DG0898
Demo Guide
PolarFire 10G SyncE Solution with ESMC and Jitter Attenuating PLL

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.

50200898. 1.0 11/21

Contents

Figures

Tables

1 Revision History

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

1.1 Revision 1.0

The first publication of this document.

2 References

- *Timing characteristics of synchronous equipment slave clock*
- *UG0677: PolarFire FPGA Transceiver User Guide*
- *DS0141: PolarFire Datasheet*
- *Core10GMAC Handbook*
- *Distribution of timing information through packet networks (G.8264)*
- *Timing and synchronization aspects in packet network*

3 Introduction

This application describes how Synchronous Ethernet technology provides a solution for transporting synchronization over Ethernet. SyncE uses frequency synchronization from the physical layer together with a slow protocol and has been standardized by ITU-T for frequency distribution. SyncE leverages the PHY layer of Ethernet to transmit frequency to remote sites or other nodes.

SyncE is implemented according to the following ITU-T recommendations:

- G.8261 – Defines the architecture of Synchronous Ethernet networks
- G.8262 – Specifies the timing characteristics of synchronous Ethernet Equipment Clock (EEC)
- G.8264 – Describes the Ethernet Synchronization Messaging Channel (ESMC) which is used to transfer the clock information in the form of Synchronous Status Message (SSM).

PolarFire® Gigabit Transceiver in conjunction with Jitter Attenuating Phase Lock Loop (JAPLL) can provide Synchronous Ethernet solution for data rates up to 12.5 Gbps when the transmitter and the receiver are in the same clock domain.

Jitter Attenuating PLL:

Jitter Attenuating PLL can modulate the transmit clock in-order to lock to an input source. In the absence of any ports, the Jitter Attenuating PLL will go in to hold over mode by locking to an on-board reference clock source.

3.1 Key Features

- ITU-T SyncE compliant PLL
- Ability to hold outputs when no reference clock is available
- Ability to detect loss of signal and perform a Hitless clock switching (Phase of the output clock does not change when PLL switches input clocks)
- Maintain frequency accuracy in the range of +/- 4.6 ppm
- Limited phase wander as measured by TDEV and MTIE.
- Multi-rate protocol support such as:
  - OTN
  - PON
  - CPRI {Interface between REC and RE}
  - Synchronous Ethernet {Gbe}
  - Video applications {SDI, HDI, 3G-SDI, etc.}

3.2 Prerequisites

Before you begin:

1. For demo design files download link:
   http://soc.microsemi.com/download/rsc/?f=mpf_dg0898_df
2. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location:
   https://www.microsemi.com/product-directory/design-resources/1750-libero-soc#downloads

3.3 Ethernet Synchronization Messaging Channel (ESMC)

SyncE uses ESMC PDU's to transfer the clock source information which allows for clock source traceability to correctly define the timing source to prevent timing loops. ESMC uses the Organization Specific Slow Protocol (OSSP) for transferring and receiving the QL value. ESMC QL values provide the information about the clock quality which helps the node to derive the timing from the most reliable source.

Figure 1 • **ESMC PDU Format**

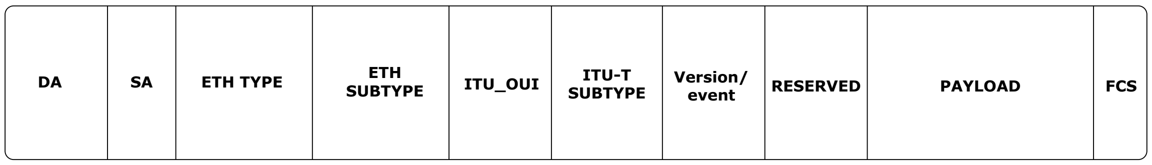


Table 1 • **ESMC QL Values**

Byte number	Size	Field	Value	
1-6	6 bytes	Destination Address	48'h0180_C200_0002	
7-12	6 bytes	Source Address	48'h0000_0000_0002	
13-14	2 bytes	Ethernet type	16'h8809	
15	1 byte	Ethernet sub-type	8'h0A	
16-18	3 bytes	ITU-OUI	24'h0019A7	
19-20	2 bytes	ITU-T Subtype	16'h0001	
21	1 byte	Version (bits 7:4)	4'h1	
		Event flag (bit 3)	1 indicates event PDU, 0 indicates info PDU	
		Reserved (bits 2:0)	Reserved	
22-24	3 bytes	Reserved	Reserved	
25-1532	4 bytes	TLV_TYPE (1byte)	8'h01	
		TLV_LENGTH (2 bytes)	16'h0004	
		SSM CODE(1byte)	PRC	8'h02
			SSU-A	8'h04
			SSU-B	8'h08
			SEC	8'h0B
			DNU	8'h0F
	32-1486 bytes	Future Enhancement		
Last 4	4 bytes	FCS	Core10GMAC computes the FCS and appends to the Frame before transmitting out.	

The Field description is as follows:

- Destination Address (DA): This is the IEEE-defined slow protocol multicast address.
- Source Address (SA): The source address is the MAC address associated with the port through which the ESMC PDU is transmitted.
- Slow Protocol Ether type: ESMC PDU's must be type encoded and carry the slow protocol type field value.
- Slow Protocol subtype: Assigned by the IEEE and fixed with a value of 0x0A.
- ITU OUI: Organizational unique identifier assigned by the IEEE registration authority.
- ITU subtype: Assigned by ITU-T. The value of 00-01 applies to all usage defined in this recommendation.
- Version: The four-bit field indicates the version of the ITU-T OSSP frame format. This field shall contain the value 0x1 to claim compliance with version 1 of this protocol.
- Event flag: This bit distinguishes the critical, time-sensitive behavior of the ESMC event PDU from the ESMC Information PDU. A value of 1 indicates an event PDU and a value of 0 indicates an information PDU.
- Data and Padding: The minimum frame size of 64 bytes and the maximum size for the ESMC PDU is 128 bytes.
- FCS: Four-byte frame check sequence as defined in clause 4 of [IEEE 802.3].

The PolarFire® FPGA 10G SyncE solution is compliant with the IEEE 802.3ae standard, which supports data transfer rates of up to 10.3125 Gbps. Advantages offered by using PolarFire FPGAs for building 10G Ethernet solutions include the use of low-power transceivers, low-power FPGA fabric, and in-built SyncE-compliant jitter attenuation.

The 10G Ethernet solution is implemented using the CORE10GMAC soft IP Media Access Control (MAC) core, which can be configured either in 10GBASE-KR mode or 10GBASE-R mode. This demo design includes the following design, which can be used as reference designs for building a 10GBASE-R

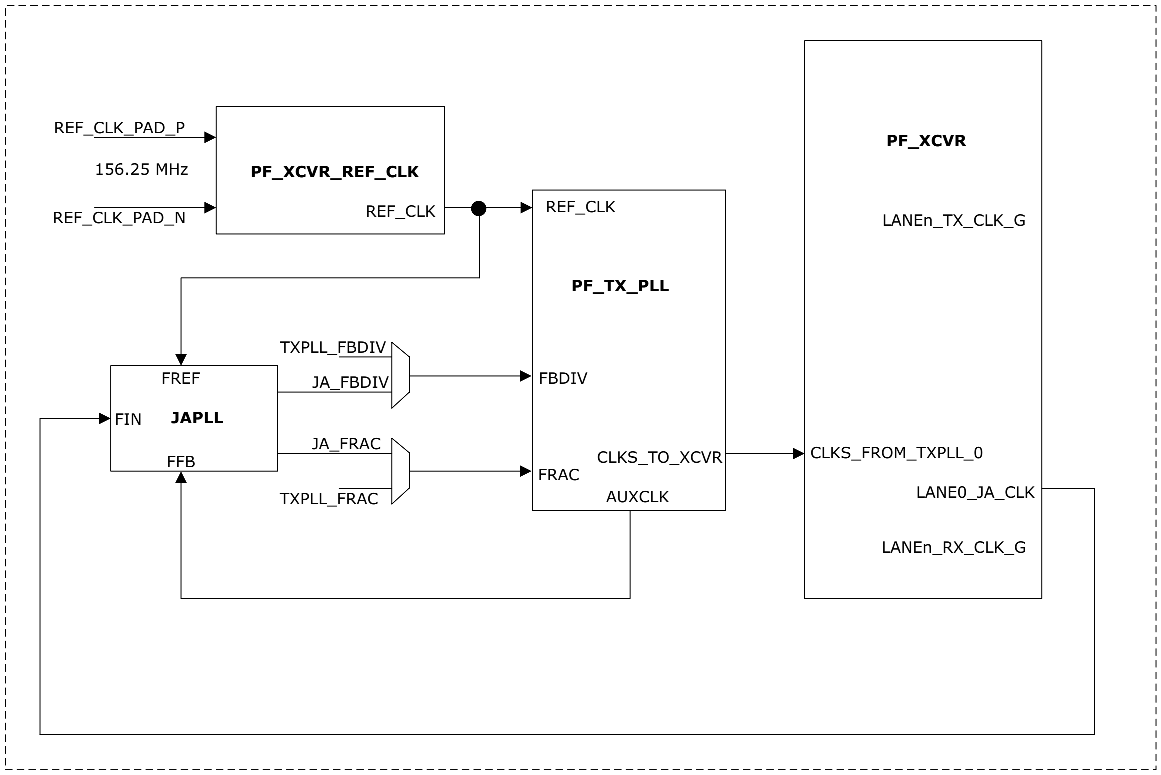
Ethernet loopback application:

- A 10GBASE-R Ethernet 64-bit loopback design that can be run on the PolarFire Evaluation Board using Paragon-X *Network Tester*.

4 PolarFire Jitter Attenuator

4.1 General Operation

Figure 2 • **General Operation of PolarFire Jitter Attenuator**



The JAPLL interfaces with Transmit PLL (PF_TXPLL) by controlling the PLL's integer and fractional feedback division bits. The PF_TXPLL will automatically be configured as a Fractional PLL when jitter cleaning mode is selected in the PF_TXPLL configuration tool in Libero. The output clock from the TXPLL is tied to the Feedback clock (FFB) input of the JAPLL to compare to the JAPLL’s noisy reference input clock (FIN). The JAPLL has three phases of acquisition, frequency tracking, coarse phase tracking, and fine phase tracking. Each phase of acquisition utilizes programmable gains to tune the performance of respective stages.

Frequency gain control and frequency tolerance control modify the response time of PLL when in frequency tracking mode and allow tracking or filtering of a reference, tunable to the expected noise on the reference. Coarse phase gain control modifies the response while in phase tracking mode, ensuring the PLL can pull-in regardless of how large the frequency error may be when frequency lock is established. Fine phase gain control permits the PLL to have very low bandwidth, effectively averaging out higher frequency noise that may be due to spread spectrum or other noisy input references

4.1.1 Frequency Tracking Mode

JAPLL will enter the frequency tracking loop after reset and will start sampling inputs FIN and FFB. Once the frequency of FIN is within 5000 ppm of the frequency of FFB, FLOCK will be set to 1 and the JAPLL will transition to phase tracking mode and TRANSITION will be set to '1'. The frequency tracking loop will continue to function while in phase tracking mode.

4.1.2 Phase Tracking Mode

If the frequency error between FIN and FFB increases beyond 5000 ppm, FLOCK is reset to '0' and the loop will exit phase tracking mode and re-enters frequency tracking mode and remains in this stage till the frequency lock is re-achieved.

Once the PLL is in phase tracking mode, it will remain in this mode for DELAYK feedback clock cycles. It will spend first DELAYK/2 cycles times in coarse phase tracking mode and the remaining half in fine phase tracking mode. At this point, PHASE_LOCK will be set to '1' and TRANSITION will be set to '0' indicating the PLL is locked to phase and frequency.

When PHASE_LOCK is '1', the PLL will not track high-frequency offset signals and its output frequency will remain relatively stable. PLL will only exit this state if the frequency lock is lost.

