# Speedster7t Ethernet User Guide (UG097)

***Speedster FPGAs***

**Preliminary Data**

Achronix®
Data Acceleration

# Copyrights, Trademarks and Disclaimers



### Preliminary Data

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

### Achronix Semiconductor Corporation

2903 Bunker Hill Lane
Santa Clara, CA 95054
USA

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

# Table of Contents

# Chapter - 1: Introduction

## Summary

Speedster®7t devices include high-speed Ethernet interfaces, which can support a wide variety of Ethernet packet protocols and speeds of up to 400 Gbps per channel. These Ethernet interfaces are paired with latest generation SerDes which individually support 100 Gbps data rates. With eight of these SerDes per Ethernet interface, each interface can support 2× 400 Gbps Ethernet IP channels.

The number of Ethernet interfaces varies according to the device. In the descriptions below, the AC7t1500 device is used as an example. This device has two Ethernet interfaces, allowing for 4× 400 Gbps interfaces, for a combined total bandwidth of 1.6 Tbps.

## Features

### 400G/200G PCS Layer

- 400G over 8× 50G SerDes or 4× 100G SerDes
- 200G over 4× 50G SerDes or 2× 100G SerDes

> **Note**
>
> No support for 25G SerDes (16x25G, 8×25G).

### 100G PCS Layer

- 100G Base-R PCS according to IEEE 802.3 Clause 82 specification.
- 2× SerDes lane with 53 Gbps or 1x SerDes lane with 106 Gbps.
- 4× SerDes lane with 25G (KR4) or 26.5G (KP4).
- Supports Reed-Solomon FEC (RS-FEC) implementing RS(528,514) and RS(544,514) for 100G-KR and 100G-KP applications respectively.

### 10/25/40/50G PCS Layers

- Configurable Base-R PCS compliant with IEEE 802.3 Clauses 49, 82, 107, 133 for 10G, 25G, 50G operation respectively.
- Independent 64bit XLGMII MAC interfaces per channel.
- Supports Reed-Solomon FEC (RS-FEC) implementing RS(528,514) and RS(544,514) for 25G and 50G applications.
- Optional support for EEE fast-wake (i.e., transfer of LPI sequences, no deep sleep).
- Optional Base-R (Firecode) FEC according to Clause 74 of IEEE 802.3.

## Reed-Solomon FEC (RS-FEC)

- Support for RS(528, 514) (KR) codewords and RS(544, 514) (KP) depending on mode of operation.
- Support for RS(272, 258) low-latency variant.
- Support for 25G (Clause 108) and 50G (Clause 134) and 25/50G Ethernet Consortium specifications.
- Support for error indication to PCS when uncorrectable errors are detected.

## MAC

- 1588 precision timing, one-step operation, for for all data rates, 10 to 400G.
- IEEE 802.3br is supported in 10...100G by providing two transmit and receive interfaces to the application.

## Multirate/Multichannel

Up to eight channels independently usable for 10G or 25G or 50G or 100G Ethernet single-lane applications.

- Up to four 50G Ethernet channels using two 25 Gbps lanes each.
- Up to two 100G Ethernet channels using four 25 Gbps lanes each.
- Up to four 100G Ethernet channels using two 50 Gbps lanes each.
- Up to two 200G Ethernet channels using two 50 Gbps lanes each.
- Up to four 200G Ethernet channels using two 100 Gbps lanes each.
- Up to two 400G Ethernet channels using four 100 Gbps lanes each.
- One 400G Ethernet channel using eight 50 Gbps lanes.

# Architecture

The architecture of each Ethernet interface is shown below.

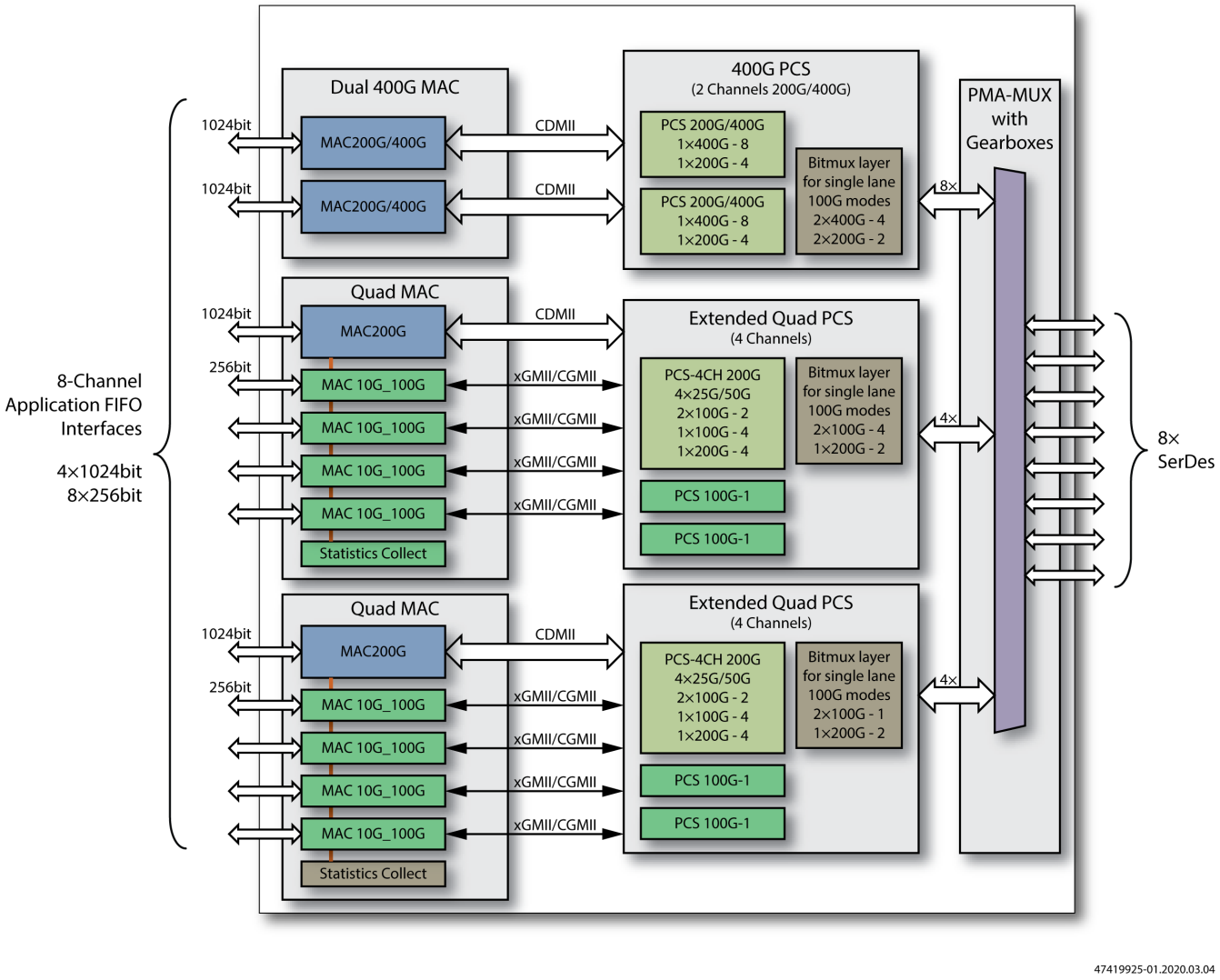


**Figure 1:** ***Ethernet Interface Block Diagram***

# PMA (SerDes)

The physical media attachment (PMA) block consists of eight next-generation SerDes. Each SerDes can operate at up to 106.25 Gbps and down to 10.3125 Gbps. In normal operation the PMA block uses a dedicated mux to connect the SerDes directly to the PCS layer. If the user requires the SerDes for applications other than Ethernet (or to use their own Ethernet PCS and MAC), then the PMA mux can be set to connect the SerDes interface directly to the fabric.

> **Note**
>
> The PMA mux must be switched for all SerDes signals within an interface; therefore, if the user wishes to use any SerDes from an Ethernet interface directly, then all SerDes from that group are switched to the fabric, and the related Ethernet MAC and PCS are not longer available for use

## PCS

The physical coding sublayer (PCS), connects between the SerDes and the MAC. The PCS consists of a dual-channel 400G PCS which can be configured to support either 2× 200G or 2× 400G operation, and twin Quad-PCS, each of which implements a combined four channel PCS, supporting up to a single channel of 200G operation, or four channels with up to 100G per channel. The PCS layer provides the coding functions for the various channel rates supported, including Reed-Solomon error correction of RS(528, 514) (KR) and RS(544, 514) (KP) code words. In addition, support for 25G (Clause 108) and 50G (Clause 134) error correction and coding are also supported.

## MAC

The media access controller (MAC) is constructed from three blocks: one a dedicated dual channel 400G MAC for 400G/200G operation, which connects to the dual channel 400G PCS, and then two instances of a Quad-MAC, each connecting to a Quad-PCS. Each of the Quad-MAC can support a single channel of 200G, or four channels operating at 100G down to 10G per channel. The dedicated 400G MAC is optimized for higher data rates and the wider bus widths necessary for the faster interfaces; the Quad-MAC is equally optimized for multiple channels of lower data rates.

