the GMII/RGMII ports and description.

Table 42 • **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 via 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
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 51 • **PolarFire 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. Users 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 should reference the pin selection rules discussed in Interface Selection Rules, page 48.

Figure 52 • **RX_DDR_G_A Interface Configuration Tab—Used with RGMII To GMII Configuration**

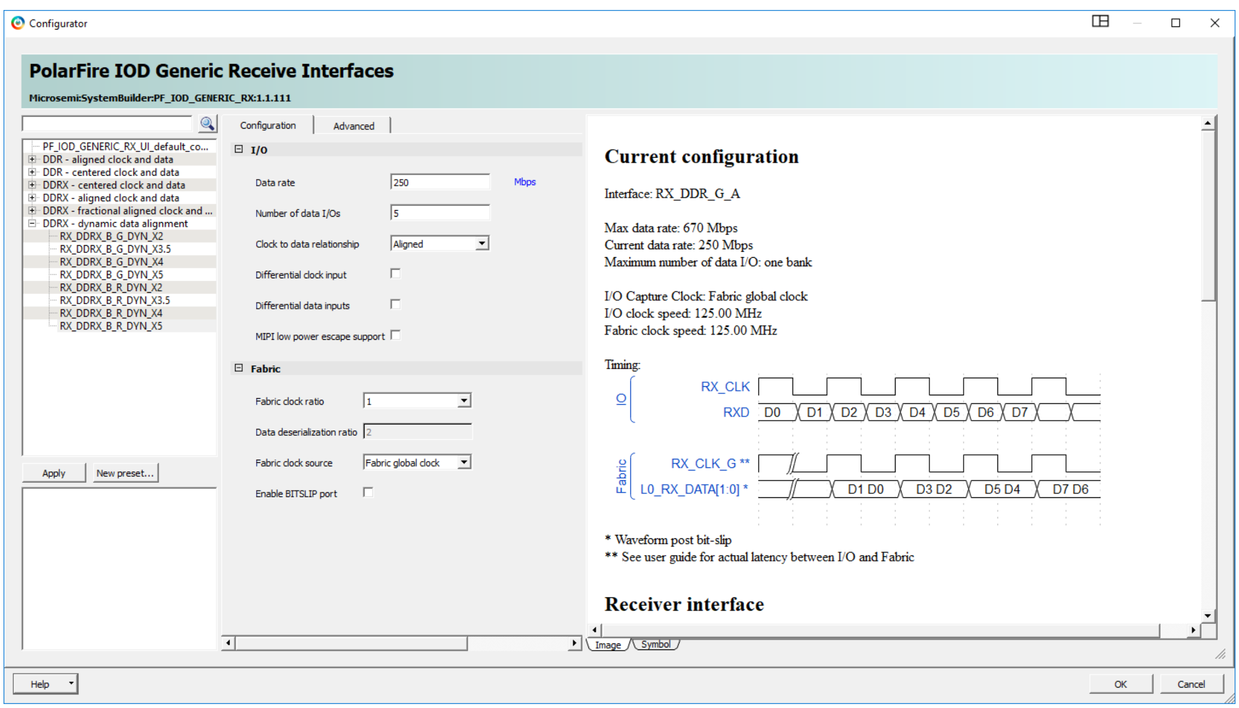


Figure 53 • **RX_DDR_G_A Interface Advanced Tab—Used with RGMII To GMII Configuration**