Figure 3 • **Jitter Attenuating PLL Block Diagram**

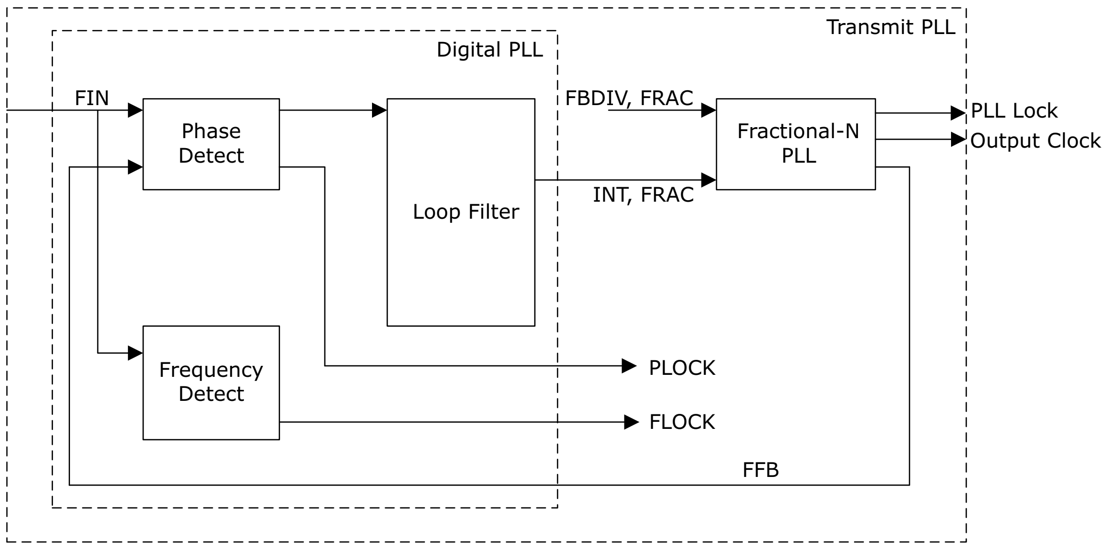


Figure 4 • **Register Information**

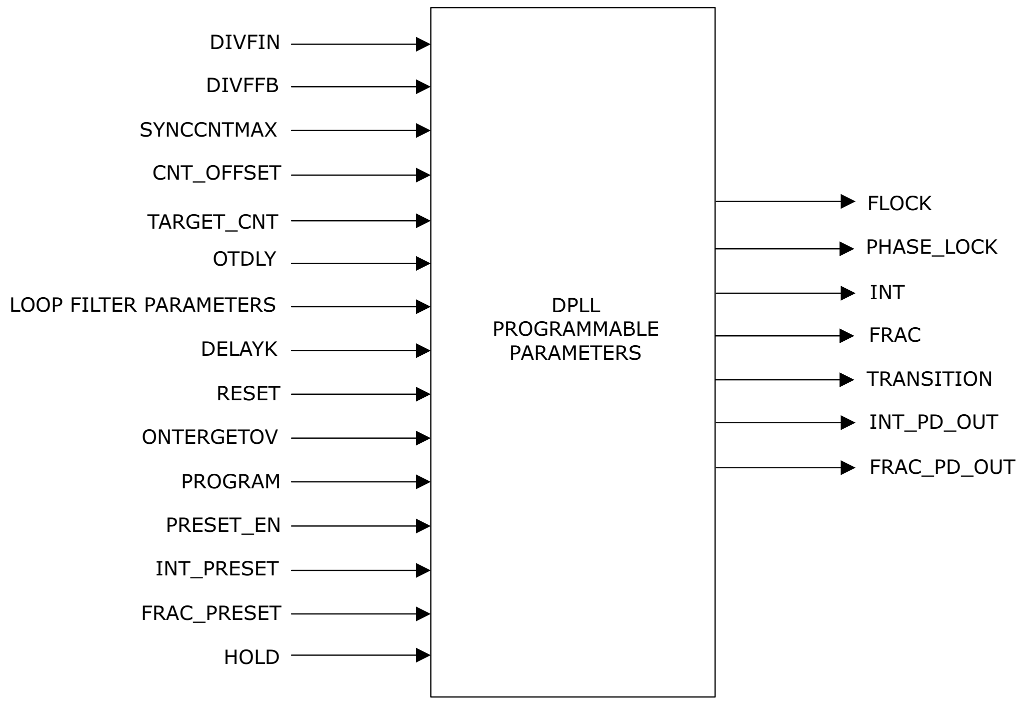


Table 2 • **Register Details**

SCB REGISTER	DPLL PARAMETER	DESCRIPTION
TXPLL_JA_1	TXPLL_JA_DIVFIN	Integer divider for FIN
	TXPLL_JA_DIVFFB	Integer divider for FFB
TXPLL_JA_2	TXPLL_JA_SYNCCNTMAX	Maximum count values to determine frequency match
TXPLL_JA_3	TXPLL_JA_CNTOFFSET	OFFSET value used for frequency tracking
	TXPLL_JA_TARGETCNT	If pulse count between FIN and FFB are less than TARGETCNT then the frequency detector will declare frequency match
TXPLL_JA_4	TXPLL_JA_OTDLY	Time duration to delay the FFB pulses when transitioning from frequency tracking to phase tracking stage to avoid glitches in FLOCK signal
	TXPLL_JA_FMI	M gain for frequency integral control
	TXPLL_JA_FKI	K gain for frequency integral control
TXPLL_JA_5	TXPLL_JA_PMP1	M gain for Proportional phase comparison loop course setting
	TXPLL_JA_PMP2	M gain for Proportional phase comparison loop fine setting
	TXPLL_JA_PMI1	M gain for Integral phase comparison loop course setting
	TXPLL_JA_PMI2	M gain for Integral phase comparison loop fine setting

Table 2 • **Register Details *(continued)***

SCB REGISTER	DPLL PARAMETER	DESCRIPTION
TXPLL_JA_6	TXPLL_JA_PKP1	K gain for Proportional phase comparison loop course setting
	TXPLL_JA_PKP2	K gain for Proportional phase comparison loop fine setting
	TXPLL_JA_PKI1	K gain for Integral phase comparison loop course setting
	TXPLL_JA_PKI2	K gain for Integral phase comparison loop fine setting
TXPLL_JA_7	TXPLL_JA_DELAYK	Phase comparison loop will settle initially using PKP1 and PKI1 (course settling) parameters, and switch over to PKP2 and PKI2 (fine settling) parameters DELAYK FFB pulses after phase comparison loop takes over (OTDLY + DELAYK FFB pulses after initial frequency lock).
	TXPLL_JA_FDONLY	1'b1: Frequency + Phase control 1'b0: Frequency control only
	TXPLL_JA_ONTARGETOV	1'b1 -> Normal Operation 1'b0 -> Diagnostic mode
	TXPLL_JA_PROGRAM	1'b1 -> Normal Operation 1'b0 -> Diagnostic mode
TXPLL_JA_8	TXPLL_JA_FRAC_PRESET	When PRESET_EN -> 1'b1, FRAC is equal to TXPLL_JA_FRAC_PRESET
	TXPLL_JA_PRESET_EN	Load preset INT and FRAC values into DPLL
	TXPLL_JA_HOLD	Hold output state of DPLL 1'b1 -> Hold, Pll will not phase lock in this state 1'b0 -> Normal operation
TXPLL_JA_9	TXPLL_JA_INT_PRESET	When PRESET_EN -> 1'b1, INT is equal to TXPLL_JA_INT_PRESET
	TXPLL_JA_INT_PD_OUT	Integer bits of phase detector output
TXPLL_JA_10	TXPLL_JA_PHASE_LOCK	1'b1 -> DPLL is in fine tune phase tracking mode 1'b0 -> DPLL is in coarse tune phase tracking mode
	TXPLL_JA_FLOCK	1'b1 -> DPLL has achieved frequency lock 1'b0 -> DPLL has not achieved frequency lock
	TXPLL_JA_FRAC_PD_OUT	Fractional bits of phase detector output

Table 2 • **Register Details *(continued)***

SCB REGISTER	DPLL PARAMETER	DESCRIPTION
TXPLL_JA_RST	TXPLL_JA_RESET	1'b1 -> DPLL reset asserted 1'b0 -> DPLL reset de-asserted
	TXPLL_JA_RESET_FFB_OVERRIDE	1'b0 -> Disables DPLL override signal for feedback clock domain reset signal. 1'b0 -> Enables DPLL override signal for feedback clock domain reset signal.
	TXPLL_JA_RESET_FFB_EXT	1'b0 -> DPLL reset for feedback clock domain is de-asserted 1'b1 -> DPLL reset for feedback clock domain is asserted
	TXPLL_JA_RESET_FIN_OVERRIDE	1'b0 -> Disables DPLL override signal for input clock domain signal 1'b1 -> Enables DPLL override signal for input clock domain reset signal
	TXPLL_JA_RESET_FIN_EXT	Reset for input clock domain when RESET_FIN_OVERRIDE is 1'b1
	TXPLL_JA_RESET_CLKS_OVERRIDE	1'b0 -> Disables DPLL override signal for PLL Sync clock domain reset signal 1'b1 -> Enables DPLL override signal for PLL Sync clock domain reset signal
	TXPLL_JA_RESET_CLKS_EXT	Reset for PLL Sync Clock Domain when RESET_CLKS_OVERRIDE is 1'b1

4.1.3 Hold Output State

When JA_HOLD is set to '1', JAPLL maintains its last value of INT and FRAC until HOLD is set to '0'. This feature allows the Transmit PLL to generate a stable frequency in the event of a link loss or and excessive drift between the transmit and receive clocks.

4.1.4 Locking from INT and FRAC PRESET Settings

The JAPLL supports locking from pre-set INT and FRAC values to speed up lock time. The feature is activated by providing values to INT PRESET and FRAC PRESET and setting PRESET EN = 1'b1. When the loop detects a PRESET EN transition from 1'b0 to 1'b1, it will assign the values of INT PRESET and FRAC PRESET to the frequency detector and re-enter frequency tracking mode and then proceed to lock as normal.

PRESET EN must maintain a value of 1'b1 for at least one FFB clock cycle to guarantee that the preset mode takes effect, however, PRESET EN may be left at a value of 1'b1 due to the fact that the preset logic only activates on a positive edge transition of PRESET EN.

In addition, PRESET EN should be set to 1'b1 only after the values of INT PRESET and FRAC PRESET have been written to ensure no glitch occurs. Specifying a new INT PRESET and FRAC PRESET requires that PRESET EN transition to 1'b0 for at least one FFB clock cycle and then to transition to 1'b1 again.

4.1.5 Exiting Hold State using PRESET Values

PRESET EN may be set to 1'b1 while the PLL is in a hold state, and once HOLD = 1'b0, the PLL will resume locking from the INT PRESET and FRAC PRESET values.