# Specifications

## Channels

Each Ethernet interface can support the following combinations of data rates

**Table 1:** ***Channel configurations***

| Mode – Lanes | SerDes Lanes (per Channel) | SerDes Rate per Lane | Possible No. of Channels | Description (with Coding Options) |
|---|---|---|---|---|
| 400G – 8 | 8 | 53.125G | 1 | 400G over 8 lanes (2:1 bitmux) |
| 400G – 4 | 4 | 106.25G | 2 | 400G over 4 lanes (4:1 bitmux) |
| 200G – 4 | 4 | 53.125G | 2 | 200G over 4 lanes (2:1 bitmux) |
| 200G – 2 | 2 | 106.25G | 4 | 200G over 2 lanes (4:1 bitmux) |
| 100G – 4 | 4 | 25.78125G or 26.5625G | 2 | 100G over 4 lanes (no FEC, RSFEC-KR4 or RSFEC-KP4) |
| 100G – 2 | 2 | 53.125G | 4 | 100G over 2 lanes (RSFEC-KP 2:1 bitmux) |
| 100G – 1 | 1 | 106.25G | 8 | 100G over 1 lane (RSFEC-KP 4:1 bitmux) |
| 50G – 2 | 2 | 25.78125G or 26.5625G | 4 | 50G single lane (RSFEC-KR or RSFEC-KP) |
| 50G – 1 | 1 | 53.125G | 8 | 50G single lane (RSFEC-KP 4:1 bitmux) |

| Mode – Lanes | SerDes Lanes (per Channel) | SerDes Rate per Lane | Possible No. of Channels | Description (with Coding Options) |
|---|---|---|---|---|
| 40G – 4 | 4 | 10.3125G | 2 | 40G over 4 lanes (optional Base-R (Firecode) FEC according to Clause 74 of IEEE 802.3) |
| 40G – 2 | 4 | 20.625G | 4 | 40G over 4 lanes (optional Base-R (Firecode) FEC according to Clause 74 of IEEE 802.3), 2:1 bitmux |
| 25G – 1 | 1 | 25.78125G | 8 | 25G single lane (66b or RSFEC-KR) |
| 10G – 1 | 1 | 10.3125G | 8 | 10G single lane (66b) |

## SerDes

The SerDes supports the following modes and interface width combinations

**Table 2:** ***SerDes Interface Configuration and Frequencies***

| SerDes Speed | PMA Interface Width | PMA Control (sd_x2)(1) select 32(0) or 64(1) | Quad-PCS Control (sd_8x)(2) | Active SerDes Width(3) | PMA Interface Frequency |
|---|---|---|---|---|---|
| 10.3125 Gbps (NRZ) | 128 | 0/1 | 0 | 32/64 | 32 : 322.265635 MHz<br>64 : 161.1328125 MHz |
| 25.78125 Gbps (NRZ) | 128 | 0/1 | 0 | 32/64 | 32 : 805.6640625 MHz<br>64 : 402.83203125 MHz |
| 26.5625 Gbps (NRZ) | 128 | 0/1 | 0 | 32/64 | 32 : 830.078125 MHz<br>64 : 415.0390625 MHz |
| 26.5625 Gbps (PAM4) | 128 | 0/1 | 0 | 32/64 | 32 : 830.078125 MHz<br>64 : 415.0390625 MHz |
| 53.125 Gbps (PAM4) | 128 | x | 1 | 64 | 830.078125 MHz |
| 106.25 Gbps (PAM4) | 128 | x | 1 | 128 | 830.078125 MHz |

| SerDes Speed | PMA Interface Width | PMA Control (sd_x2)(1) select 32(0) or 64(1) | Quad-PCS Control (sd_8x)(2) | Active SerDes Width(3) | PMA Interface Frequency |
|---|---|---|---|---|---|

**Table Note**

1. The PMA control input `sd_x2` is transparent to the PCS and only configures the PMA layer gearboxes to use a double-wide interface to the SerDes (applicable only for SerDes rates less 50G).
2. The Quad-PCS control input `sd_8x` selects the width to use for SerDes rates of 50G and higher.
3. The SerDes width relates to the used bits of the interface. The following applies:

- 32 bit –`sdN_tx[31:0]` and `sdN_rx[31:0]` are used, upper bits ignored/arbitrary.
- 64 bit – `sdN_tx[63:0]` and `sdN_rx[63:0]` are used, upper bits ignored/arbitrary.
- 128 bit – `sdN_tx[127:0]` and `sdN_rx[127:0]` are used.

## Low Latency

Allowing lower latency with 10G and 25G modes when not using any FEC the Quad-PCS supports a fast one-lane mode. In this mode, the PCS provides the SerDes interface data on its internal data path and the PMA layer must bypass all gearboxes to allow minimized delay. The fast one-lane mode is enabled for a lane 'x' by Quad-PCS input `'fast_1lane_mode[x]=1'` and the PMA input `"tx_gb_bypass[x]=1"`. It is available only for a SerDes width of 32 bit, hence `sd_x2` cannot be used. The table below summarizes the settings.

**Table 3:** ***SerDes Low-Latency Settings***

| SerDes speed | PMA Interface Width | PMA Control | | Quad-PCS Control | | Active SerDes Width | PMA Interface frequency |
|---|---|---|---|---|---|---|---|
| | | sd_x2 | tx_gb_bypass | sd_8x | fast_1lane_mode | | |
| 10.3125 Gbps (NRZ) | 128 | 0 | 1 | 0 | 1 | 32 | 322.265625 MHz |
| 25.78125 Gbps (NRZ) | 128 | 0 | 1 | 0 | 1 | 32 | 805.6640625 MHz |

# Chapter - 2: Implementation

## Clocks

### Clock Domains

Within an Ethernet interface there are four clock domains, as shown in the diagram below

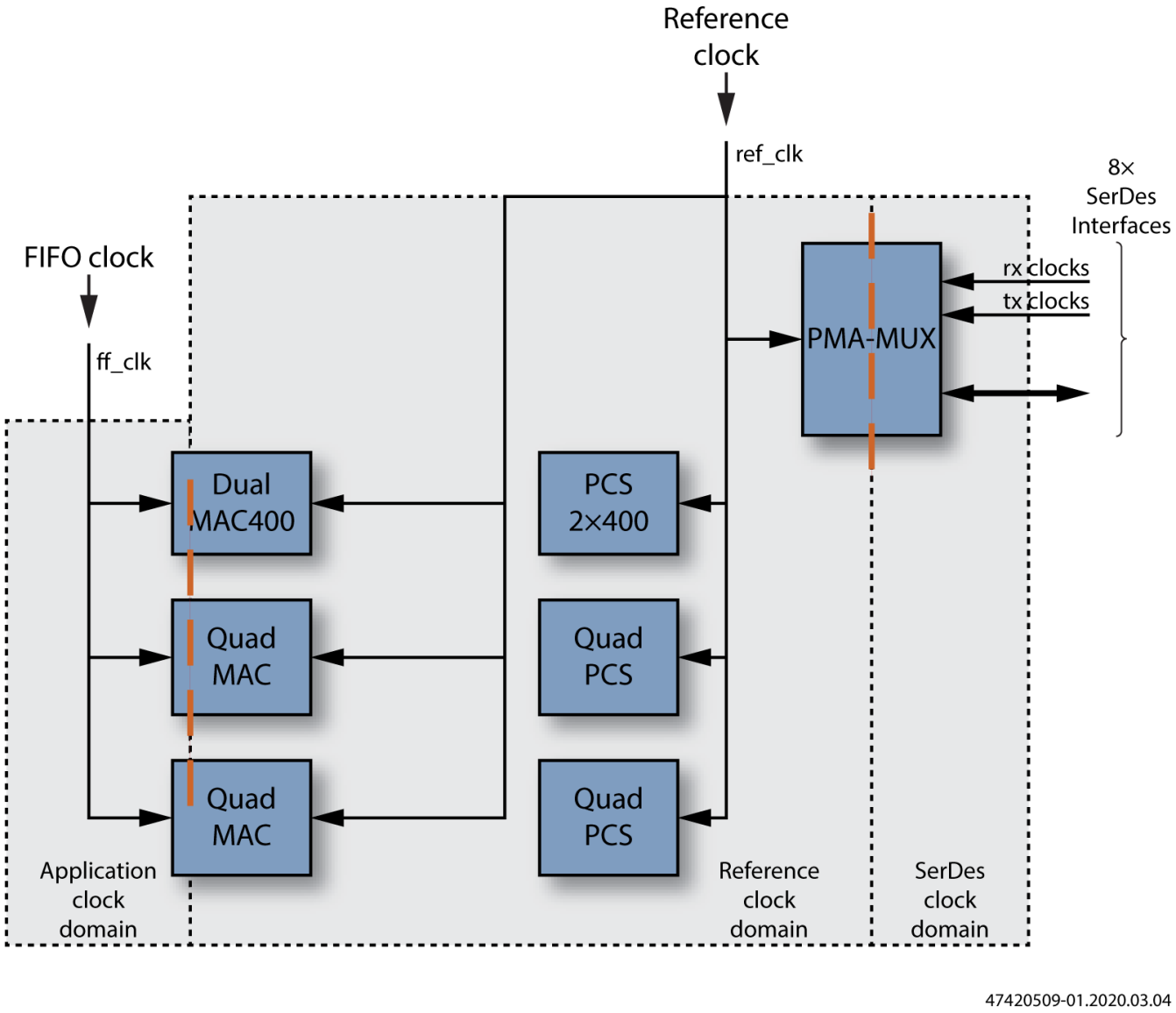


**Figure 2:** ***Ethernet Interface Clock Domains***

The clocks driving these clock domains are as follows

- `ff_tx_clk` – FIFO transmit clock. This clock is driven from the user logic and is used to write the data into the transmit FIFOs. All user interface signals named `ff_tx_XX` are synchronous to this clock. This clock must be fast enough to ensure that it can supply data at the required data rate to avoid the transmit FIFO running empty during packet transmission. The minimum frequencies for `ff_tx_clk` based on the data rate are detailed in Table: Reference and FIFO Clock Frequencies (see page 13).
- `ff_rx_clk` – FIFO receive clock. This clock is driven from the user logic and is used to receive the data from the receive FIFOs. All user interface signals named `ff_rx_YY` are synchronous to this clock. This clock must be fast enough to ensure that it can receive data at the required data rate in order to avoid the receive FIFO overflowing and dropping packets. The minimum frequencies for `ff_rx_clk` based on the data rate are detailed in Table: Reference and FIFO Clock Frequencies (see page 13).
- `ref_clk` – This clock drives the internal logic of the MAC and PCS. This clock domain interfaces to the FIFO clock domains from the user interface, through to the PMA interface driving the SerDes. The clock must be fast enough to allow the PCS and MAC to process the packets. The minimum frequencies for `ref_clk` based on the data rate are detailed in Table: Reference and FIFO Clock Frequencies (see page 13).
- `serdes_clk` – This clock drives the SerDes and is related to the SerDes line rate. The SerDes clock requirements are detailed in Table: SerDes Clock Frequencies (see page 14).