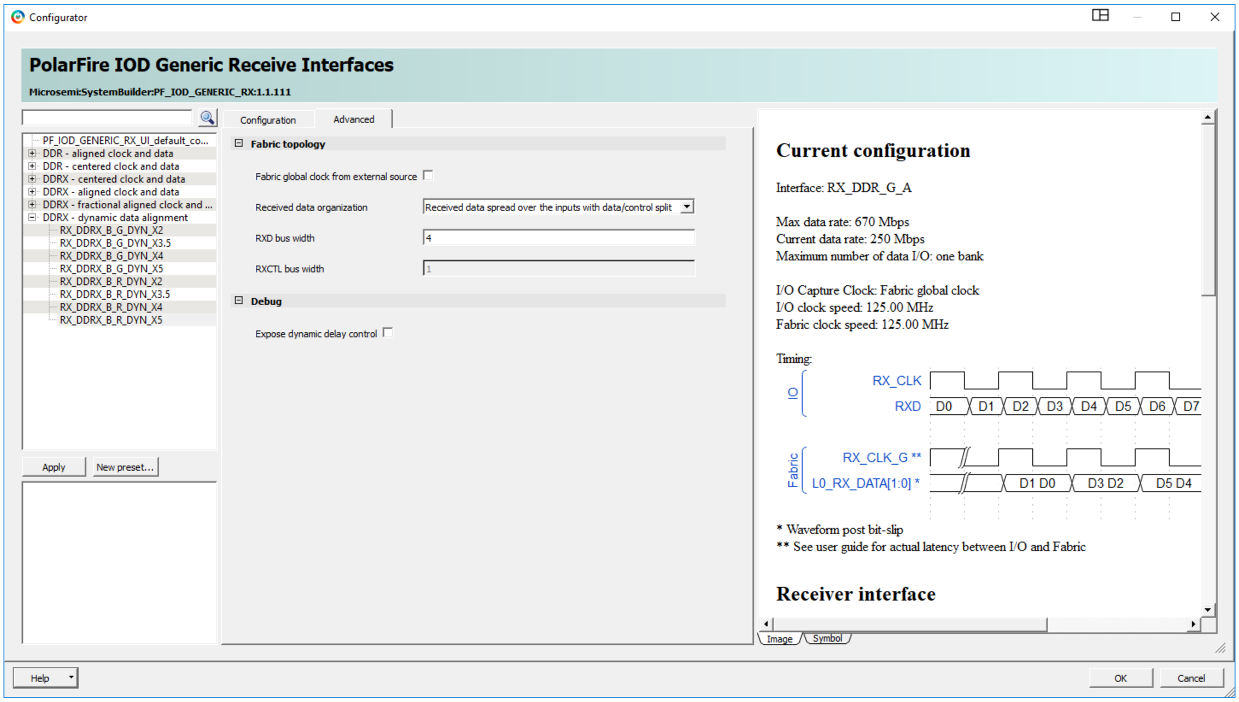


Figure 54 • **RX_DDR_G_A Waveform**

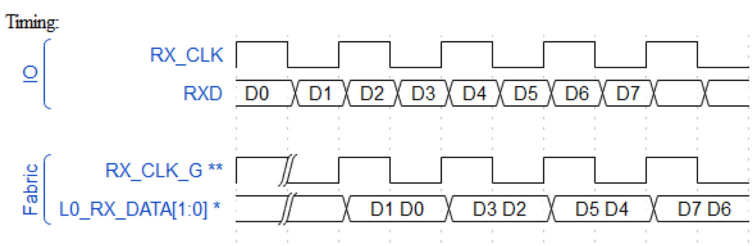


Figure 55 • **TX_DDR_G_A Configuration Tab—Used with RGMII To GMII Configuration**