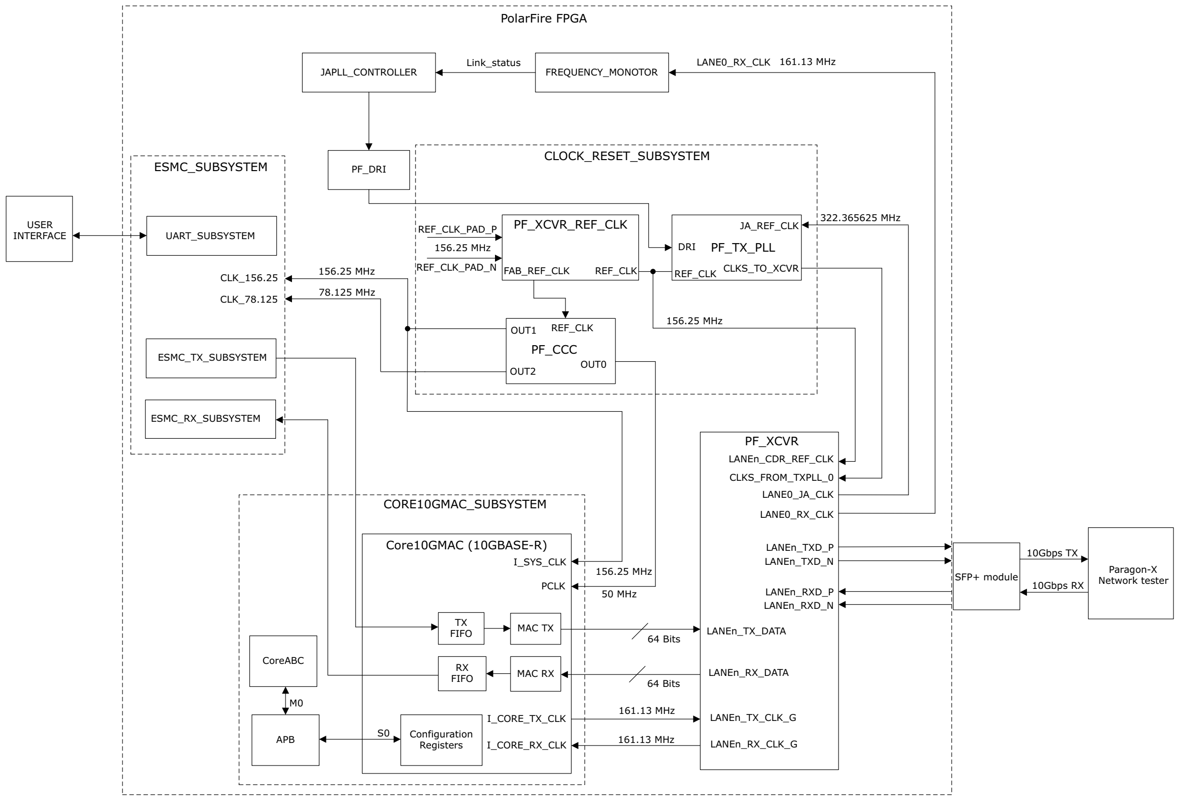
5 Demo Design

The PolarFire 10G SyncE solution receives the Ethernet traffic generated by the Paragon tester and checks for the destination address field in the Ethernet packet header.

When the destination address matches the IEEE 802.3 specified OSSP slow protocol destination address field, the packet is processed, and appropriate response is generated in the form of an ESMC PDU.

The following is the top-level block diagram of the PolarFire 10G Synchronous Ethernet solution featuring ESMC.

Figure 5 • **Top Level Block Diagram of the PolarFire 10G Synchronous Ethernet Solution Featuring ESMC**



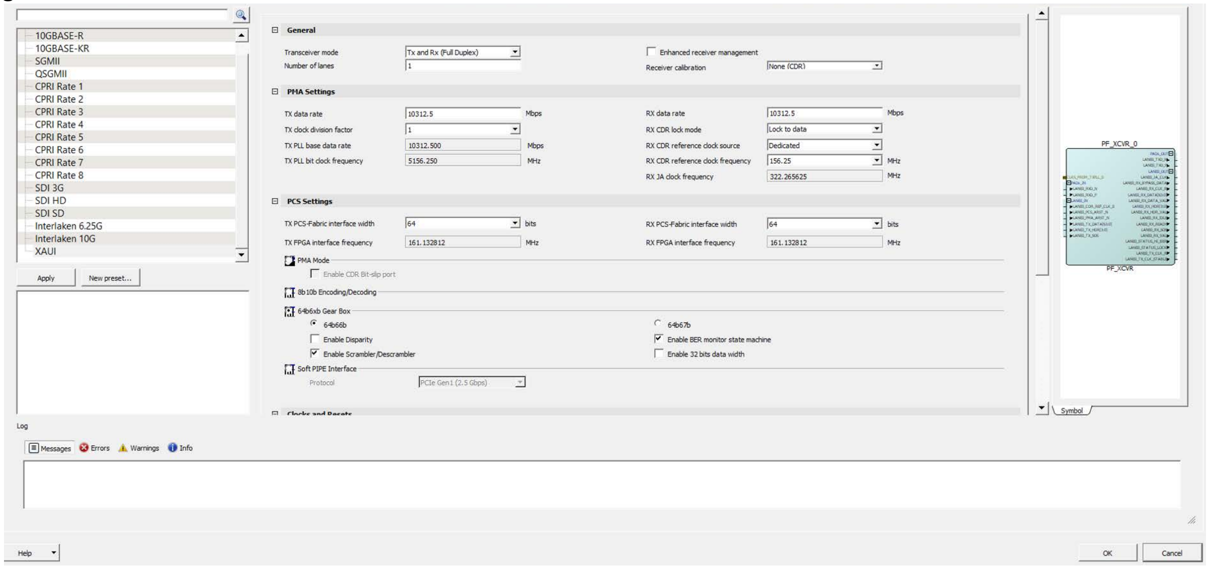
5.1 Design Implementation

The 10G SyncE solution includes the following components:

5.1.1 PF_XCVR

The PolarFire high-speed transceiver (PF_XCVR) is a hard IP block and supports data rates from 250 Mbps to 12.5 Gbps. In this demo, PF_XCVR is configured for the data rate of 10312.5 Mbps. It is configured with a CDR reference clock of 156.25 MHz with Lock to data selected as the CDR lock mode. The PCS of the transceiver is interfaced with CORE10GMAC. It is configured for the 64/66b mode with scrambler/descrambler enabled. The self-synchronizing scrambler, generates sufficient transitions to aid data and clock recovery at the CDR. Figure 6, page 12 shows the transceiver interface configuration.

Figure 6 • **Transceiver Interface**



General

Transceiver mode	Tx and Rx (Full Duplex)		☐ Enhanced receiver management		
Number of lanes	1		Receiver calibration	None (CDR)	

PMA Settings

TX data rate	10312.5	Mbps	RX data rate	10312.5	Mbps
TX clock division factor	1		RX CDR lock mode	Lock to data	
TX PLL base data rate	10312.500	Mbps	RX CDR reference clock source	Dedicated	
TX PLL bit clock frequency	5156.250	MHz	RX CDR reference clock frequency	156.25	MHz
			RX JA clock frequency	322.265625	MHz

PCS Settings

TX PCS-Fabric interface width	64	bits	RX PCS-Fabric interface width	64	bits
TX FPGA interface frequency	161.132812	MHz	RX FPGA interface frequency	161.132812	MHz

PMA Mode
- ☐ Enable CDR Bit-slip port

8b10b Encoding/Decoding

64b6xb Gear Box
- ● 64b66b
- ○ 64b67b
- ☐ Enable Disparity
- ✓ Enable BER monitor state machine
- ✓ Enable Scrambler/Descrambler
- ☐ Enable 32 bits data width

Soft PIPE Interface
- Protocol: PCIe Gen1 (2.5 Gbps)

Clocks and Resets

5.2 Core10GMAC Subsystem

Core10GMAC subsystem consists of the following components.

5.2.1 Core10GMAC

Core10GMAC is configured for 10GBASE-R mode with a core data width of 32 bits. Core data width is the width of the data path connected to the transceiver interface. The system data width is the width of the interface to the user logic, and is configured as 64 bits. (In this demo, the FiFo_wrapper_top module provides this interface.

The Tx and Rx Pause features are disabled, and both the MAC TX FIFO depth and MAC RX FIFO depth are set to 256.

The Core10GMAC IP is configured using the CoreABC soft processor. The Core10GMAC configuration for the demo design is as follows.

Table 3 • **Core10GMAC Configuration**

Register	Address	Offset	Bit	Binary Value
MAC Tx Config Register	(0xA)	0x3	cfg_sys_mac_tx_en	1
		0x4	sys_mac_tx_fcs_ins	1
MAC Rx Config Register	(0xB)	0x0	mac_rx_fcs_remove	1
		0x3	cfg_sys_mac_rx_en	1

For more information about the features and registers of Core10GMAC, refer *Core10GMAC Handbook*.

5.2.2 CoreABC

CoreABC is a configurable, low-gate count controller intended for Advanced Microcontroller Bus Architecture Advanced Peripheral Bus (AMBA APB) based designs. Because this demo design requires only a few registers to be configured, and no dynamic changes are required in the configuration, the CoreABC processor is used in this design. Depending on the application requirements, RISC-V, Cortex-M1, or any other soft processor may be used for configuring the registers.

5.2.3 CoreAPB3

CoreAPB3 is a bus component that provides an AMBA AHB fabric for interconnection between an APB master and up to 16 APB slaves. CoreAPB3 supports a single APB3 master. In this design, CoreAPB3 is used to connect the CoreABC APB master interface to the CORE10GMAC APB slave interface.

5.3 ESMC Susbsystem

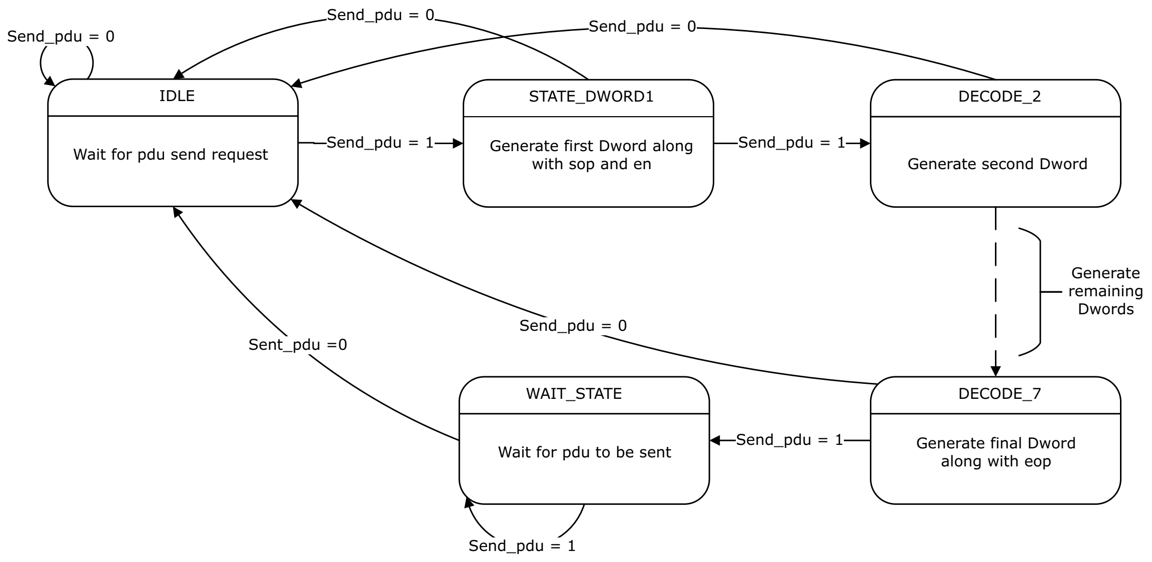
ESMC subsystem processes data received from Core10GMAC and uses the information to control the Jitter Attenuating PLL (JAPLL) to enable/disable SyncE.

5.3.1 ESMC TX Subsystem

ESMC TX subsystem consists of the following modules:

- pdu_generator: This module handles the following features:
  - pdu_generate_request_servicing: This module services the request received from the pdu_checker and also generates an acknowledgment signal to indicate the successful transmission of PDU.
  - pdu_packet_framer: This module frames 58-byte packet based on a different type of PDU request received. Currently, the framer accepts a request to generate 10 different PDU's.
  - pdu_counter_logic: Counter logic to keep a log of how many PDU's have been sent out. Current logic can keep track of all the different types of PDU's sent.
  - pdu_sent_max_threshold: Logic to limit the number of PDU's that can be sent using a 1 second timer.
- fifo_mgmt: FIFO management module is used to efficiently manage the clock domain crossing between a PDU generator (slow clock domain) and core10GMAC (fast clock domain). The module also consists of a controller that manages the writes and reads for a group of 10 FIFO's.
  - FIFO controller reads from FIFO when i_sys_mac_tx_fifo_af signal is '0'.
  - The writes to the FIFO's are performed by pdu_generator when a PDU send request is generated from the Rx subsystem.