The four clock domains operate independently, using asynchronous FIFOs to decouple the data flows between them. There are no frequency or phase requirements between the clocks; however, they must meet the minimum frequencies detailed in the tables below.

# Clock Frequencies

## Reference and FIFO Clocks

The frequency requirement for the reference clock `ref_clk` , and the user FIFO clocks, ff_tx_clk and ff_rx_clk, depend on the mode of operation. The highest mode active in any of the channels defines the minimum requirement. The clock frequencies ranges are detailed in the table below.

**Table 4: *Reference and FIFO Clock Frequencies***

| Mode | ref_clk (Min) | ref_clk (Max) | ff_clk (Recommended) | |
|---|---|---|---|---|
| | | | Min Theoretical | Min Recommended |
| 10G | 323 MHz | 900 MHz | 59 MHz | 160 MHz |
| 25G | 782 MHz (25G no RSFEC)<br>831 MHz (25G with RSFEC) | 900 MHz | 147 MHz | 210 MHz |
| 50G | 831 MHz | 900 MHz | 293 MHz | |
| 100G | 831 MHz | 900 MHz | 586 MHz | |
| 200G | 831 MHz | 900 MHz | 392 MHz | |
| 400G | 831 MHz | 900 MHz | 782 MHz | |

**Warning!**

The given minimum frequencies must be respected. For example, for 25G, 781.25 MHz is not sufficient, it must be 782 MHz or higher.

### SerDes Clocks

SerDes clock frequency is directly proportional to the SerDes speed and interface width, as detailed in the table below.

**Table 5: *SerDes Clock Frequencies***

| SerDes Speed | Active SerDes Width | PMA Interface Frequency |
|---|---|---|
| 10.3125 Gbps (NRZ) | 32/64 | 32 : 322.265635 MHz<br>64 : 161.1328125 MHz |
| 25.78125 Gbps (NRZ) | 32/64 | 32 : 805.6640625 MHz<br>64 : 402.83203125 MHz |
| 26.5625 Gbps (NRZ) | 32/64 | 32 : 830.078125 MHz<br>64 : 415.0390625 MHz |
| 26.5625 Gbps (PAM4) | 32/64 | 32 : 830.078125 MHz<br>64 : 415.0390625 MHz |
| 53.125 Gbps (PAM4) | 64 | 830.078125 MHz |
| 106.25 Gbps (PAM4) | 128 | 830.078125 MHz |

### Selecting Clock Frequencies

The selection of clock frequencies is greatly enhanced by use of the ACE I/O Designer tool, as detailed in Ethernet IP Software Support in ACE (see page 26). Upon selecting the desired Ethernet data rates, I/O Designer specifies what the required minimum clock frequencies are. In addition, I/O Designer actively checks that the clock sources are connected to suitable PLLs, and that those PLLs are correctly configured to generate the desired frequencies. Using I/O Designer greatly simplifies the task of ensuring the correct frequencies are defined at the start of a design.

## Loopback

For verification and validation purposes several, two loopback modes are implemented in the PMA:

- A direct (local) loopback returning all data from the PCS transmit back to the PCS receive. Transmitted data will be transmitted normally to the SerDes. Received data from the SerDes is ignored.
- A reverse (remote) loopback re-transmitting all data from SerDes receive back to SerDes transmit. Received data from the SerDes is passed through to the PCS. Transmitted data from the PCS is ignored.

The loopbacks operate all within the system reference clock domain, hence avoiding need for specific buffering.

**Note**

The loopback modes can be enabled for all lanes only — either all channels operate in a loopback mode or their normal data path.

## Direct (Local) PMA Loopback Mode

The direct PMA loopback mode is enabled by setting loopback input pin (`loopback`) to high. Even if the direct loopback mode is enabled, the transmit SerDes clock (`sd_tx_clk`) must be run at a frequency corresponding the selected operational mode.

## Reverse SerDes Loopback Mode

The reverse SerDes loopback mode is enabled by setting the loopback input pin (`loopback_rev`) to high. Even if the reverse loopback mode is enabled, the transmit/receive SerDes clock (`sd_clk_tx/rx`) must be run at a frequency corresponding the selected operational mode (and the transmit and receive SerDes bandwidths must the same).

## Loopback Limitations

The direct (local) PMA loopback can only be used when RS-FEC mode is active (i.e., 10G/25G plain 64/66b modes cannot use loopback). The root cause for this limitation is that for non-RS FEC mode, the PCS output is 66-bit block based (0 to 65 bits). The final translation from the 66-bit internal bus to 40/80-bit SerDes interface is done by the PMA block in the SerDes clock domain. For the non-RS FEC. the receive PCS input mode always expects 40-bit data. In order to loopback the 66-bit output to 40-bit input, an extra gear box with memory is required.

**Note**

When using fast_1lane_mode=1, the loopback is functional also for 10G/25G modes, but operates slower than the line rate by a factor of (40/32).

# Chapter - 3: Ethernet Ports

The interface to the Ethernet controller is split into three access mechanisms:

- Data is delivered via the NoC, and using the NAP in data streaming mode.
- Control and status lines are accessed directly via the direct-connect interface.
- Internal Ethernet controller registers are accessed via a NAP used in AXI-4 mode.

The control and status lines are documented below.

## Naming Convention

The Ethernet ports have a logical and consistent naming scheme, consisting of `<prefix>_<mac_identifier>_function`. The details of <prefix> and <mac_identifier> are listed below.

### Prefix

Wrappers for the Ethernet IP blocks are generated using ACE I/O Designer. Each Ethernet interface is named during generation. The Ethernet interface name is prefixed to each signal name, to distinguish different Ethernet interfaces and to aid users in having logical names for each interface. In the tables below, this prefix is shown as `<prefix>_` on each signal name.

### MAC Identifier

The control and status lines use identifiers to logically group signals from the same MAC:

- `m80` is for 400G/200G MAC
- `mq0` is for QUAD0 MAC
- `mq1` is for QUAD1 MAC

This identifier follows the prefix field in the signal name table below

**Table 6: *Ethernet Controller Direct Connect Interface***

| Pin Name(1,2) | Direction | Width | Comments | Clock | Raw Mode = 0 Data Going to Ethernet |
|---|---|---|---|---|---|
| **Quad MAC** | | | | | |
| <prefix>_mq<n>_tx_hold_req | Input | 4 | Per channel 100G MACs. Holds and preempts if needed the pMAC (optional, e.g., for test/debug or if higher layer function anticipates an eMAC frame being written soon and wants to prepare the MAC instead having the MAC doing it automatically from the eMAC FIFO being non-empty). | Synchronous to (Application0 Clock)/2 | mq<n>_tx_hold_req |
| <prefix>_mq<n>_lpi_txhold | Input | 4 | Per channel 100G MACs. Prevents MAC transmit from transmitting a frame even if data is stored in FIFO. | Asynchronous Input Synchronized internally on Reference Clock | mq<n>_lpi_txhold |

| Pin Name[1,2] | Direction | Width | Comments | Clock | Raw Mode = 0 Data Going to Ethernet |
|---|---|---|---|---|---|
| <prefix>_mq<n>_time_1ms_tgl | Input | 4 | Per MAC channel 1 ms time base. It is required for internal timing functions. The signal must toggle its value every 1 ms. | Asynchronous Input | mq<n>_time_1ms_tgl |
| <prefix>_mq<n>_mac_stop_tx | Input | 4 | Per channel control of MAC transmit. For each lane, when it's respective input is asserted (1'b1), the MAC transmit state machine stops after any ongoing frame has been sent completely (i.e., does not corrupt outgoing frames). If further frames are available in the transmit FIFO the MAC will not begin transmitting them until the respective mac_stop_tx is de-asserted (1'b0) again. | Synchronous to (ref_clk)/2 | mq<n>_mac_stop_tx[3:0] |
| <prefix>_mq<n>_emac_xoff_gen | Input | 32 | Transmit flow control generate (8 bits per channel) to eMAC /pMAC. When PFC pause mode is enabled, an 8-bit input vector is used to signal the creation of PFC control frames. When link pause mode is enabled, bit 0 (per channel, i.e., bits 0,8,16,24) is used only. | Asynchronous input synchronized to ref_clk | mq<n>_emac_xoff_gen[31:0] |
| <prefix>_mq<n>_pmac_xoff_gen | Input | 32 | | | mq<n>_pmac_xoff_gen[31:0] |
| <prefix>_mq<n>_mac_peer_delay | Input | 120 | A peer delay value that can be added to the correction field for all one-step updates (30 bits per channel). Must be wired to 0 if unused. | Synchronous to ref_clk | mq<n>_mac_peer_delay_val[119:0] |
| <prefix>_mq<n>_mac_peer_delay_val | Input | 4 | Per channel valid strobe for mac_peer_delay (1 bit per channel). Must assert for 1 ref_clk clock cycle when the peer delay was updated to write the mac_peer_delay value to the MAC internal register. Once the value has been written (i.e., peer_delay_val 0 again) the peer_delay input is no longer relevant and can have arbitrary data. Must be wired to 0 if unused. | Synchronous to (ref_clk)/2. | mq<n>_mac_peer_delay_val[3:0] |
| <prefix>_mq<n>_pmac_pause_on | Output | 32 | Transmit paused/class congestion Indication (8-bit value per channel) from eMAC/pMAC. Bit 0 (per channel, i.e., bits 0,8,16,24) is also used to indicate link pause. When asserted to '1' indicates a running pause counter that has been started because a Xoff frame (pause/PFC) was received. In link pause mode, the transmitter is also stopped when the command_config configuration bit PAUSE_PFC_COMP is not set. When the PAUSE_PFC_COMP bit is set, the transmitter will not be stopped and it is the responsibility of the application to assert mac_stop_tx to implement proper flow control. | Synchronous to ref_clk/2 | mq<n>_pmac_pause_on[31:0] |
| <prefix>_mq<n>_emac_pause_on | Output | 32 | | | mq<n>_emac_pause_on[31:0] |
| <prefix>_mq<n>_pmac_pause_en | Output | 4 | General-purpose indication that the eMAC/pMAC is configured to react on pause frames (1-bit per channel). It is a direct result of the inverted COMMAND_CONFIG (PAUSE_IGNORE) control bit. | Synchronous to reg_clk | mq<n>_pmac_pause_en[3:0] |
| <prefix>_mq<n>_emac_pause_en | Output | 4 | | | mq<n>_emac_pause_en[3:0] |
| <prefix>_mq<n>_pmac_enable | Output | 4 | General-purpose indication that the eMAC/pMAC datapaths have been enabled. It is a direct result of both the COMMAND_CONFIG(TX_EN & RX_EN) control bits. Per channel from 100G MACs. Can be left unconnected if not used.<br>**Note**<br>A corresponding signal from the 200G MAC does not exist. | Synchronous to reg_clk | mq<n>_pmac_enable[3:0] |
| <prefix>_mq<n>_emac_enable | Output | 4 | | | mq<n>_emac_enable[3:0] |