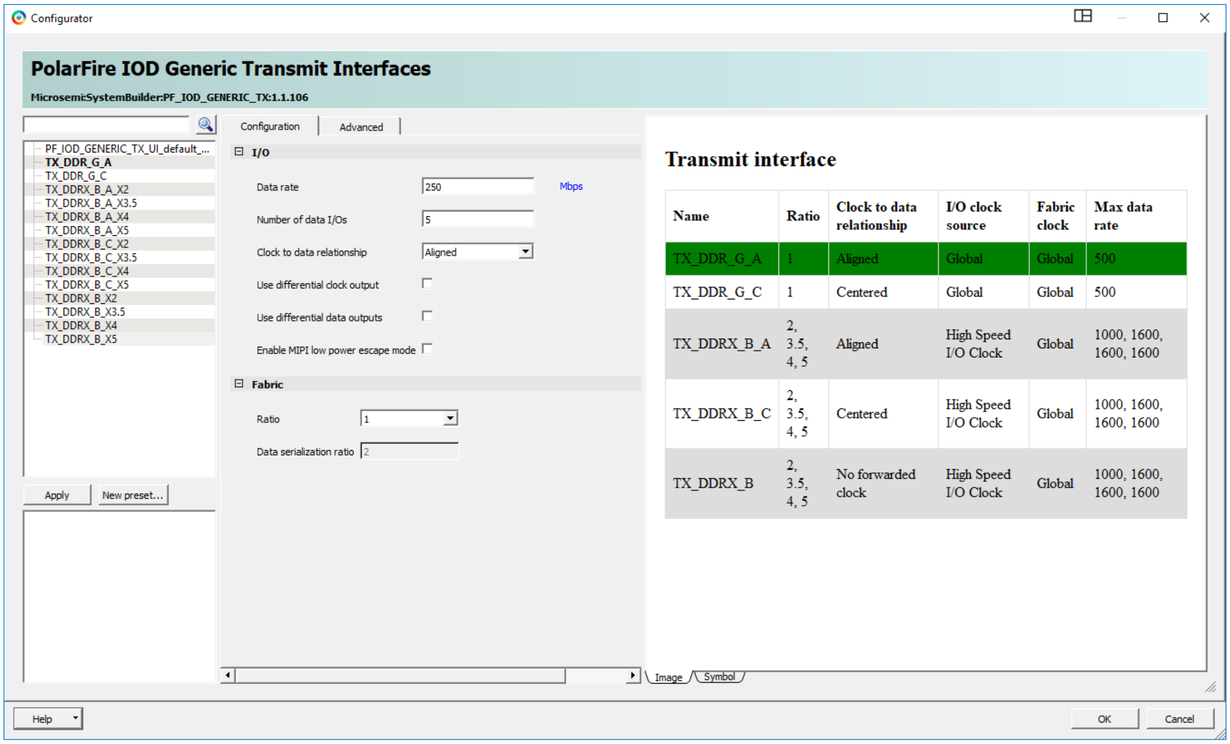
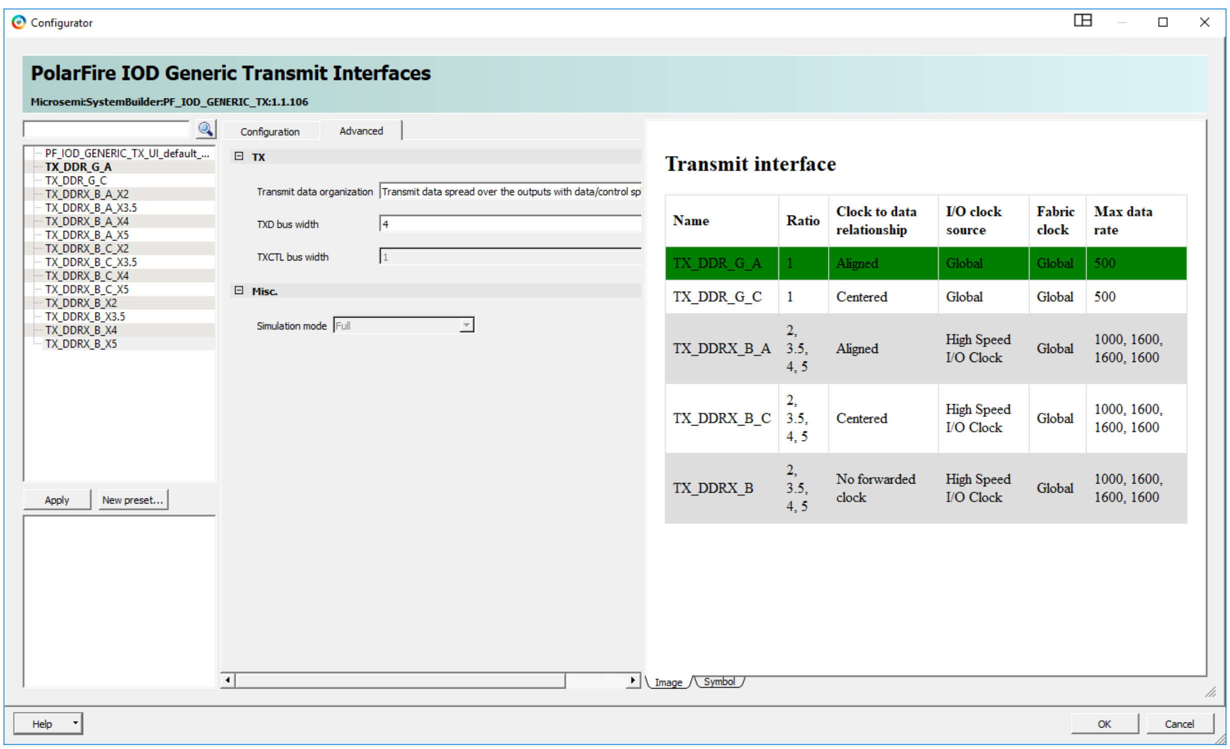


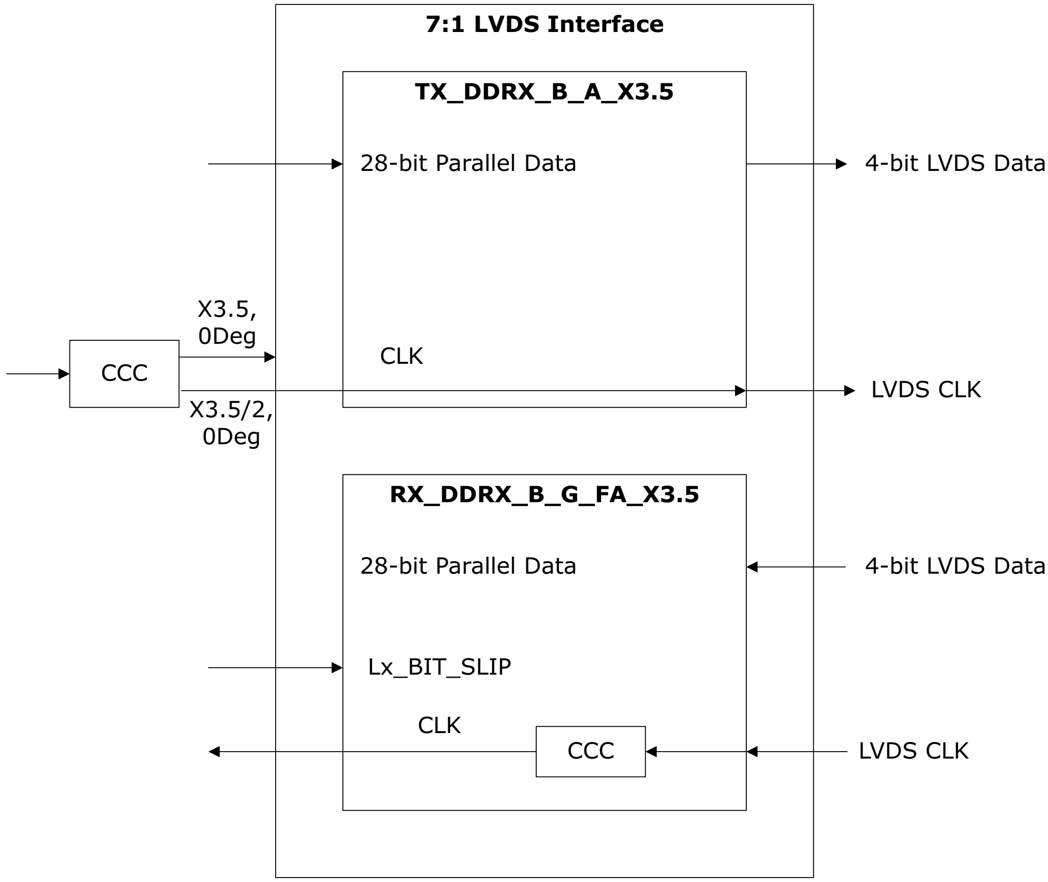
Figure 56 • **TX_DDR_G_A Advanced Tab—Used with RGMII To GMII Configuration**