Figure 7 • **PDU Generator Finite State Machine**



PDU generator finite state machine traverses through the following states when any one of the send PDU requests are received:

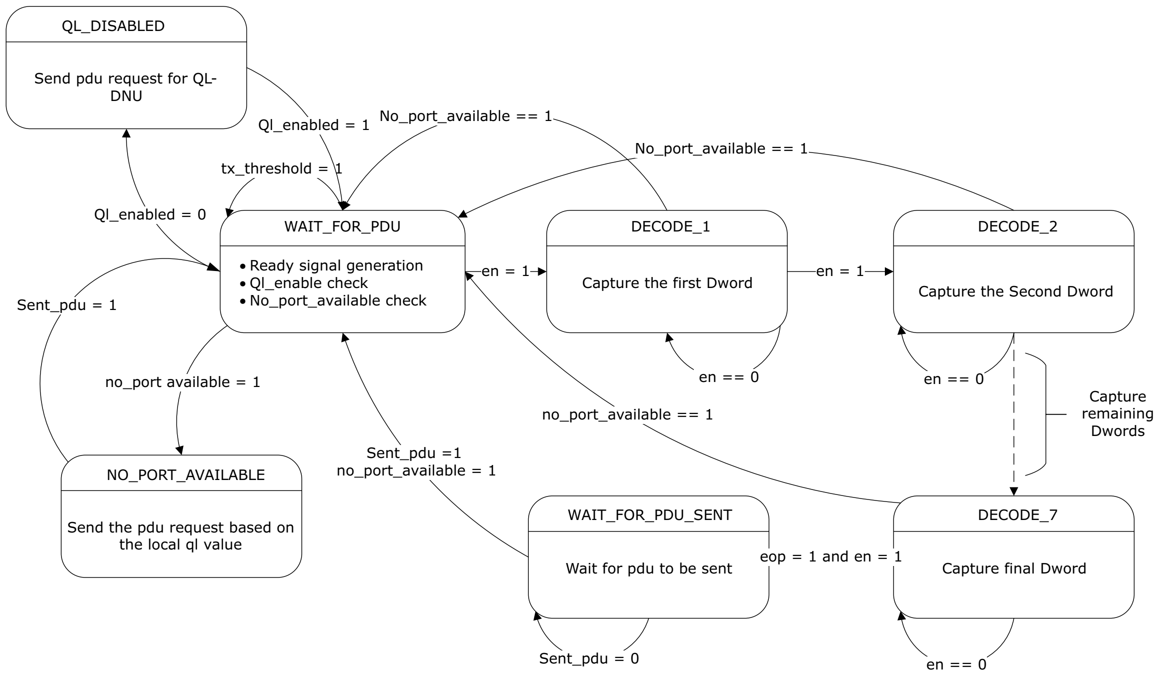
- IDLE: Check for any of the PDU send requests to be '1' and then traverse to STATE_DWORD1.
- STATE_DWORD1 to STATE_DWORD8: The framer accesses the predefined data word from the defined file based on the request sent from pdu_checker and also traverses to the next state.
- WAIT_STATE: Logic checks for any of the PDU send requests to be '1' and then traverse to the IDLE state.

5.3.2 ESMC RX Subsystem

ESMC RX subsystem consists of the following modules:

- pdu_parser: It implements a FIFO management system to detects and store the ESMC PDU by checking the Destination Address (48'h0180_C200_0002).
  - Header decoder
  - Storing pdu
- pdu_checker:
  - pdu_info_extraction: A finite state machine decodes and stores the data fields from the received PDU.
  - ql_priority_selection_algo: Logic to compare the local QL value with the decoded QL value and selects the QL value which has higher priority (better clock quality).
  - pdu_generation_request: When ql_enabled is '0', a DNU info pdu send request is sent to the pdu_generator, whereas when ql_enabled is '1' pdu_checker module sends a pdu generation request to pdu_generator based on the result of QL selection logic.
  - Port availability information

Figure 8 • **PDU Checker Finite State Machine**



PDU checker finite state machine traverses through the following states when any one of the send PDU requests are received:

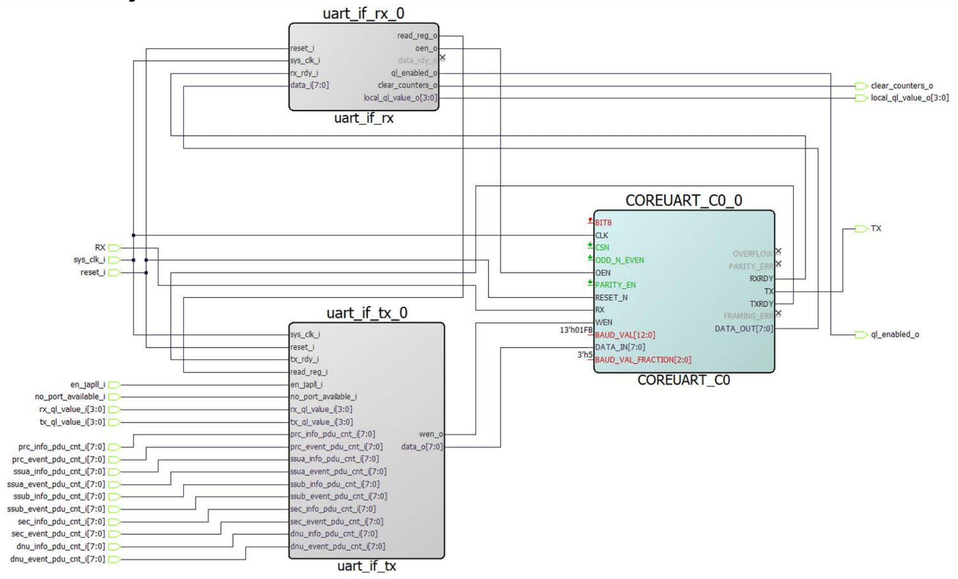
- Enabling Jitter Cleaner:
  - After reset is de-asserted enable_japll signal is '1', when pdu_data_ready signal is '1' pdu_checker will de-assert the JAPLL by setting enable_japll to '0'.
  - This sequence is performed only once when ql_enable is '1' to show the actual working of the jitter cleaner PLL.
  - Enable_japll will remain '0', until there is any change in the input QL value.
  - If the QL selection logic detects a change in the input QL value, enable_japll becomes '1' for the transmitter to lock to the new clock reference which has a better clock quality than the local reference clock.

5.3.3 UART Subsystem

The UART subsystem implements the fabric UART logic to interface with GUI, which is used as a user interface to control and display status of the running design. The UART subsystem has three sub-modules:

- UART RX Interface
- UART TX Interface
- CORE UART

Figure 9 • **UART Subsystem**



5.3.3.1 UART RX Interface

The UART RX interface finite state machine receives the write request and write data from CORE_UART and provides the following data to the esmc_tx and esmc_rx subsystems.

- QL enable/disable: Determines whether ESMC IP operates in synchronous mode or non-synchronous mode.
  - When ql_enabled is '1', design is in synchronous mode and processes the ESMC PDU's.
  - When ql_enabled is '0', design is in non-synchronous mode and does not process ESMC PDU's.
- Local QL value: Local QL value of the EEC equipment to be used after reset.
- Clear counter: Clear counter signal clears all the PDU counters.

5.3.3.2 UART TX Interface

The UART TX interface finite state machine receives the following design status and provides the status to CORE_UART upon receiving read request.

- QL enable/disable
- Current local QL value
- PRC info pdu count
- PRC event pdu count
- SSUA info pdu count
- SSUA event pdu count
- SSUB info pdu count
- SSUB event pdu count
- SEC info pdu count
- SEC event pdu count
- DNU info pdu count
- DNU event pdu count

5.3.3.3 Core UART

CORE UART configures at baud value 91600 bps and a 78.125 MHz clock is provided as reference.

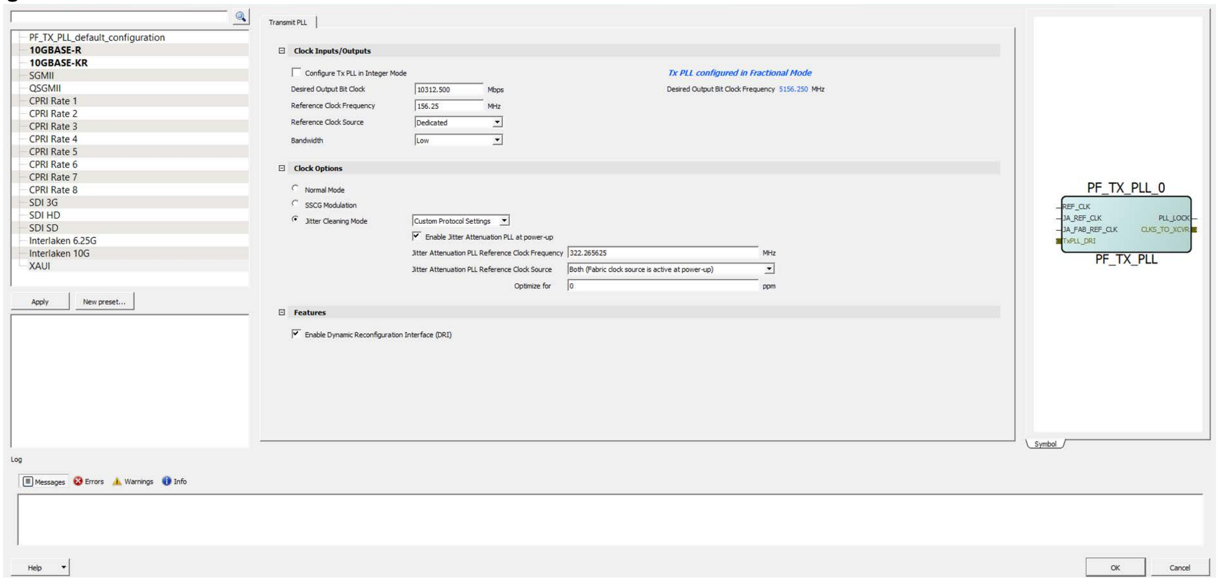
5.4 Clock Reset Subsystem

Clock Reset Subsystem generates the necessary clocks and resets for PF_XCVR, Core10GMAC, and ESMC_TOP.

5.4.1 Transmit PLL

The PolarFire Transmit PLL (PF_TX_PLL) is a hard IP block that provides a bit clock and a reference clock to the transceiver block. The transmit PLL is configured with a reference clock of 156.25 MHz and generates an output clock of 10312.5 Mbps.

Figure 10 • **Transmit PLL**



Clock Inputs/Outputs

Configure Tx PLL in Integer Mode

Tx PLL configured in Fractional Mode

Desired Output Bit Clock 10312.500 Mbps

Desired Output Bit Clock Frequency 5156.250 MHz

Reference Clock Frequency 156.25 MHz

Reference Clock Source Dedicated

Bandwidth Low

Clock Options

Normal Mode

SSCG Modulation

[✓ Jitter Cleaning Mode] Custom Protocol Settings

[✓ Enable Jitter Attenuation PLL at power-up]

Jitter Attenuation PLL Reference Clock Frequency 322.265625 MHz

Jitter Attenuation PLL Reference Clock Source Both (Fabric clock source is active at power-up)

Optimize for 0 ppm

Features

[✓ Enable Dynamic Reconfiguration Interface (DRI)]

5.4.2 Transceiver Reference Clock

The Transceiver Reference Clock (PF_XCVR_REF_CLK) is a hard IP block that provides a reference clock (REF_CLK) of 156.25 MHz to the transmit PLL and a fabric reference clock (FAB_REF_CLK) which is provided as input to the Clock Conditioning Circuit (CCC) to generate the pclk (for configuration) and I_SYS_CLK of the CORE10GMAC.

5.4.3 PF_POWER_INIT

The PF_POWER_INIT block ensures the device is powered up systematically. The process of powering up the device includes three steps:

- Power-on reset
- Programmed device boot
- Design initialization

During design initialization, the transceiver configuration is initialized using the data stored in the non-volatile memory. The output of the PF_POWER_INIT block is ANDed with the resets used in the design to reset entire logic.