| Pin Name[1,2] | Direction | Width | Comments | Clock | Raw Mode = 0 Data Going to Ethernet |
|---|---|---|---|---|---|
| <prefix>_mq<n>_mac_tx_ovr_err | Output | 4 | FIFO overflow truncation error indication. Asserts when the FIFO write control logic had to truncate a frame as either the ff_tx_rdy deassertion was not respected by the application, or the frame transferred is larger than the FIFO in store-and-forward mode of operation. | Synchronous to ref_clk/2 | mq<n>_mac_tx_ovr_err[3:0] |
| <prefix>_mq<n>_ffp_tx_ovr | Output | 4 | | | mq<n>_ffp_tx_ovr[3:0] |
| <prefix>_mq<n>_ffe_tx_ovr | Output | 4 | | | mq<n>_ffe_tx_ovr[3:0] |
| <prefix>_mq<n>_mac_tx_underflow | Output | 4 | Transmit FIFO became empty during transmission. A frame has been corrupted. | Synchronous to ref_clk/2 | mq<n>_mac_tx_underflow[3:0] |
| <prefix>_mq<n>_pmac_tx_empty | Output | 4 | Transmit FIFO Empty Indication from pMAC. When set to 1, indicates that the transmit FIFO is empty. When set to 0, indicates transmit FIFO has data. When the Quad operates in 200G, pmac_tx_empty[0] indicates empty for the 200G MAC FIFO. | Synchronous to ref_clk/2 | mq<n>_pmac_tx_empty[3:0] |
| <prefix>_mq<n>_emac_tx_empty | Output | 4 | Transmit FIFO Empty Indication from eMAC. When set to 1, indicates that the transmit FIFO is empty. When set to 0, indicates transmit FIFO has data. When the Quad operates in 200G, emac_tx_empty is unused/not relevant. | | mq<n>_emac_tx_empty[3:0] |
| <prefix>_mq<n>_mac_tx_isidle | Output | 4 | Transmit datapath is not transmitting when 1. Will toggle during normal operation, but is not necessarily frame accurate (i.e., may not always de-assert during minimum IPG). | Synchronous to ref_clk/2 | mq<n>_mac_tx_isidle[3:0] |
| <prefix>_mq<n>_link_up | Input | 4 | Per channel indication from the PCS that the link is up. The application must drive this signal from the PCS link_status, and potentially from loc_fault and rem_fault. It is used to detect a link loss in the MAC's transmit merge sub-layer to disable pre-emption. | Synchronous to (ref_clk)/2 | mq<n>_link_up[3:0] |
| <prefix>_mq<n>_mac_tx_ts_val | Output | 4 | Timestamp valid. Asserted for one clock cycle to indicate that mac_tx_ts_id and mac_tx_tsN are valid. The signal is not asserted for internally generated pause frames. | Synchronous to ref_clk/2 | mq<n>_mac_tx_ts_val[3:0] |
| <prefix>_mq<n>_mac_tx_ts_id | Output | 16 | Frame identifier return (4-bit value per channel). The value that was provided by the application at ff_tx_id(3:0) for the frame. | Synchronous to ref_clk | mq<n>_mac_tx_ts_id[4*4-1:0] |
| <prefix>_mq<n>_mac_fault_ored4l | Output | 1 | Local, remote, and link interruption faults. ORed into a single output. | Synchronous to ref_clk/2 | \|mq<n>_mac_loc_fault[3:0]) \|\| (\|mq<n>_mac_rem_fault[3:0]) \|\| (\|mq<n>_mac_li_fault[3:0] |
| <prefix>_mq<n>_mac_tx_ts | Output | 256 | Frame timestamp value return (64-bit value per channel). Transmit timestamp value for the frame sent with the frame identifier set on mac_tx_ts_id. Returns the value sampled from mac_frc_i_tx. | Synchronous to ref_clk | mq1_mac_tx_ts |
| **400G/200G MAC[3]** | | | | | |
| <prefix>_m80_c<m>_xoff_gen | Input | 8 | Transmit flow control generate. When PFC pause mode is enabled, an 8-bit input vector is used to signal the creation of PFC control frames. When link pause mode is enabled, bit 0 is used only. | Asynchronous input synchronized to ref_clk | m80_c<m>_xoff_gen[7:0] |

| Pin Name[1,2] | Direction | Width | Comments | Clock | Raw Mode = 0 Data Going to Ethernet |
|---|---|---|---|---|---|
| <prefix>_m80_c<m>_tx_smhold | Input | 1 | Instructs the MAC to stop reading further data from the transmit FIFO at the next possible frame boundary. | Synchronous to (ref_clk)/2 | m80_c<m>_tx_smhold |
| <prefix>_m80_c<m>_peer_delay | Input | 30 | Current value of the link delay measured at the ingress port that receives SYNC messages from a master. This input is a global value which is updated from time to time by the PTP software when implementing peer-to-peer transparent clock systems. | Synchronous to ref_clk | m80_c<m>_peer_delay[29:0] |
| <prefix>_m80_c<m>_peer_delay_val | Input | 1 | Indicates validity of peer_delay(). Must be a pulse for one cdmii_txclk cycle whenever the peer_delay() input was updated. | Synchronous to (ref_clk)/2. | m80_c<m>_peer_delay_val |
| <prefix>_m80_c<m>_pause_on | Output | 8 | Transmit paused/class congestion indication, one bit per priority class. When asserted to '1' indicates a running pause counter has been started because an Xoff frame (pause/PFC) was received. In link pause mode, the transmitter is also stopped (if not disabled by COMMAND_CONFIG.PAUSE_PFC_COMP configuration setting). | Synchronous to ref_clk/2 | m80_c<m>_pause_on[7:0] |
| <prefix>_m80_c<m>_tx_ovr_err | Output | 1 | FIFO overflow truncation error indication. Asserts when the FIFO write control logic had to truncate a frame as either the ff_tx_rdy de-assertion was not respected by the application, or the frame transferred is larger than the FIFO in store-and-forward mode of operation. | Synchronous to ref_clk/2 | m80_c<m>_tx_ovr_err |
| <prefix>_m80_c<m>_tx_underflow | Output | 1 | Transmit FIFO became empty during transmission. A frame has been corrupted. | Synchronous to ref_clk/2 | m80_c<m>_tx_underflow |
| <prefix>_m80_c<m>_fault | Output | 1 | Rx receives a local or remote fault. | Synchronous to ref_clk/2 | m80_c<m>_loc_fault \|\| m80_c<m>_rem_fault |
| <prefix>_m80_c<m>_tx_ts | Output | 64 | Frame timestamp value return (64-bit value per channel). Transmit timestamp value for the frame sent with the frame identifier set on mac_tx_ts_id. Returns the value sampled from mac_frc_i_tx. | Synchronous to ref_clk | m80_c<m>_tx_ts[63:0] |
| <prefix>_m80_c<m>_tx_ts_id | Output | 4 | Frame identifier return (4-bit value per channel). The value that was provided by the application at ff_tx_id(3:0) for the frame. | Synchronous to ref_clk | m80_c<m>_tx_ts_id[3:0] |
| <prefix>_m80_c<m>_tx_ts_val | Output | 1 | Timestamp valid. Asserted for one clock cycle to indicate that mac_tx_ts_id and mac_tx_tsN are valid. The signal is not asserted for internally generated pause frames. | Synchronous to ref_clk/2 | m80_c<m>_tx_ts_val |
| <prefix>_m80_c<m>_tx_empty | Output | 1 | Transmit FIFO empty. | Synchronous to ref_clk/2 | m80_c<m>_tx_empty |
| <prefix>_m80_c<m>_tx_isidle | Output | 1 | Indicates (when 1) transmit state machine is not transmitting a frame currently. | Synchronous to ref_clk/2 | m80_c<m>_tx_isidle |
| <prefix>_m80_c<m>_frm_drop | Output | 1 | Frame drop indication. A frame drop occurs when a frame contains less than 64 data bytes or the application was not ready to accept a frame (ff_rx_rdy=0) at begin of the frame. In both cases, the frame is not delivered to the application. | Synchronous to ref_clk/2 | m80_c<m>_frm_drop |
| **Common** | | | | | |
| <prefix>_ref_clock_divby2 | Output | 1 | Reference clock divided by 2 | | |

| Pin Name(1,2) | Direction | Width | Comments | Clock | Raw Mode = 0 Data Going to Ethernet |
|---|---|---|---|---|---|
| <prefix>_m0_ff_clk_divby2 | Output | 1 | Application0 clock divided by 2 | | |
| <prefix>_m1_ff_clk_divby2 | Output | 1 | Application1 clock divided by 2 | | |

**Table Notes**

1. The Ethernet interface name (shown as `<prefix>_`) is prefixed to each signal name, to distinguish different Ethernet interfaces and to aid users in having logical names for each interface.
2. The variable <n> indicates the MAC quad number: 0 for QUAD0 MAC, and 1 for QUAD1 MAC.
3. The variable <m> indicates the channel number: 0 or 1.