6.3 LVDS 7:1

A typical source-synchronous interface application is the 7:1 LVDS video interface (used in Channel Link, Flat Link, and Camera Link). This have 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.

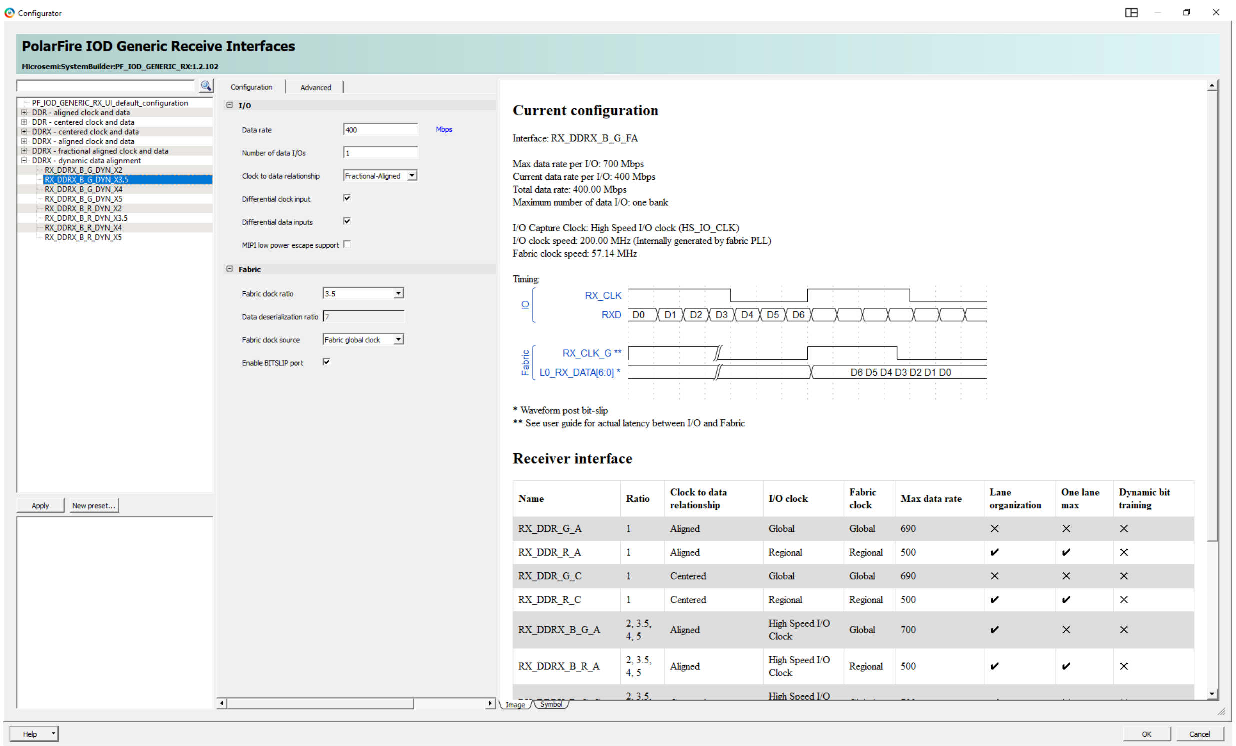
Figure 57 • **Example of 7:1 LVDS Interface—Four Data and One Clock**



6.3.1 7:1 LVDS Receive Interface

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 used to re-align the data words arriving on the rising edge of the fractional clock. The data is deserialized to 7-bit data that is sent to the fabric with a forwarded clock.