5.4.4 PF_CCC

The PolarFire CCC block takes an input clock of 156.25 MHz from the FAB_REF_CLK signal (output of PF_XCVR_REF_CLK) and generates a 50 MHz clock at OUT0 and 156.25 at OUT1. The OUT0 port of the CCC is used for the configuration and OUT1 port of the CCC are used for the user logic in the design.

5.4.5 Clocking Structure

In the reference design there are 3 clock domains RX_CLK (161.13 MHz), TX_CLK (161.13 MHz), and CLK_156pt25 (156.25 MHz). For more information about the top level block diagram, refer Figure 5, page 11.

Table 4 • **Clocking Structure**

Clock Name	Frequency	Instance Name
clk_156pt25 MHz	156.25	esmc_tx_subsystem_0
		esmc_rx_subsystem_0
clk_78pt125 MHz	78.125	esmc_tx_subsystem_0
		esmc_rx_subsystem_0
		UART_IF_0
TX_CLK_R	161.13	CORE10GMAC0_0
RX_CLK_R	161.13	CORE10GMAC0_0
		Frequency_change_monitor_0
clk_156pt25 MHz	156.25	CORE10GMAC0_0
clk_50 MHz	50	CORE10GMAC_SUBSYSTEM
		Japll_controller_0
		PF_DRI_0
		COREABC0_0

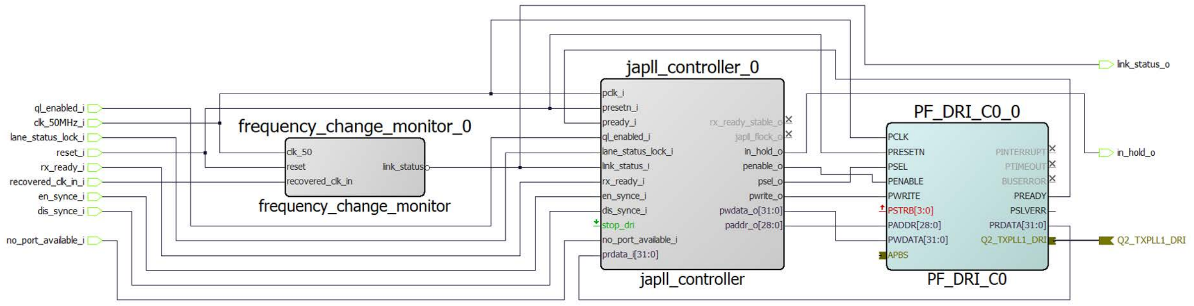
5.5 JAPLL Controller Subsystem

5.5.1 Frequency Change Monitor

- The frequency change monitor takes the recovered clock as input and monitors the frequency change with respect to a stable clock.
- The logic checks the frequency of the clock at fixed intervals and determines the stability of the clock. If the deviation in the frequency exceeds the pre-determined limits, a link loss signal is asserted, and the japll_controller is informed of a break link condition.

5.5.2 JAPLL Controller

Figure 11 • **JAPLL Controller**



japll_controller implements a DRI master interface to interact with the JAPLL via a PF_DRI core.

Both fabric logic and the PF_DRI core operates on the same frequency of 50 MHz. Enabling and disabling of syncE is done based on the requests received from the ESMC IP.

Initially, after powerup, JAPLL will be locked to the external auxiliary clock. When the en_synce command is received from the ESMC IP JAPLL Controller will execute a sequence of steps for the JAPLL to switch its input to the cdr recovered clock. JAPLL generates the INT and FRAC values which are derived based on the cdr recovered clock. TX clock synchronizes with the recovered clock to show that synchronous ethernet is enabled. For more information, refer to Hitless Clock Switching: Enable SyncE, page 22.

When the dis_synce command is received from ESMC IP, JAPLL input is switched back to the local auxiliary clock. JAPLL generates the INT and FRAC values based on the local auxiliary clock. The TX clock synchronizes with the local auxiliary clock to prove that synchronous ethernet is disabled. For more information, refer to Hitless Clock Switching: Disable SyncE, page 23.

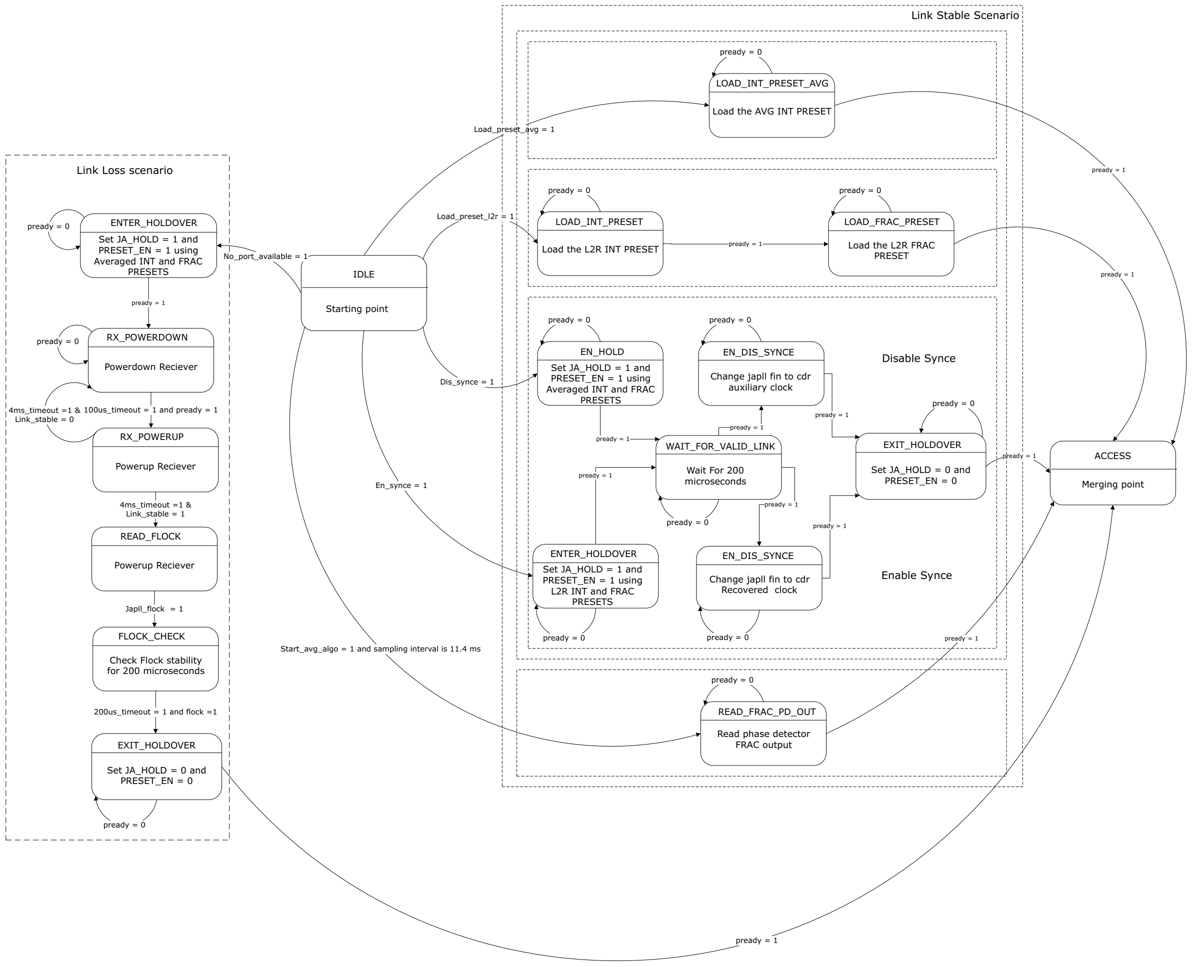
Japll_apb3_master performs the following register write operations:

Table 5 • **Register Write Operation**

Function	Register	Description
Write	EXT_PLL_JA_8	Enabling/disabling JA_HOLD and PRESET_EN
Write	EXT_PLL_JA_8	Load FRAC PRESET value
Write	EXTPLL_JA_9	Load INT_PRESET value
Write	DES_RSTPD	For RX power-down and powerup
Read	EXTPLL_JA_10	Read JAPLL Flock and Plock values
Read	EXTPLL_JA_9	Read INT_PD_OUT value
Read	EXTPLL_JA_10	Read FRAC_PD_OUT value

JAPLL Controller handles two different scenarios in this reference design.

Figure 12 • **Reference Design**

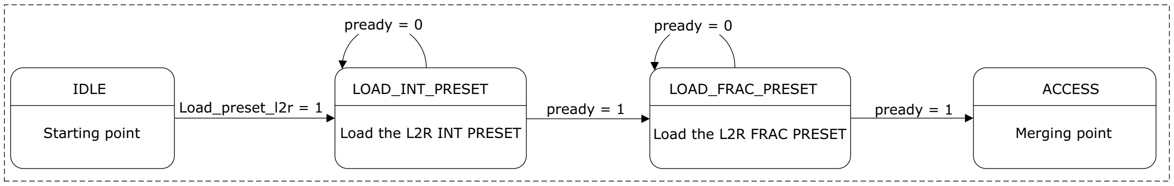

Link Stable Scenario
pready = 0
LOAD_INT_PRESET_AVG
Load the AVG INT PRESET
Load_preset_avg = 1
pready = 1
Link Loss scenario
pready = 0
ENTER_HOLDOVER
Set JA_HOLD = 1 and PRESET_EN = 1 using Averaged INT and FRAC PRESETS
No_port_available = 1
IDLE
Starting point
Load_preset_l2r = 1
LOAD_INT_PRESET
Load the L2R INT PRESET
pready = 1
LOAD_FRAC_PRESET
Load the L2R FRAC PRESET
pready = 1
pready = 1
RX_POWERDOWN
Powerdown Reciever
4ms_timeout =1 & Link_stable = 0
100us_timeout = 1 and pready = 1
RX_POWERUP
Powerup Reciever
4ms_timeout =1 & Link_stable = 1
READ_FLOCK
Powerup Reciever
Japll_flock = 1
FLOCK_CHECK
Check Flock stability for 200 microseconds
200us_timeout = 1 and flock =1
EXIT_HOLDOVER
Set JA_HOLD = 0 and PRESET_EN = 0
Dis_synce = 1
EN_HOLD
Set JA_HOLD = 1 and PRESET_EN = 1 using Averaged INT and FRAC PRESETS
EN_DIS_SYNCE
Change japll fin to cdr auxiliary clock
Disable Synce
WAIT_FOR_VALID_LINK
Wait For 200 microseconds
EXIT_HOLDOVER
Set JA_HOLD = 0 and PRESET_EN = 0
ACCESS
Merging point
En_synce = 1
ENTER_HOLDOVER
Set JA_HOLD = 1 and PRESET_EN = 1 using L2R INT and FRAC PRESETS
EN_DIS_SYNCE
Change japll fin to cdr Recovered clock
Enable Synce
Start_avg_algo = 1 and sampling interval is 11.4 ms
READ_FRAC_PD_OUT
Read phase detector FRAC output
pready = 1

- Stop the DRI access on request.
- Load the JAPLL with the default INT and FRAC preset values.
- Hitless Clock Switching: Enable and Disable SyncE.
- Read the JAPLL phase detector output every 11.4ms.
- Implementing the averaging algorithm.
- Load the JAPLL with the computed averaged INT preset value.
- Handling the break-link and make-link conditions.