# Chapter - 4: Speedster7t Ethernet NoC Connectivity

As previously detailed, the data from the Ethernet IP is delivered via the NOC, using the NAPs in data streaming mode.

The Ethernet MAC connects directly to specific columns on the NoC and can communicate to FPGA fabric logic connected to NAPs along those specific columns using Ethernet packets. Each Ethernet MAC has two dedicated columns and can send transactions to NAPs placed only on those two specific columns. The table below lists the specific columns connected to the Ethernet MACs.

**Table 7: *NoC Columns for Ethernet MACs***

| Ethernet MAC location | Ethernet MAC 0 (West) | Ethernet MAC 1 (East) |
|---|---|---|
| **NoC Column 1** | 1 | 4 |
| **NoC Column 2** | 2 | 5 |

**Table Note**

NoC Columns are numbered 1 at the west-most column and increment going east.

There are a few modes available, depending on how the user wishes to handle the Ethernet packets in the FPGA fabric. For interfaces using 100GE or slower, the Ethernet sends 256-bit packets down the columns directly to NAPs. For interfaces running 200GE or 400GE, there are two modes to choose from: packet mode or quad-segmented mode.

## Packet Mode

The NoC rearranges the 1024-bit data bus into four narrower data paths, funneling a separate packet to each of four NAPs and spliting the full 1024-bit data bus into four 256-bit (32-byte) data paths. This solution results in less congestion in the fabric because the user logic can reside in four separate engines distributed down the NoC columns rather than a single large engine immediately next to the Ethernet MAC. This mode also reduces the needed frequency in the FPGA fabric design and makes the design easier because each NAP can have its own individual packet processing engine.

Packet mode can result in larger latency as each packet can take more cycles to transfer. Importantly packets can arrive out of order, with the NoC sending a sequence number along with each packet. The user logic is then responsible for reordering the packets (if necessary), in order to completely retrieve the original data sequence. The figure below shows how the Ethernet MAC data bus is rearranged into four separate 256-bit wide data buses. Each packet can take multiple cycles to complete.

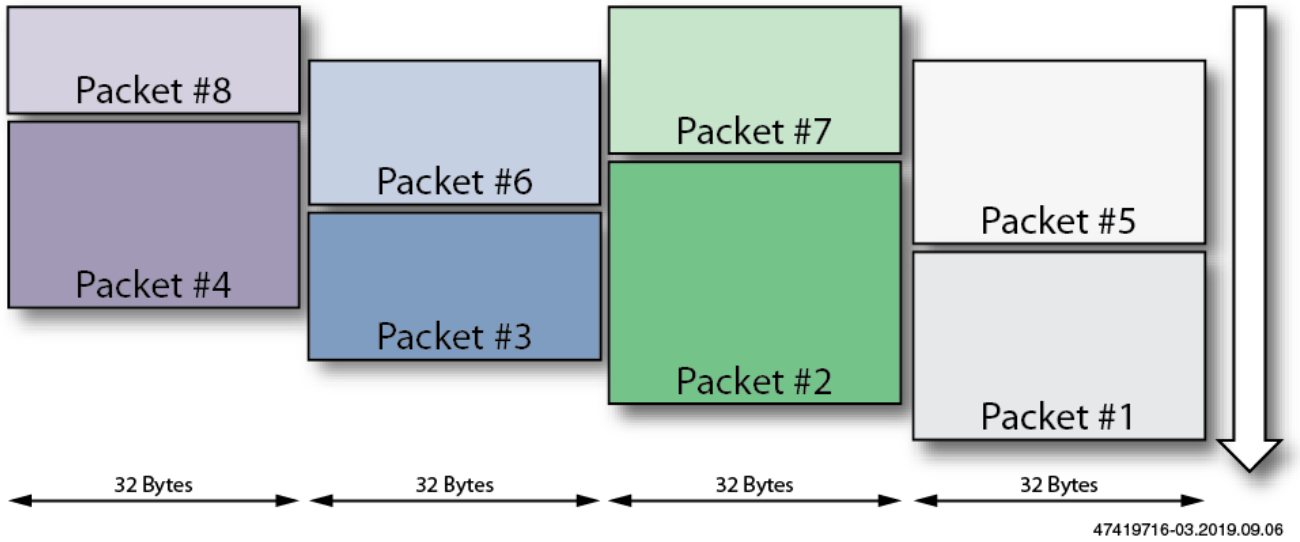


**Figure 3:** ***Data Bus Rearrangement for Packet Mode***

The four packets shown above are sent to four separate NAPs distributed down the designated NoC columns. Each NAP talks to an individual packet processing engine in the FPGA fabric that can run at a lower frequency than a single engine processing the full 1024-bit bus, thus simplifying the system design. The NoC automatically handles the load balancing, sending the next available packet to the next free NAP. For more details on Ethernet packet mode, refer to the Speedster7t Ethernet User Guide (see page 5).

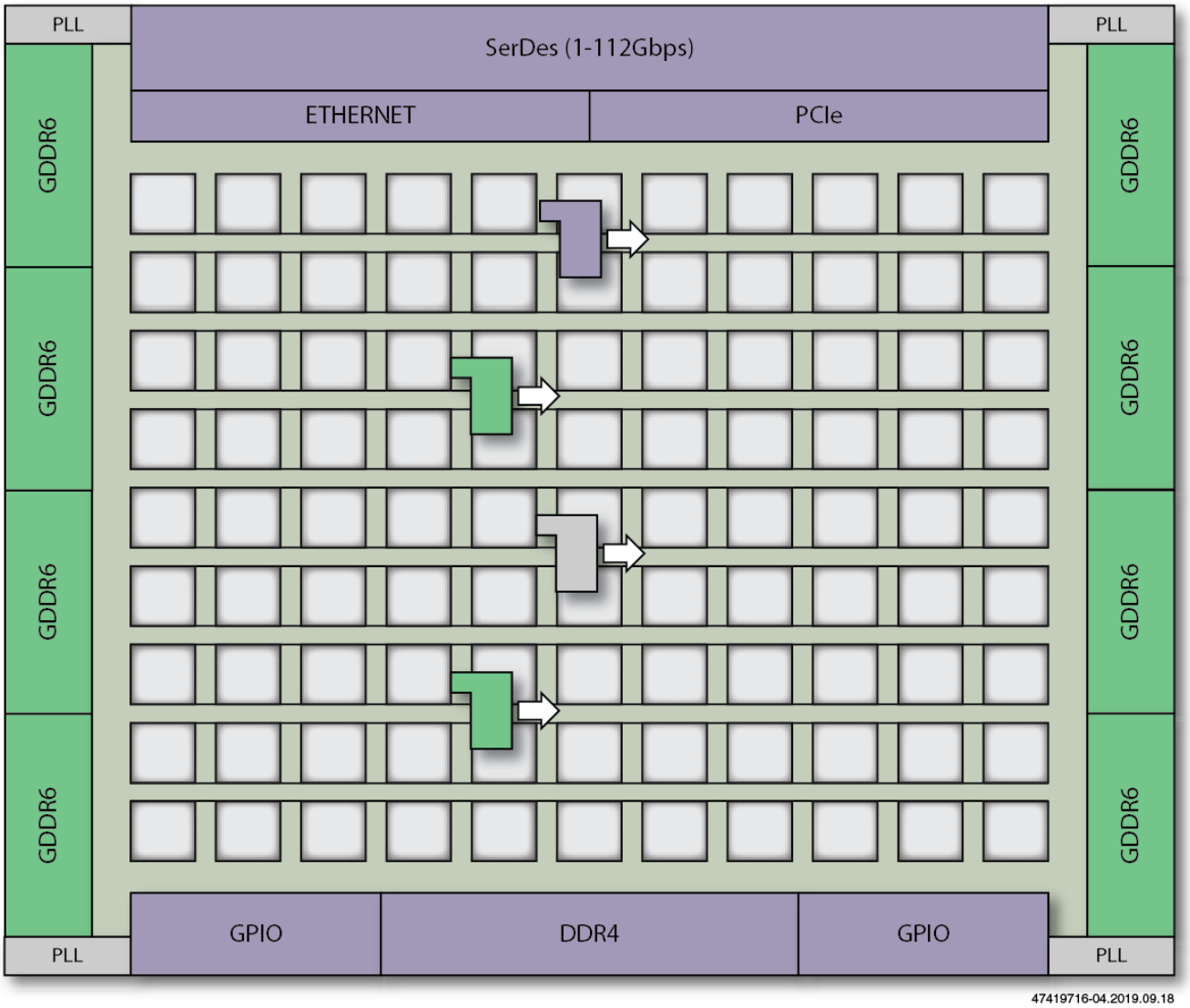


**Figure 4:** ***Ethernet Packet Mode on the NoC***

## Quad-Segmented Mode

In quad-segmented mode, the NoC sends a 1024-bit bus that is segmented across four NAPs. This mode makes the user logic a little more complex as the design logically is one large packet processing engine distributed across the four NAP locations. This mode does guarantee in-order packet arrival, and larger packets arrive with less latency than in packet mode described above. Because the bus is segmented, packets can potentially start at any of the four NAPs, and up to two packets can arrive in a single cycle.