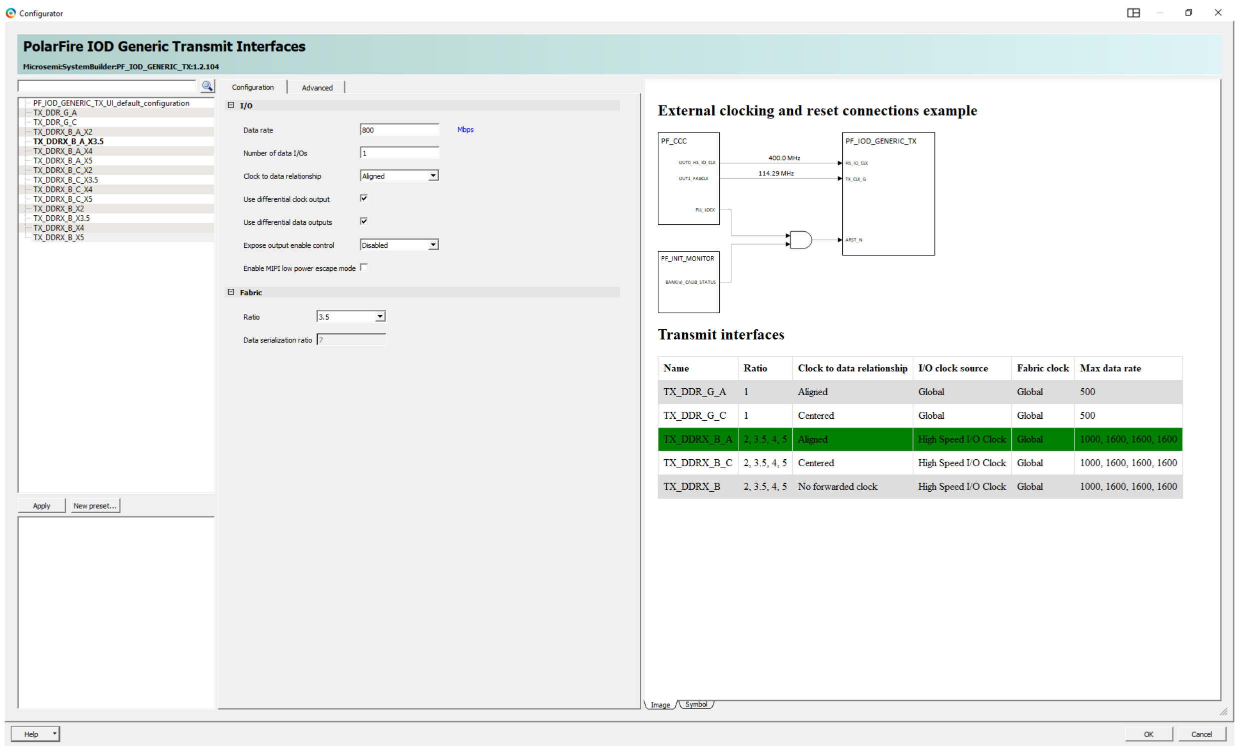
Figure 58 • **RX_DDRX_B_G_FA Interface**



6.3.2 7:1 LVDS Transmit Interface

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 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 59 • **TX_DDRX_B_A_X3.5**



7 Dynamic IOD Interface Training

7.1 Clock to Data Margin Training

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).

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 IO 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 RX_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.

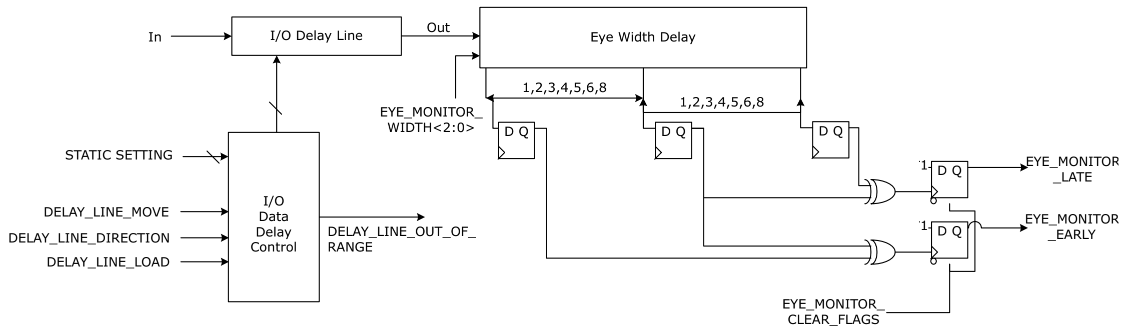
The IOD block has a data eye monitor (DEM) used to optimize the clock and input data relationship. The DEM includes EYE_MONITOR_EARLY and EYE_MONITOR_LATE flags used to analyze the clock-to-data relationship. IOD designs can utilize these flags to determine the input data edge relationship to the clock 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.