Each scenario is explained briefly in the following sections:

5.5.2.1 Load the JAPLL with the Default INT and FRAC Preset Values

Figure 13 • **Load the JAPLL with the Default INT and FRAC Preset Values**

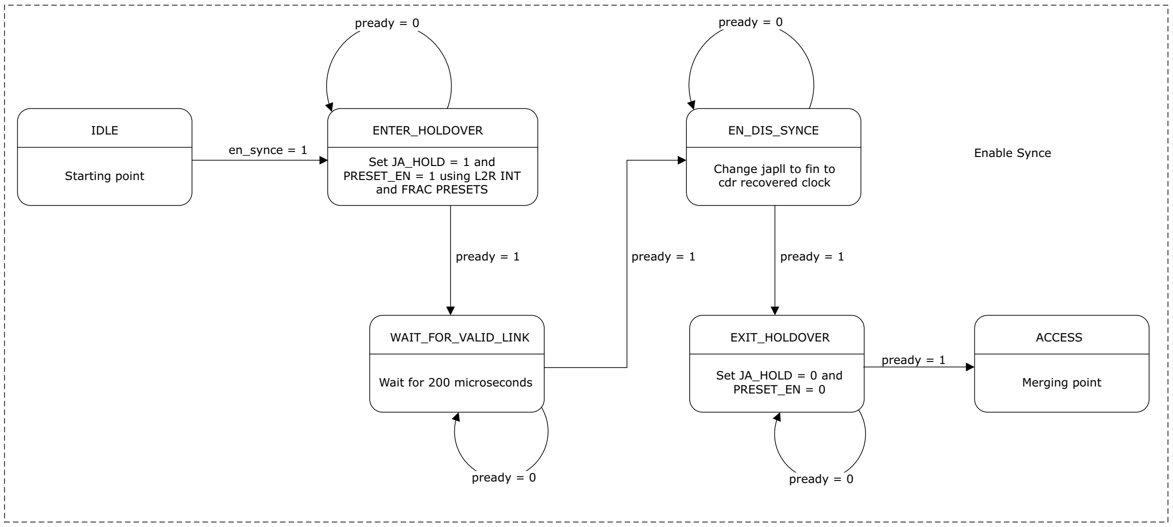


The first task carried out by the JAPLL Controller finite state machine after reset is to load the JAPLL INT and PRESET values with the default's integer and fractional PLL divider settings with respect to the local auxiliary clock frequency. Loading the JAPLL with new INT and FRAC preset values speeds up the JAPLL locking to the reference clock.

5.5.2.2 Hitless Clock Switching: Enable SyncE

The following figure shows the Hitless clock switching for Enable SyncE.

Figure 14 • **Hitless Clock Switching: Enable SyncE**



Whenever the ESMC receiver subsystem receives a new QL-value, the selection algorithm compares the incoming QL-value with the already existing local QL-value and makes the appropriate decision.

If the incoming QL-value has higher priority than the existing local QL-value, then the selection algorithm will notify the JAPLL Controller to take necessary action such that the JAPLL will lock to the better-quality recovered clock and enable the SyncE.

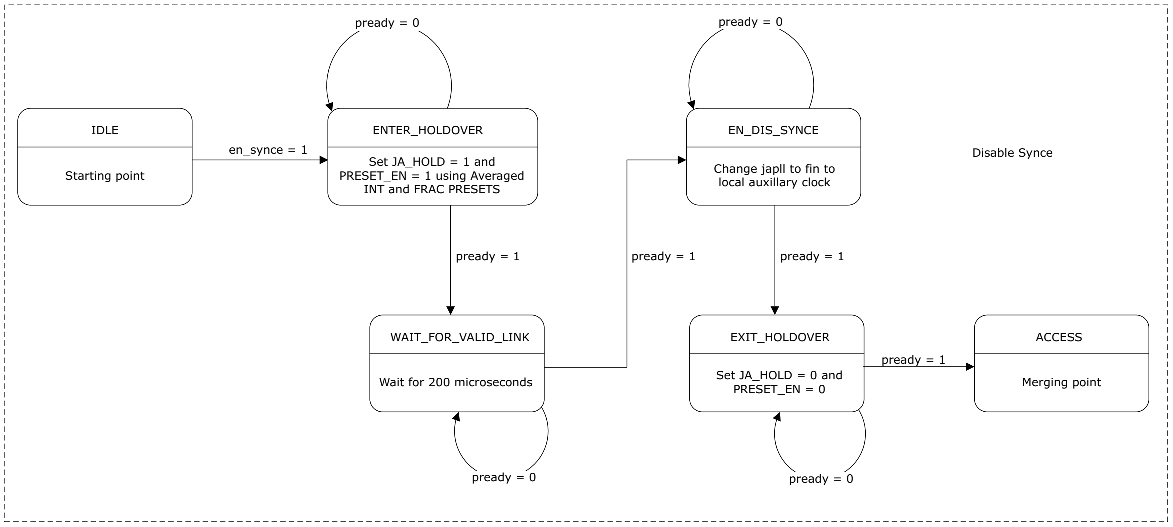
The JAPLL Controller will perform the following sequence of steps to enable the SyncE.

- ENTER_HOLDOVER:
  - Enable JALL hold by setting JA_HOLD to '1' and PRESET_EN to '1' to use the default INT and FRAC preset values.
- WAIT_FOR_VALID_LINK:
  - Wait for 200 microseconds to make sure the JAPLL enters into holdover.
- EN_DIS_SYNCE:
  - Switch the FIN of JAPLL to the cdr recovered clock.
  - Now JAPLL starts locking to the recovered clock.
- EXIT_HOLDOVER:
  - Disable JAPLL hold by setting JA_HOLD to '0' and PRESET_EN to '0'.
- Start the averaging algorithm: For more information, refer to Implement the Averaging Algorithm, page 24.

5.5.2.3 Hitless Clock Switching: Disable SyncE

The following figure shows the Hitless clock switching for Disable SyncE.

Figure 15 • **Hitless Clock Switching: Disable SyncE**



Whenever the ESMC receiver subsystem receives a new QL-value, the selection algorithm compares the incoming QL-value with the already existing local QL-value and makes the appropriate decision.

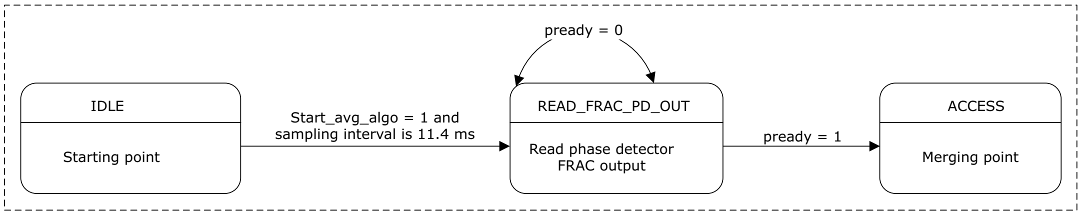
If the incoming QL-value has a lower priority than the existing local QL-value, then the selection algorithm will notify the JAPLL Controller to take necessary action such that the JAPLL will lock to the better-quality local auxiliary clock and disable the SyncE.

The JAPLL Controller will perform the following sequence of steps to disable the SyncE.

- ENTER_HOLDOVER:
  - Enable JALL hold by setting JA_HOLD to ‘1’ and PRESET_EN to ‘1’ to use the Averaged INT and FRAC preset values.
- WAIT_FOR_VALID_LINK:
  - Wait for 200 microseconds to make sure the JAPLL enters holdover.
- EN_DIS_SYNCE:
  - Switch the FIN of JAPLL to the cdr recovered clock.
  - Now JAPLL starts locking to the local auxiliary clock.
- EXIT_HOLDOVER:
  - Disable JAPLL hold by setting JA_HOLD to ‘0’ and PRESET_EN to ‘0’.
- Stop the averaging algorithm: For more information, refer to Implement the Averaging Algorithm, page 24.

5.5.2.4 Read the JAPLL Phase Detector Output every 11.4ms

Figure 16 • **Read the JAPLL Phase Detector Output**


pready = 0
IDLE
Starting point
Start_avg_algo = 1 and sampling interval is 11.4 ms
READ_FRAC_PD_OUT
Read phase detector FRAC output
pready = 1
ACCESS
Merging point

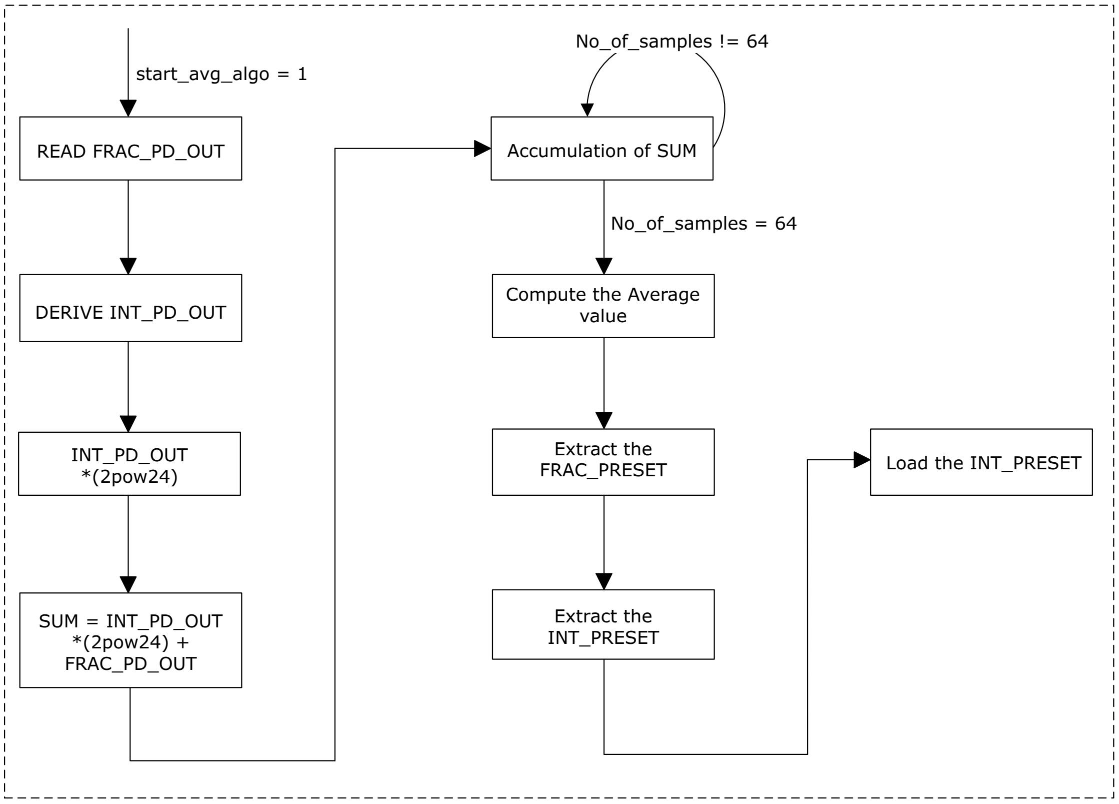
For the JAPLL to be G.8262 compliant, the loop bandwidth has to be the minimum so that the averaging frequency can be within the Wander cut-off frequency limits, which is 1-10 Hz.

The sampling rate is set to 11.4 milliseconds, and the total number of samples to 64, the averaging frequency will be 1.36 Hz which falls in the range of 1-10 Hz.

5.5.2.5 Implement the Averaging Algorithm

The following figure shows the sampling algorithm block diagram.

Figure 17 • **Averaging Algorithm**



When the start_avg_algo_i signal is asserted, the japll_controller finite state machine executes the averaging algorithm by continuously reading the INT_PD_OUT and FRAC_PD_OUT values of the Phase detector and computes the average values till it reaches the user-specified limit.

The procedure is as follows:

3. Read INT_PD_OUT and FRAC_PD_OUT.
4. Accumulate the INT_PD_OUT and FRAC_PD_OUT.
5. Repeat the preceding steps till the count reaches AVG_COUNT specified by the user.
6. After AVG_COUNT iterations, compute the average value.
7. Extract the INT_PRESET and FRAC_PRESET from the computed average value.
8. Load the JAPLL INT preset using the new computed INT_PRESET.

In the event of a clock switching or link loss, JAPLL is programmed to use the computed INT_PRESET and FRAC_PRESET when entering in to hold state using the PRESERT_EN function. During Hold state, the phase detector is shut down, and the outputs of JAPLL are held to their last values.

Averaging algorithm forces the JAPLL outputs to be loaded with the computed INT_PRESET and FRAC_PRESET values in order to make sure the Transmit PLL receives the INT and FRAC values which are closer to the actual and do not exceed the +/- 7.5 ppm requirement of syncE.