Similar to the packet mode above, the FPGA logic can be spread across the space of four NAPs on the designated columns, rather than having to be placed immediately next to the Ethernet MAC. This arrangement helps ease congestion, and because the design can be split across four NAPs, the frequency can be reduced similar to the packet mode. The figure below shows how the packets are arranged and segmented for the quad-segmented mode.

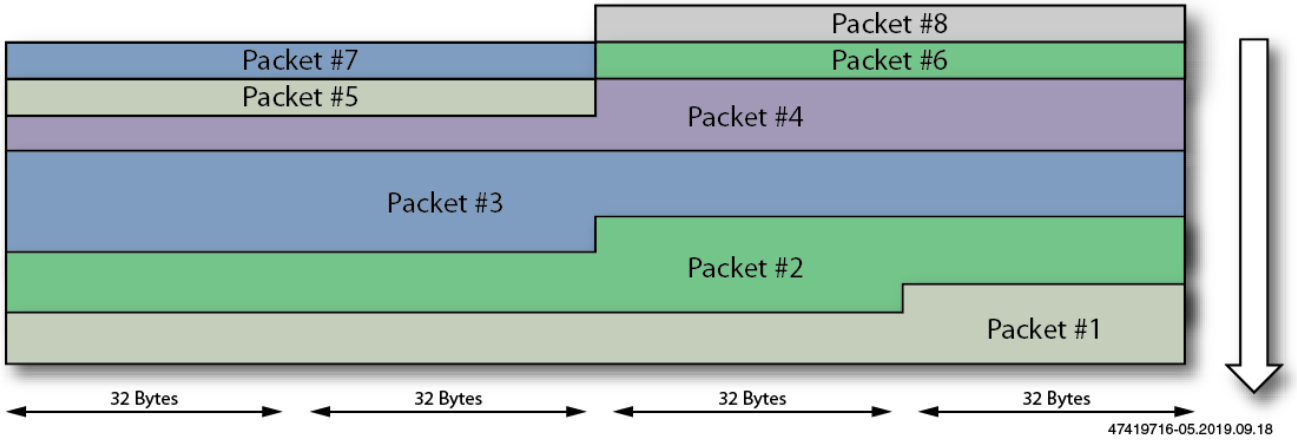


**Figure 5:** ***Packet Segmentation for Quad-Segmented Mode***

Each packet is distributed across four NAPs located on the designated columns of the NoC. The packet processing engine should be located close to the four NAPs.

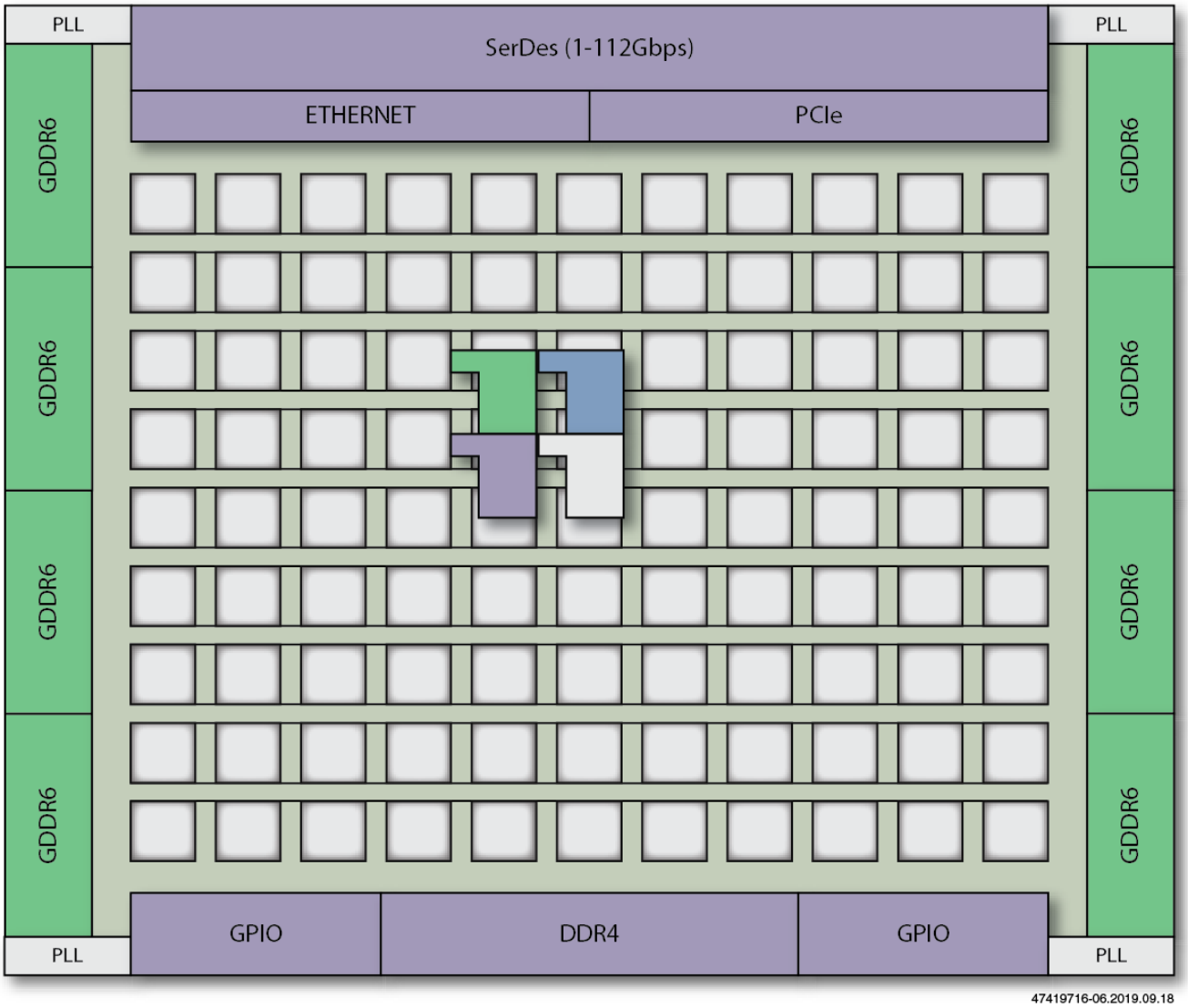


**Figure 6:** ***Quad-segmented Mode on the NoC***

# Chapter - 5: Ethernet IP Support in ACE

## Overview

Ethernet Interface IP generation in ACE provides a GUI to generate and integrate the Ethernet Interface instances based on the user specified inputs. The I/O Designer in ACE supports the integration of all the chosen IP for the user design and also allows the user to select the placement and visualize package routing. Once the desired IP is configured via the I/O Designer GUI, the tool generates a bitstream for the entire IP interface which is independent of the bitstream generated for the core fabric. The tool then integrates both these bitstreams into a single configurable bitstream targeting a Speedster7t device.

The following steps provide a brief description on creating an Ethernet IP interface design:

## Step 1 – Creating a Project

Create a project in ACE, and then in the 'Project perspective', select the target device **AC7t1500ES0** which ensures that the appropriate IP options are available in the IP Perspective window in ACE.