The data edge 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 should 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 should 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 60 • **IOD Training Block Diagram**



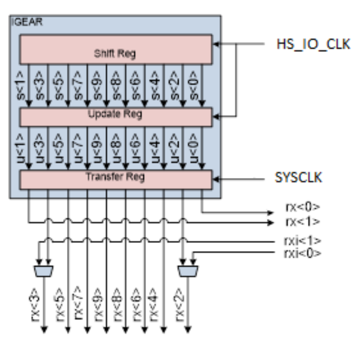
7.1.1 HS_IO_CLK and System Clock Training

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.

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 61, page 82.

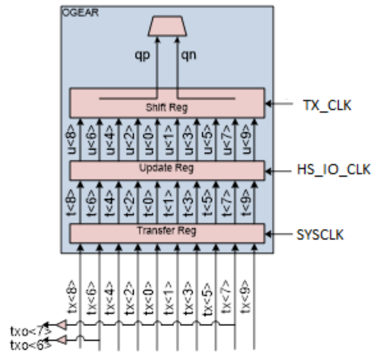
- Shift register
- Update register
- Transfer register

Figure 61 • **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 62 • **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 63 • **SYSCLK to HS_IO_CLK Before Training**

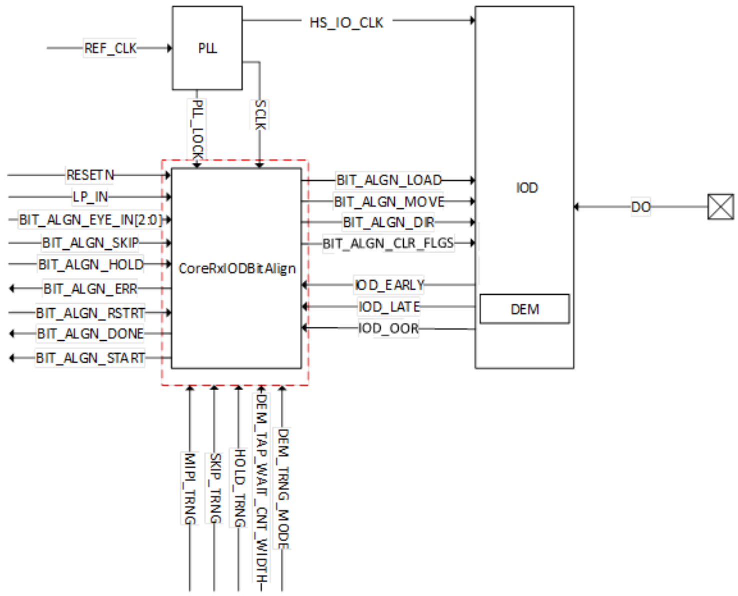
Figure 64 • **SYSCLK to HS_IO_CLK After Training**

In the Figure 63, page 83 and Figure 64, page 83, 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.

7.2 CoreRxIODBitAlign

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

Figure 65 • **CoreRxIODBitAlign Implementation Diagram**



This CoreRxIODBitAlign IP works based on Fabric clock or SCLK or OUT2_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 1000Mbps in DDRx4 fabric mode. The OUT2_FABCLK_0 or SCLK should be driven from the PLL or 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 neg 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.