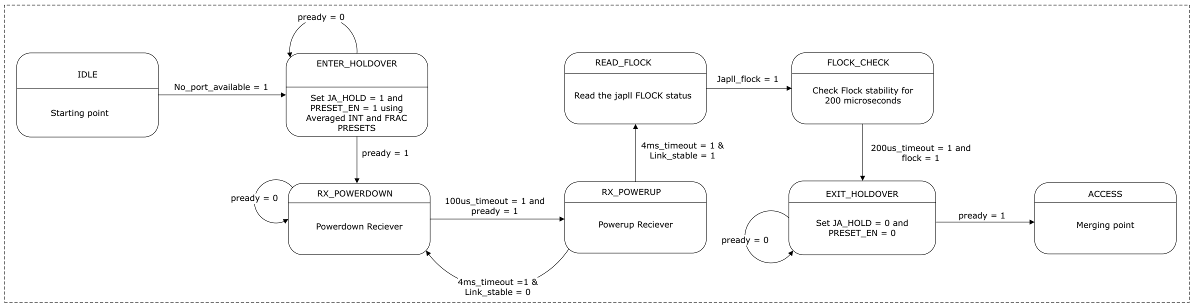
Averaging Algorithm starts when start_avg_algo signal becomes ‘1’ and stops for the following cases:

- When start_avg_algo signal becomes ‘0’ (happens during the disable syncE process)
- When there is a break-link situation

5.5.2.6 Handling the Break-link and Make-link conditions

The following figure show the break-link and make-link conditions.

Figure 18 • **Break-link and Make-link Conditions**


pready = 0
IDLE
Starting point
No_port_available = 1
ENTER_HOLDOVER
Set JA_HOLD = 1 and PRESET_EN = 1 using Averaged INT and FRAC PRESETS
pready = 1
pready = 0
RX_POWERDOWN
Powerdown Reciever
100us_timeout = 1 and pready = 1
4ms_timeout =1 & Link_stable = 0
RX_POWERUP
Powerup Reciever
4ms_timeout = 1 & Link_stable = 1
READ_FLOCK
Read the japll FLOCK status
Japll_flock = 1
FLOCK_CHECK
Check Flock stability for 200 microseconds
200us_timeout = 1 and flock = 1
EXIT_HOLDOVER
Set JA_HOLD = 0 and PRESET_EN = 0
pready = 0
pready = 1
ACCESS
Merging point

1. When the frequency monitor logic detects the deviation in the recovered clock frequency, it asserts the link status signal indicating the break-link condition.
2. ESMC RX SUBSYSTEM will generate a no_port_available signal to the JAPLL Controller.
3. ENTER_HOLDOVER: When no_port_available = 1, Enable JALL hold by setting JA_HOLD to ‘1’ and PRESET_EN to ‘1’ to use the previously loaded preset values and proceed to RX_POWERDOWN state.
4. RX_POWERDOWN: To check the make link condition, FSM will periodically check for the cable re-insert sequence by powering up and shutting down the receiver multiple times.

   Wait for 100 microseconds in RX_POWERDOWN state and proceed to RX_POWERUP state.

5. RX_POWERUP: Wait for four milliseconds to check the stability of the following signals.

- rx_ready from transceiver
- lane_status_lock from transceiver
- link status from frequency change monitor

   If all the preceding signals are stable at the end of 4milliseconds time, then FSM will proceed to READ_FLOCK state else return to the RX_POWEDOWN state.

6. READ_FLOCK: After the make-link condition is successfully verified, the JAPLL FLOCK signal is verified for its stability.
7. FLOCK_CHECK: Check the FLOCK stability for 200 microseconds and if the signal is stable, then proceed to EXIT_HOLDOVER state else return to the READ_FLOCK state.
8. EXIT_HOLDOVER: Disable the JAPLL hold by setting JA_HOLD to ‘0’ and PRESET_EN to ‘0’.

Table 6 • **Port Description**

Port Name	Direction	Description
LANE0_RXD_N	IN	ESMC Receiver inverted input
LANE0_RXD_P	IN	ESMC Receiver non-inverted input
REF_CLK_PAD_N	IN	Inverted reference clock obtained from on-board 156.25 MHz oscillator
REF_CLK_PAD_P	IN	Non-Inverted reference clock obtained from on-board 156.25 MHz oscillator
UART_RX_IF_I	IN	UART receiver interface
reset_i	IN	Asynchronous reset
LANE0_TXD_N	OUT	ESMC Transmitter inverted input
LANE0_TXD_P	OUT	ESMC Transmitter non-inverted input
UART_TX_IF_I	OUT	UART Transmitter interface

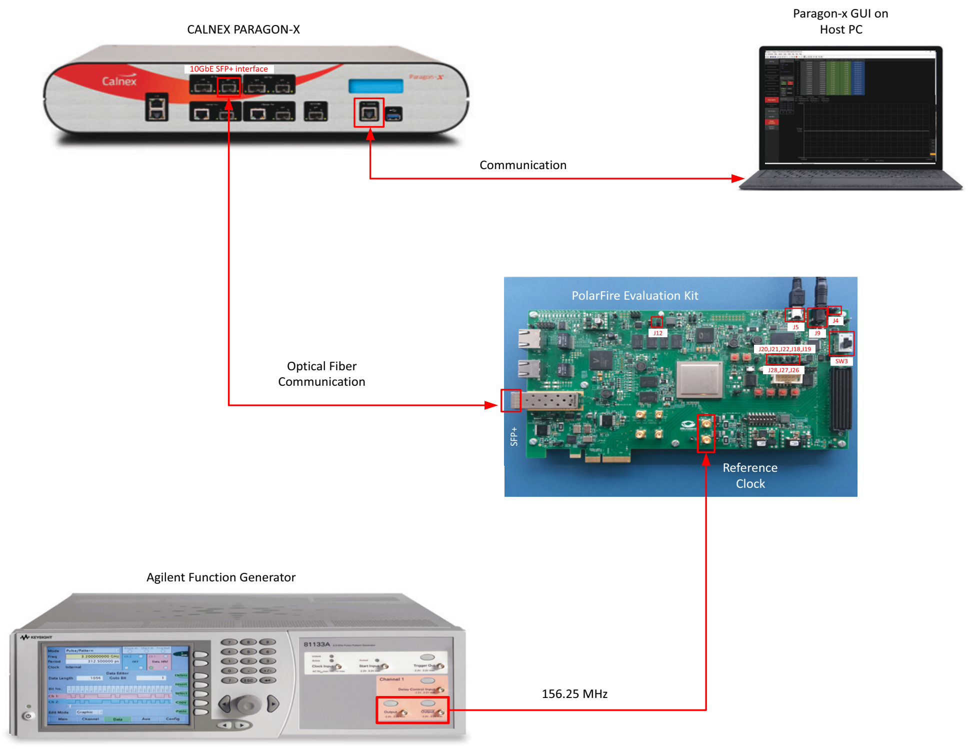
6 Validation Setup

This chapter describes how to install and use the GUI to run the ESMC demo.

The following are the prerequisites:

1. The PolarFire Evaluation board is connected
2. The PolarFire FPGA is programmed with the ESMC design

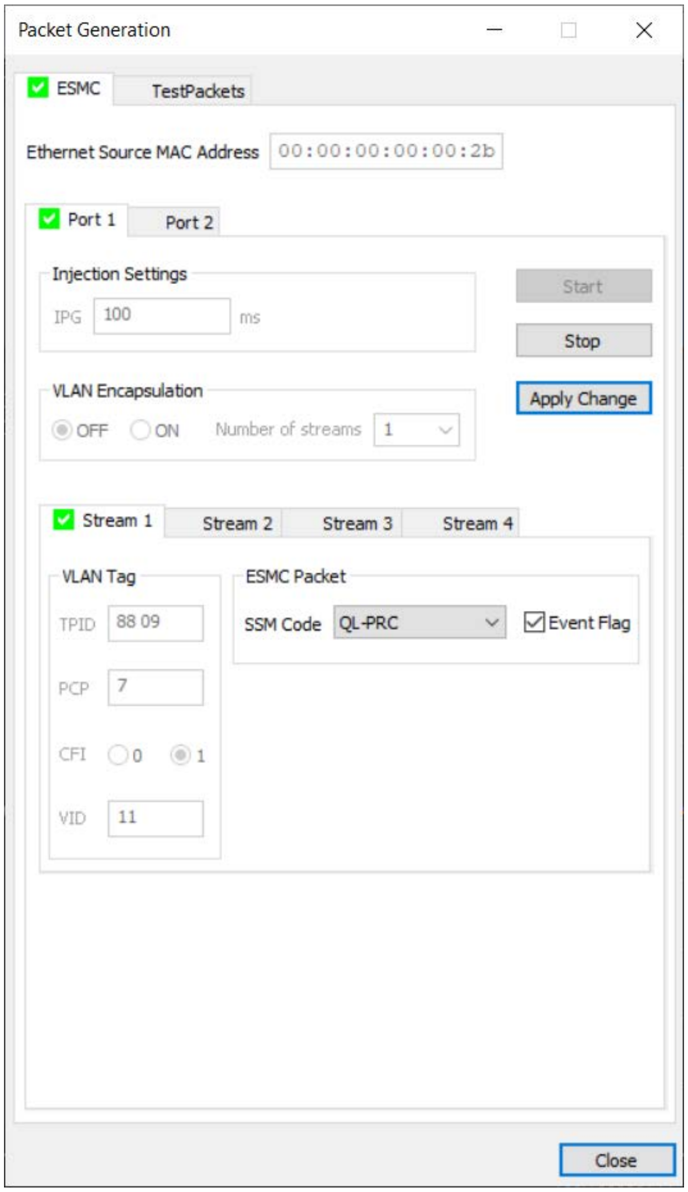
Figure 19 • **Validation Setup Diagram**



To run the ESMC demo:

1. Extract the contents of the `mpf_dg0898_df.zip` file.
2. After programming the design into the PolarFire Evaluation board, open the **SmartDebug** tool from Libero and execute the `japll_params.tcl` from the `mpf_dg0898_df\Libero_Design_Files\TCL_Scripts` folder.
3. Double-click the `setup.exe` file from the `mpf_dg0898_df\ESMC_demo_GUI` folder.
4. Follow the instructions displayed on the installation wizard. After successful installation, ESMC_Demo_GUI appears on the Start menu of the host PC desktop.
5. The reference design is validated using the Paragon-X tester. Configure the paragon-X tester to transmit 10G ethernet data including ESMC PDU's and start the traffic on port 1.
6. Open Packet Generation tab and set the **IPG** to 100 ms and select **SSM Code** as QL-PRC.

Figure 20 • **Configuring Paragon-X Tester**



Ethernet Source MAC Address: 00:00:00:00:00:2b

IPG: 100 ms

VLAN Encapsulation: [✓ OFF] ON; Number of streams: 1

TPID: 88 09
PCP: 7
CFI: 0 [✓ 1]
VID: 11

SSM Code: QL-PRC; [✓ Event Flag]

7. From the **Start** menu, click **ESMC_Demo_GUI** to start the GUI application.

The GUI detects the COM port number and automatically connects to the PolarFire Evaluation board, as shown in Figure 22, page 29. Port numbers may vary.

6.1 Running the Demo

Follow these steps to program the PolarFire device.

1. Ensure that the jumper settings on the board are as listed in the following table

Table 7 • **Jumper Settings for PolarFire Evaluation Board**

Jumper	Description
J18, J19, J20, J21, and J22	Short pin 2 and 3 for programming the PolarFire FPGA through FTDI
J28	Short pin 1 and 2 for programming through the on-board FlashPro Express
J26	Short pin 1 and 2 for programming through the FTDI SPI
J27	Short pin 1 and 2 for programming through the FTDI SPI
J39	Short pin 1 and 2 for enabling the TX
J4	Short pin 1 and 2 for manual power switching using SW3

2. Connect the power supply cable to the **J9** connector on the board.
3. Connect the host PC to the **J5** connector (FTDI port) on the board using a USB cable.
4. Power on the board using the **SW3** slide switch.