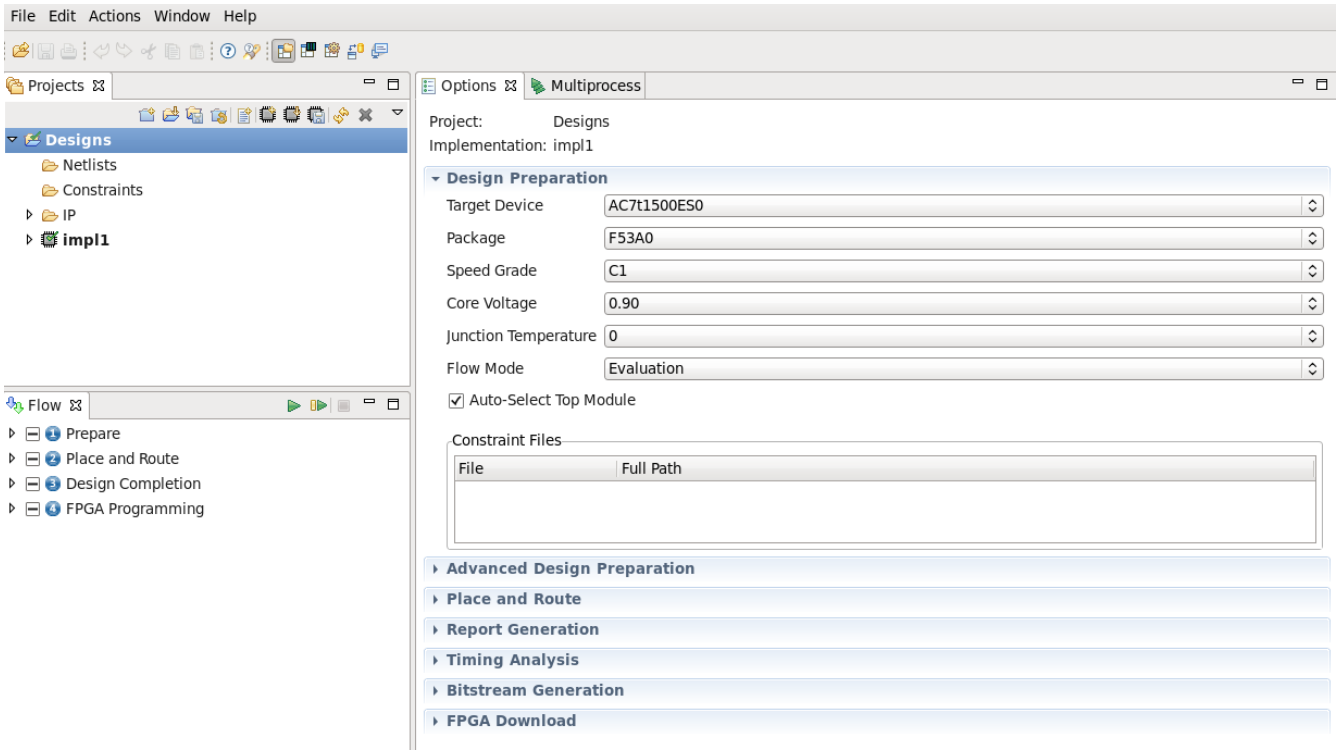


**Figure 7:** ***Design Preparation Options in the ACE Project***

# Step 2 – Configure GPIO as a Clock Input

Switch to the 'IP Configuration' perspective and under **Speedster7t** select **IO Ring** then select **Programmable IO.** Select a suitable name for the instance (such as `clock_io.acxip`). In the Programmable IO wizard, click **Add**, and then set the **I/O Instance Name** to sys_clk, select the **Signal Type** to *Clock*, and the **Bank Type** to *CLKIO*. Assign the external clock input pins in the **Placement** field to automatically populate the **Ball Name** and **Bank**. On the additional pages, configure the desired frequency for this clock input. Once the selection is made, under the **I/O Designer** pane, the **Layout Diagram** highlights the chosen clock input. The **IP Problems** pane highlights any errors or warnings which occurred while configuring this GPIO as a clock input.

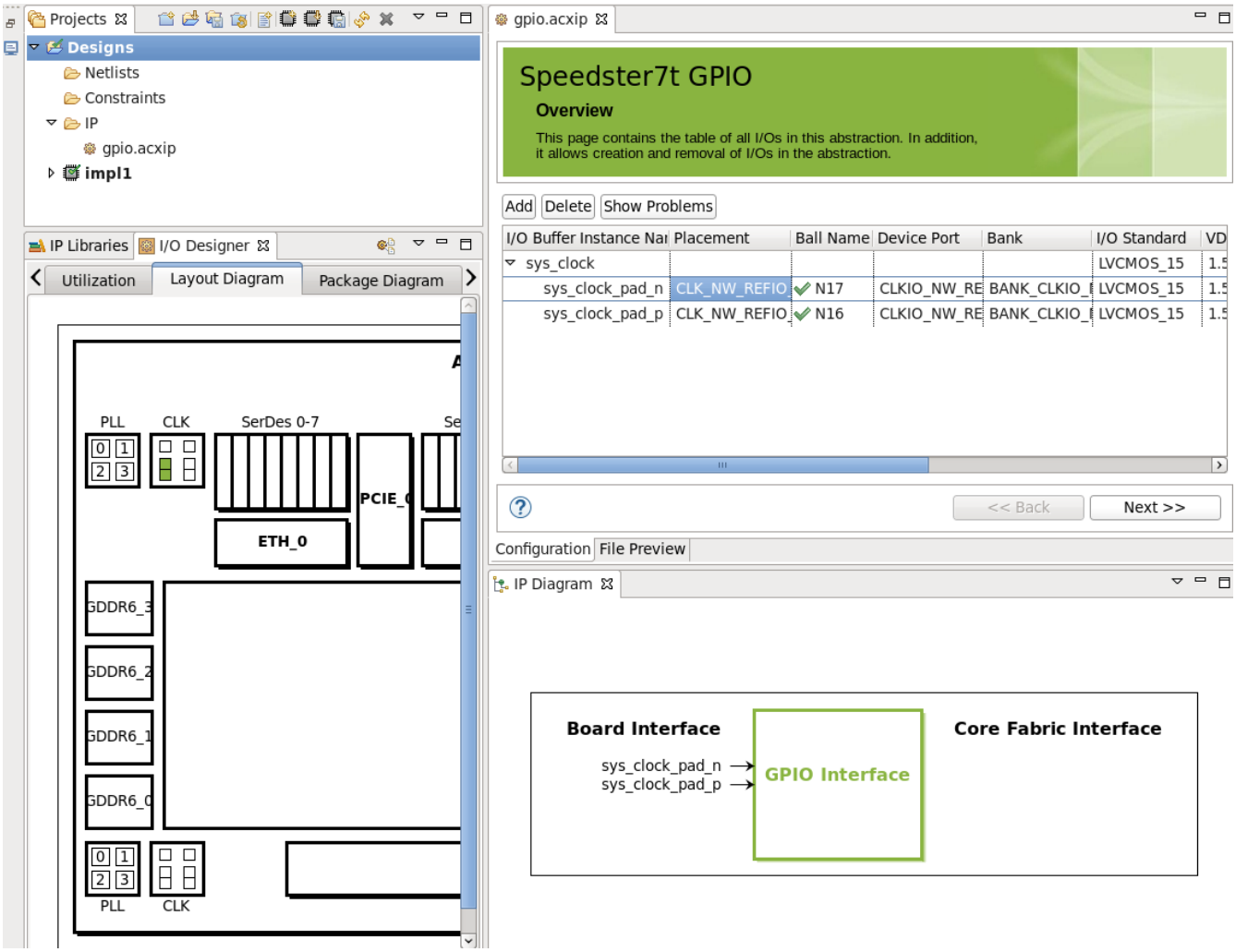


**Figure 8:** ***GPIO IP Configuration in ACE I/O Designer***

# Step 3 - Configure the PLL

Next, in the same IP Libraries, select **PLL**. Configure the PLL IP with the desired placement in the same corner as the input clock and the appropriate clock output frequencies based on the required Ethernet Interface data rates. The Ethernet IP Interface clock requirements are listed in Table: Reference and FIFO Clock Frequencies. Each Ethernet Interface requires at least two clocks, a reference clock (ref_clk) and a user application clock (ff_clk) per Ethernet channel. The name of the PLL, (defined by the `.acxip` file used to generate it) defines the clock names used for connecting to subsequent IP. In this example, leave the PLL with the default name of `advanced_pll_1`.

For the purposes of this example, four clocks from the PLL are required:

- 200 MHz for the NoC – name this signal `noc_clk`
- 900 MHz as the Ethernet reference clock – name this signal `eth_ref_clk`
- 600 MHz as the Ethernet channel 0 user application clock – name this signal `eth_ff1_clk`
- 600 MHz as the Ethernet channel 0 user application clock – name this signal `eth_ff2_clk`

The common VCO frequency for these four frequencies is set to be 7200 MHz.

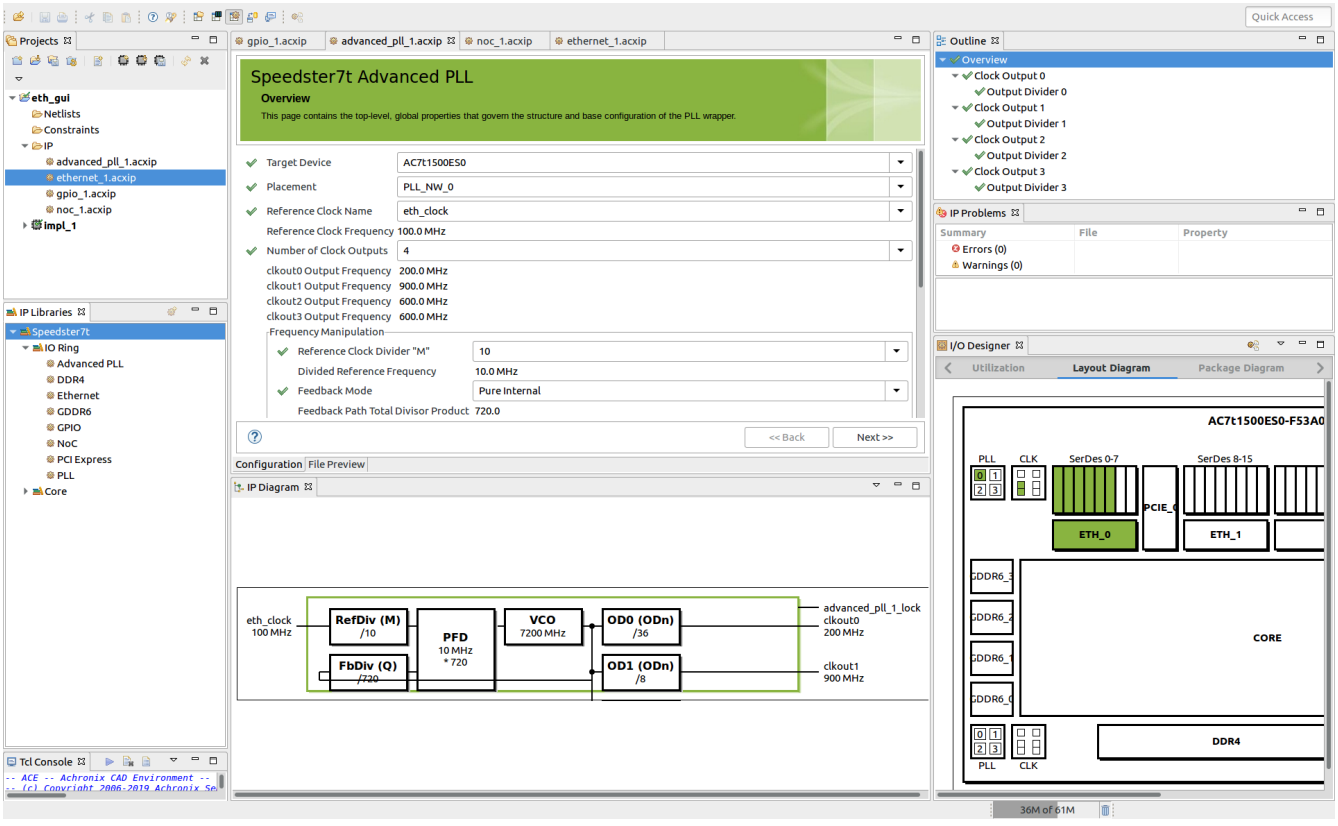


**Figure 9: *PLL IP Configuration in ACE I/O Designer***

# Step 4 – Configure the NoC

As the Ethernet Interface also uses the NoC for packet data, then the user also must instantiate NoC IP from the **IP Libraries** pane. The NoC requires a 200 MHz clock input, which in this example is provided from the PLL previously configured. Set the NoC **Reference Clock Name** to `noc_clk`.