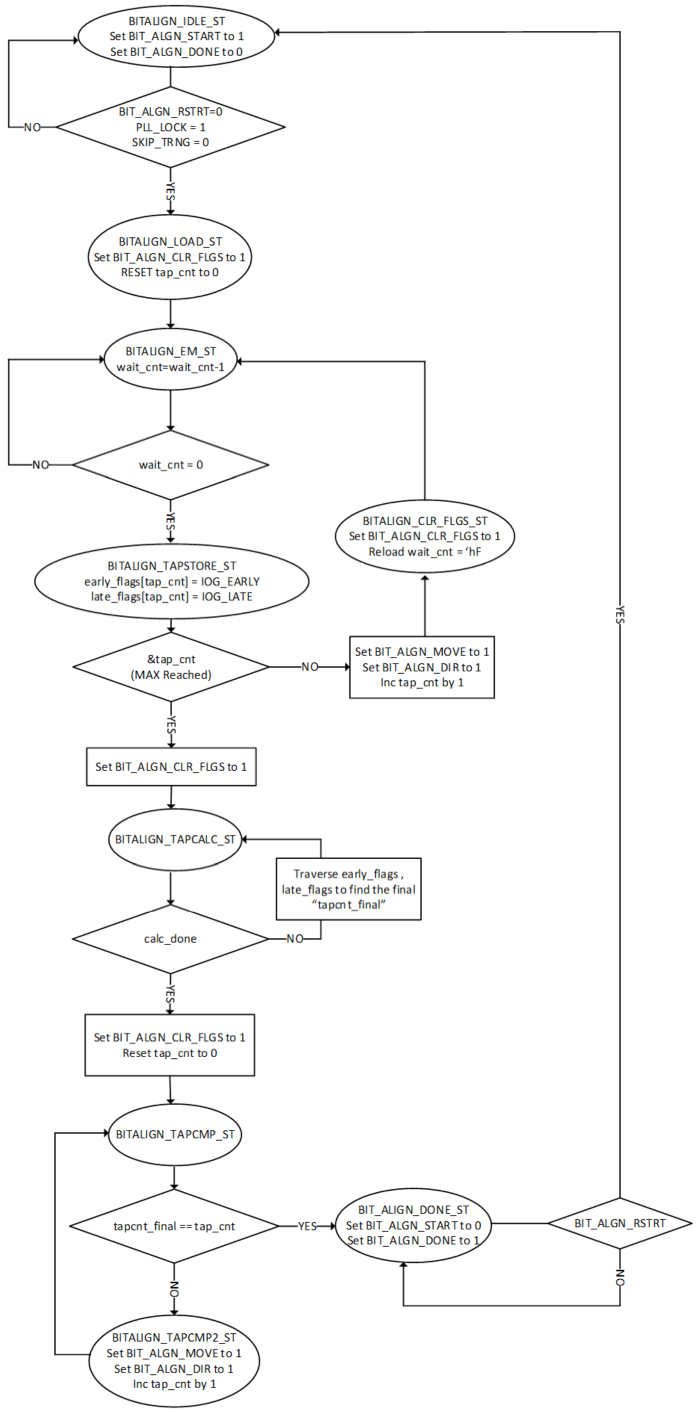
The CoreRxIODBitAlign IP loads the calculated TAP delays and drives BIT_ALGN_START low and BIT_ALGN_DONE high to indicate the completion of the training.

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 should 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 should 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 66 • **CoreRxIODBitAlign Training State Diagram**



7.2.1 CoreRxIODBitAlign Training Algorithm

Figure 66, page 85 shows the states of the training IP. The states of the diagram are detailed as follows.

0. BITALIGN_IDLE_ST

- If the input PLL_LOCK(1), BIT_ALGN_RSTRT(0), SKIP_TRNG(0) parameter, Goto Step 1 else Goto Step 0
- Set BIT_ALGN_START to 1 and BIT_ALGN_DONE to 0

1. BITALIGN_LOAD_ST

- Set BIT_ALGN_CLR_FLGS to 1 to reset the IOD_EARLY and IOD_LATE outputs from the DEM IOG block
- Set BIT_ALGN_LOAD to 1 for default configuration
  - Reset the tap_cnt[7:0] = 0 for starting the training sequence
- Goto Step 2

2. BITALIGN_EM_ST

- Decrement the wait_cnt by 1 (Loaded initially as 'hF when RESETN is 0)
- Wait for delay count (wait_cnt == 0) till the tap delays take effect
  - if (wait_cnt == 0) then Goto Step 3 else Goto Step 2

3. BITALIGN_TAPSTORE_ST

- Check tap_cnt[7:0]
  - If MAX Reached, set BIT_ALGN_CLR_FLGS to 1 to reset the IOD_EARLY and IOD_LATE outputs from the DEM IOG block and Goto Step 4
  - If MAX NOT Reached, set BIT_ALGN_MOVE, BIT_ALGN_DIR to increment the tap_cnt by 1. Save early_flags[tap_cnt], late_flags[tap_cnt] status using the value of tap_cnt as an index and Goto Step 5

4. BITALIGN_TAPCALC_ST

- If calc_done (Completion flag for BITALIGN_TAPCALC_ST) is set, then set BIT_ALGN_CLR_FLGS to 1, Reset the tap_cnt to 0 and Goto step 6 or follow the below step
- Traverse the early_flags[emflag_cnt] and late_flags[emflag_cnt] to find the final tap delays using emflag_cnt as an index.
  - Initial values during RESETN(0)
    early_set = 0 (First Early), early_val = 0 (FIRST Early value)

    late_set = 0 (Next late), late_val = 0 (NEXT Late value)

    tapcnt_final=0 (Final tap delay to be set for alignment)

    calc_done=0 (Completion flag for BITALIGN_TAPCALC_ST)
  - Increment emflag_cnt by 1
  - Traverse early_flags[emflag_cnt] and find LAST early_set and early_val
    Set early_set to 1

    Assign emflag_cnt value to early_val as tapcnt_final (FIRST Early value)
  - Traverse late_flags[emflag_cnt] and find FIRST late_set and late_val
    Check if early set is set to 1 for completion