The following figure shows the validation setup diagram for programming the device and running the reference design.

Figure 21 • **GUI Before Connecting**

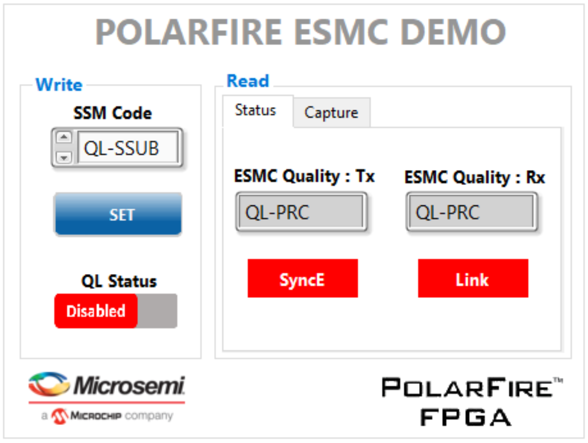
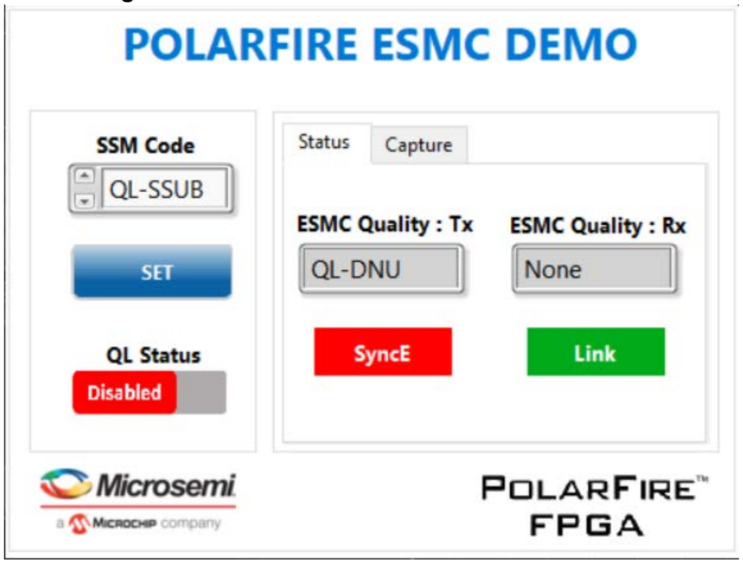
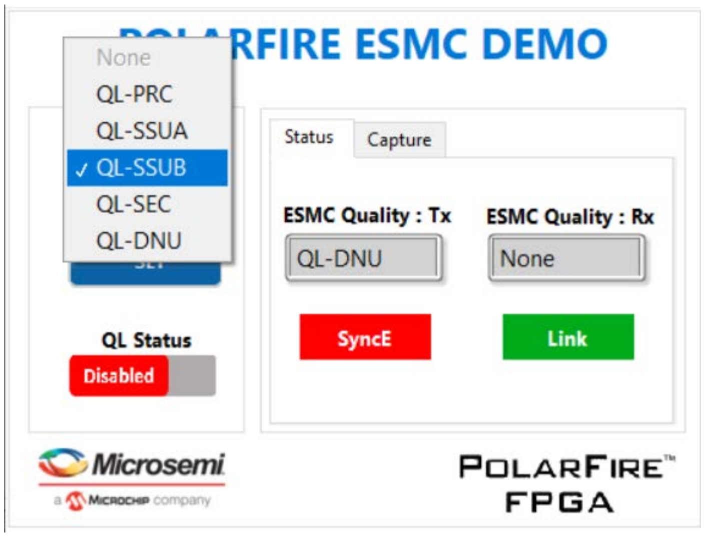


Figure 22 • **GUI After Connecting**



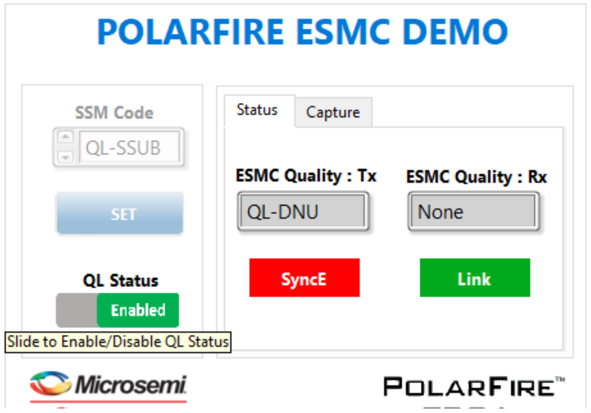
GUI shows the connection has been established by changing the GUI title to Blue color. The Link button turns **Green** to indicate that the link has been established between PolarFire Evaluation Kit and paragon-x tester.

Figure 23 • **PolarFire ESMC Demo1**



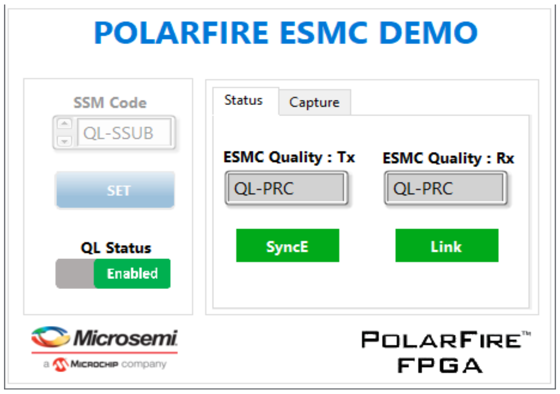
Selecting the **SSM code:** Select any one SSM code from the drop down list and click **SET** button to configure the ESMC IP with QL value.

Figure 24 • **PolarFire ESMC Demo2**



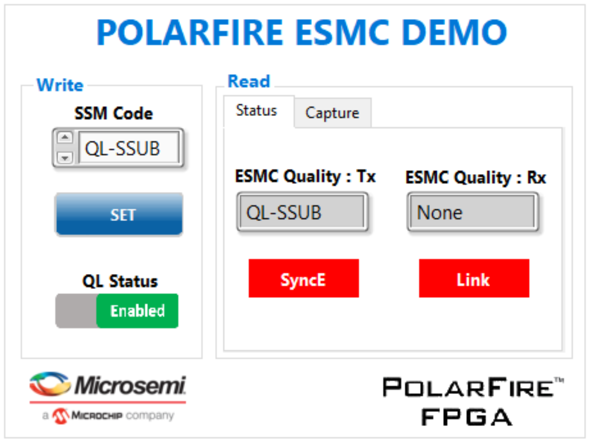
QL Status Enabled: Slide the QL status button to the right to enable the QL mode for the ESMC IP to start accepting the ESMC PDU's.

Figure 25 • **PolarFire ESMC Demo3**



SyncE enabled: SyncE button turns **Green** when the ESMC IP has locked to the external clock source as the other node has better clock quality.

Figure 26 • **PolarFire ESMC Demo4**



Link lost: When there is no link fault the Link button remains Green and turns Red in case of a link fault.

6.2 Validating the G.8262 Short term Phase Transient Response

The device is forced into holdover for 15 seconds by manually disconnecting the optical cable form paragon tester to the PF_Evalkit. For more information, refer to Handling the Break-link and Make-link conditions, page 25.

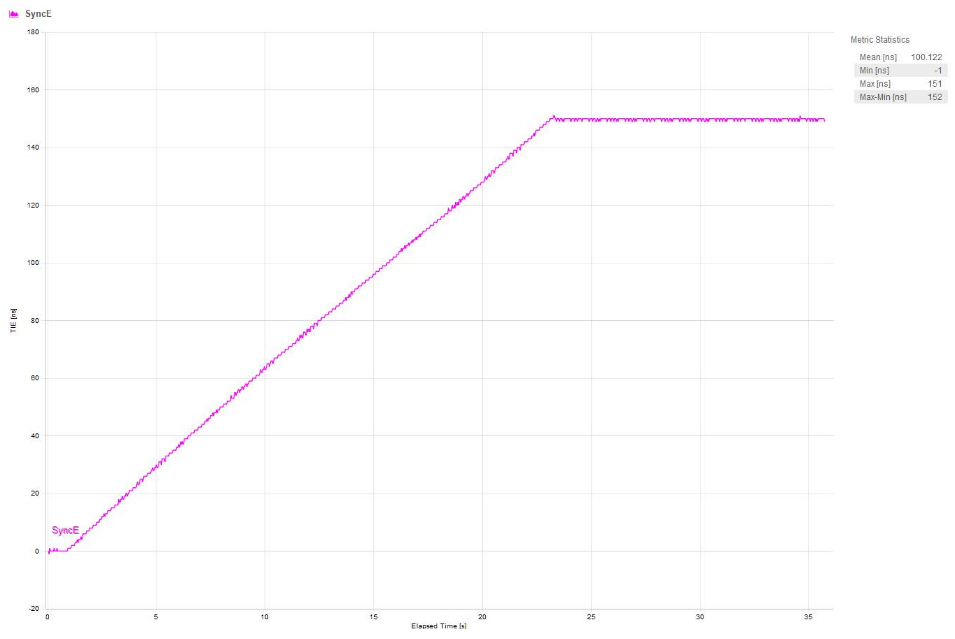
6.2.1 Results

According to G.8262 short term phase transient response for option 1 the initial phase error must not exceed 120ns with in a duration of 16 milliseconds from the event of link loss. The maximum phase error must not exceed 1000ns at the end of 15 seconds window.

The following TIE and MTIE graphs are captured for the preceding validation steps:

Figure 27 • **TIE Analysis**

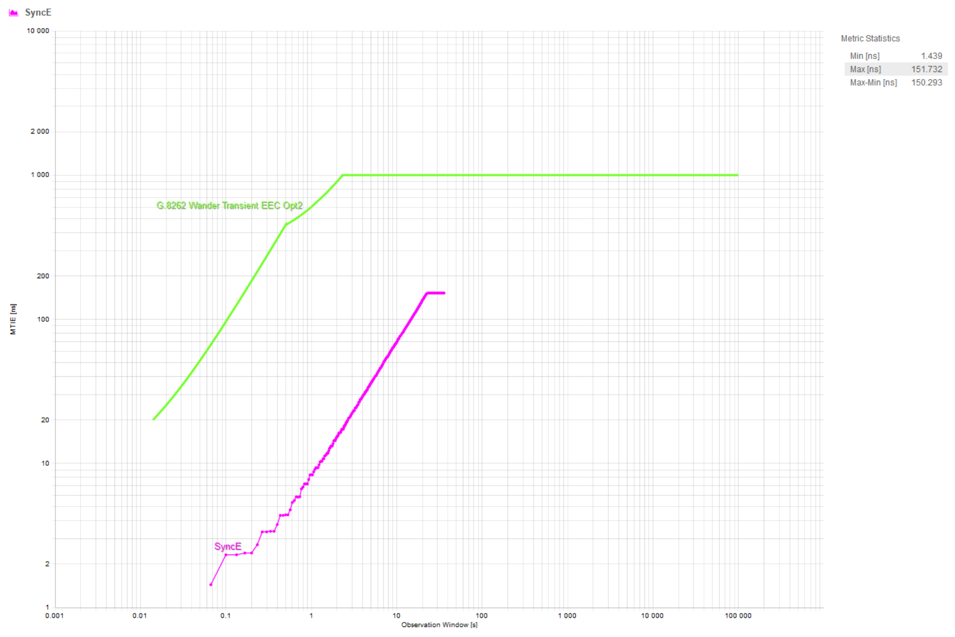
Freq Offset Removal	Off
Frequency Offset	0.0042 ppm



Metric Statistics

Mean [ns]	100.122
Min [ns]	-1
Max [ns]	151
Max-Min [ns]	152

Figure 28 • MTIE Analysis



Metric Statistics

Min [ns]	1.439
Max [ns]	151.732
Max-Min [ns]	150.293