It is possible to provide the NoC clock from another PLL or any other global clock source in the design.

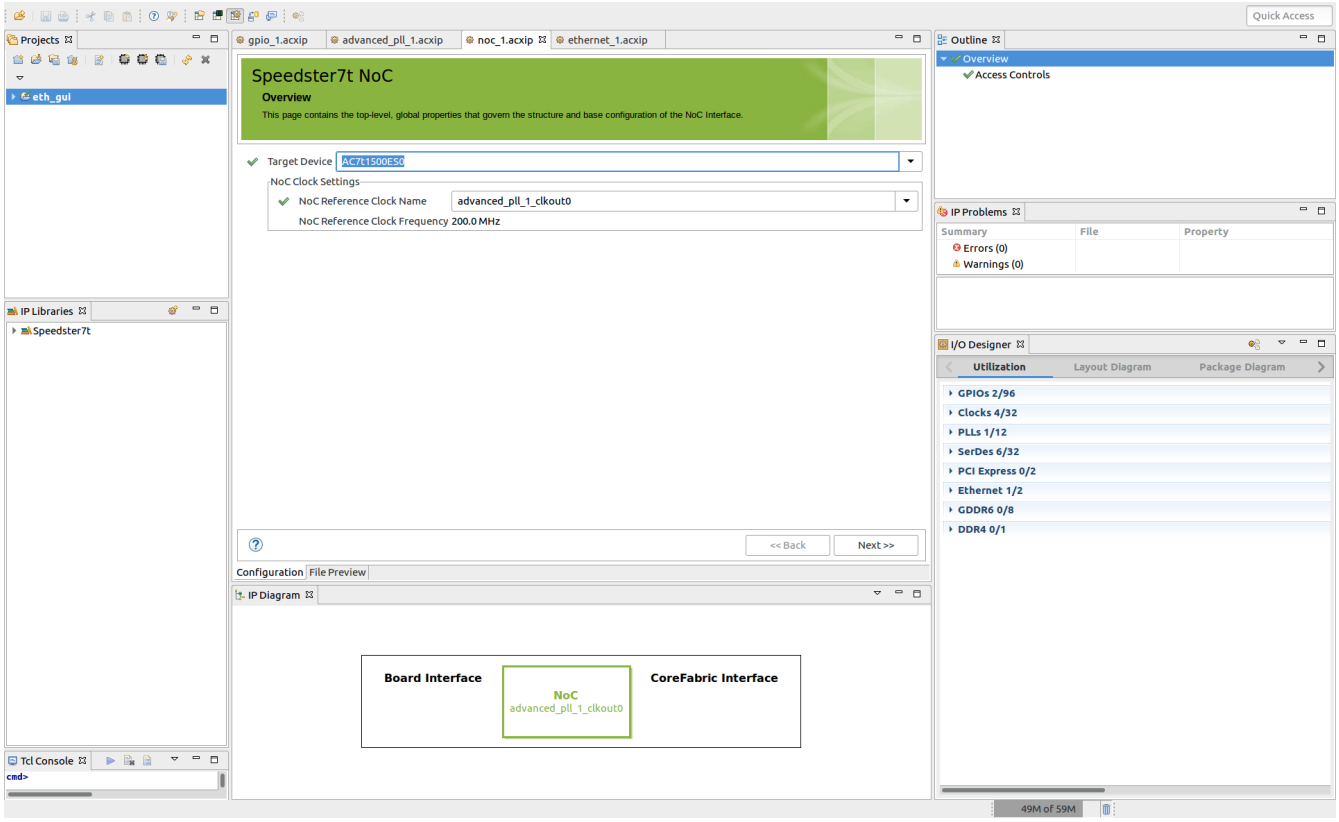


**Figure 10:** ***NoC IP Configuration in ACE I/O Designer***

# Step 5 – Configure the Interface

Next, the user must configure the Ethernet interface. From the IP Libraries select **Ethernet**. Select the desired **Ethernet MAC Placement** for the interface. In the **Lane Configurations** table, select the desired Ethernet channel operating modes. For this example, chose the following configuration:

- Lanes 0-3 – 200Gx4
- Lanes 4-5 – 100Gx2
- Lanes 6-7 – Disabled, (the MAC cannot support any further Ethernet channels)

Once the lanes are correctly specified, then the clocks from the PLL need to be connected to the **MAC Clock Settings**. The clock names use those specified in the previous PLL configuration, `eth_ref_clk, eth_ff1_clk, eth_ff2_clk.`

.

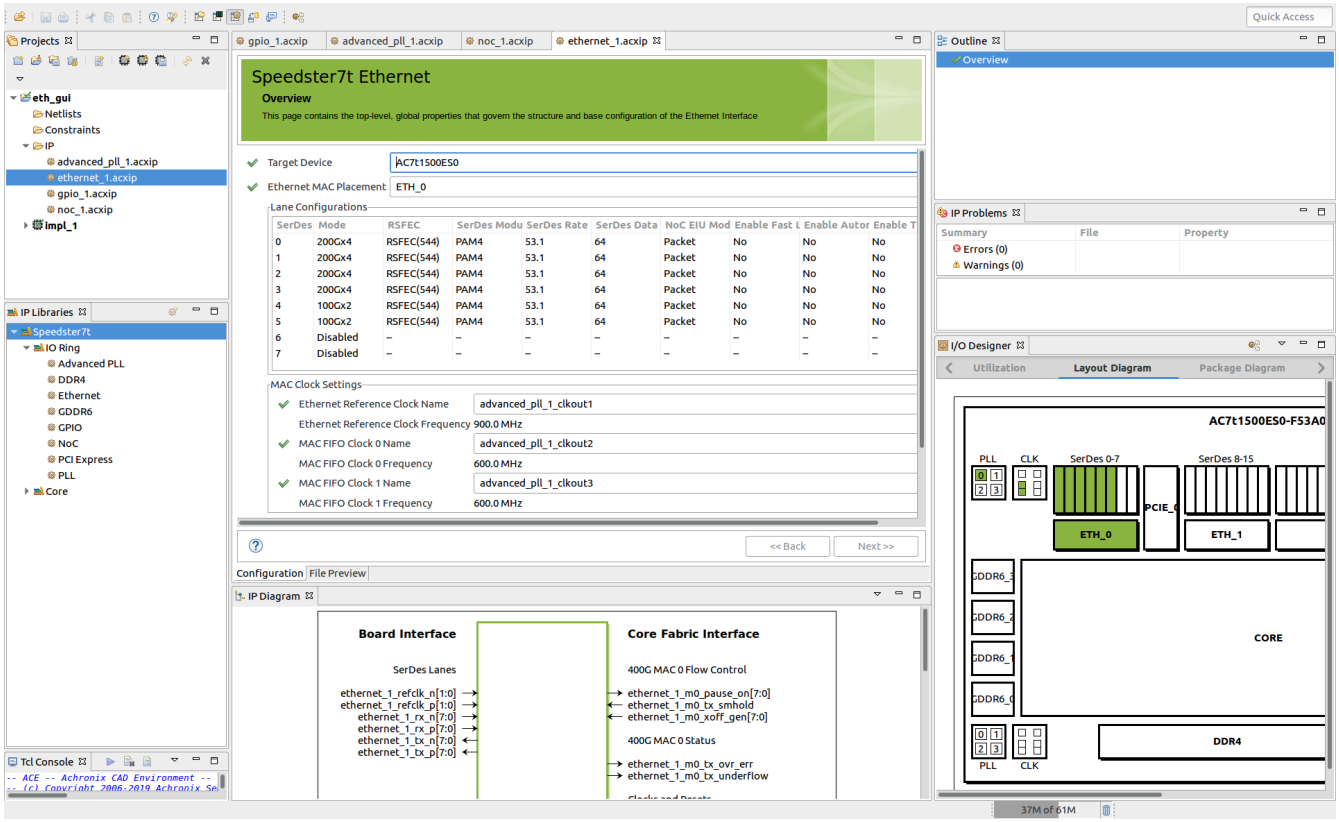


**Figure 11:** ***Ethernet Subsystem Configuration in ACE I/O Designer***

If the user intends to build a design with multiple Ethernet interfaces, then the existing Ethernet interface can be cloned. In the Project pane, under IP, select the Ethernet IP `.acxip` file. Right-clicking this file provides a **Clone IP** option. The cloned IP instance(s) can subsequently be configured individually.

# Step 6 – Check for Errors and Generate the Bitstream

After all the configuration options are selected, the **IP Problems** pane reports any errors or warnings that occurred with the configuration. If there are no errors or warnings reported, the user can be assured that the entire I/O interface with all the required IP are integrated properly. Once these checks are done, click **Generate IO Design Files** in the I/O Designer window to generate all the necessary files including the bitstream for the entire I/O ring. Clicking this icon also generates the necessary simulation models and placement files required for integrating with the core design.

This step completes the I/O ring configuration. The user can now switch to the core design. This core design will be integrated with the bitstream generated for the I/O interface to obtain the final full-chip integrated bitstream. This output file generation will be enabled in future ACE releases.

# Revision History

| Version | Date | Description |
|---|---|---|
| 0.9 | 10 Apr 2020 | Initial draft release. |