    Set late_set to 1

    Assign emflag_cnt value to late_val as tapcnt_final (NEXT late value)
  - Compare ii and iii
    Check if early set and late_set is set to 1 for completion

    if (early_val < late_val), set tapcnt_final = (early_val + late_val) >> 1

    if (early_val > late_val), set tapcnt_final = early_val

    Set calc_done to 1 and Go step 4.1

5. BITALIGN_CLR_FLGS_ST

- Set BIT_ALGN_CLR_FLGS to 1 to reset the IOD_EARLY and IOD_LATE outputs from the DEM IOG block

- Reload the wait_cnt as 'hf and Goto Step 2

6. BITALIGN_TAPCMP_ST – (Set the tapcnt_final delays FOUND in DEM)

- If tapcnt_final (FOUND) matches with tap_cnt Goto step 8 else set BIT_ALGN_CLR_FLGS to 1 and Goto Step 7

7. BITALIGN_TAPCMP2_ST

- Set BIT_ALGN_MOVE, BIT_ALGN_DIR to increment the tap_cnt by 1 and Goto Step 6

8. BITALIGN_DONE_ST

- Set BIT_ALGN_START to 0 and BIT_ALGN_DONE to 1

7.2.2 Interface Parameters

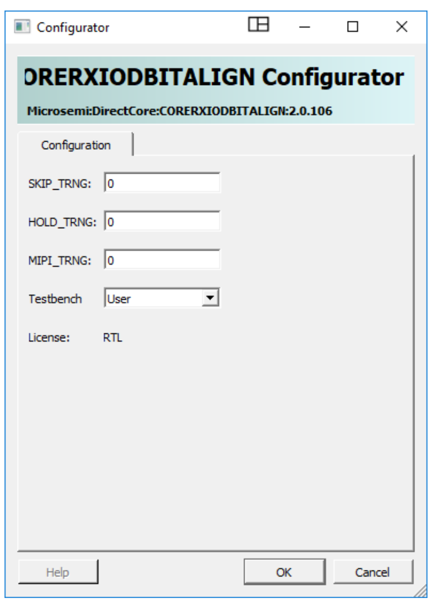
The following table lists the interface parameters of CoreRxIODBitAlign IP.

Table 43 • **CoreRxIODBitAlign Interface Parameters**

Parameter	Description
SKIP_TRNG	0 - Skip All Training (Bypass Mode) 1 - Perform training as usual
HOLD_TRNG	0 - Hold Training 1 - Release Training to continue
MIPI_TRNG	0 - Perform training as usual (LP_IN port is Disabled) 1 - Perform MIPI training (LP_IN port is Enabled)
DEM_TRN_MODE	0 - 128 Tap delay-based Training 1 - 256 Tap delay-based Training (Default)
DEM_TAP_CNT_WIDTH	3 - Width of the Tap delay counter (Default) Note: This delay count corresponds to the wait delay of the IP for each tap and so as to record the early late flags. Say the width is set as 3 then 7 clocks are used as a delay counter in the IP.

The following figure shows the CoreRxIODBitAlign Libero Configurator.

Figure 67 • **CoreRxIODBitAlign Libero Configurator**



7.2.3 CoreRxIODBitAlign Ports

The following table lists the CoreRsIODBitAlign ports.

Table 44 • **CoreRxIODBitAlign Ports**

Port Name	I/O	Description
Clocks and Reset		
SCLK	Input	Fabric clock
PLL_LOCK	Input	PLL Lock
RESETN	Input	Active low Asynchronous Reset
Data Bus and Control Signals		
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
BIT_ALGN_EYE_IN[2:0]	Output	Sets the data eye monitor width by user
BIT_ALGN_RSTRT	Input	Bit Align Training restart (Pulse based Assertion) 1 - Restart Training 0 - No Restart Training
BIT_ALGN_CLR_FLGS	Output	Clear Early/Late flags
BIT_ALGN_LOAD	Output	Load default
BIT_ALGN_DIR	Output	Delay line up/down direction 1 - up (increment 1 tap) 0 - down (decrement 1 tap)
BIT_ALGN_MOVE	Output	Increment the delay on move pulse
BIT_ALIGN_HOLD	Input	Bit Align Training hold (Level based Assertion) 1 - Hold the training and valid only when HOLD_TRNG parameter is set to 1 0 - Training should proceed as normal
BIT_ALIGN_ERR	Output	Bit Align Training error (Level based Assertion) 1 - Error 0 - No Error
BIT_ALGN_START	Output	Bit Align Training start (Level based Assertion) 1 - Started 0 - Not Started
BIT_ALGN_DONE	Output	Bit Align Training done (Level based Assertion) 1 - Completed 0 - Not Completed
BIT_ALGN_SKIP	Input	Bit Align Training skip (Level based Assertion) 1 - Skip the training and valid only when SKIP_TRNG parameter is set to 1 0 - Training should proceed as normal
LP_IN	Input	Bit Align MIPI Training 1 - Perform the training and valid only when MIPI_TRNG parameter is set to 1 0 - Training should proceed as normal