



# USXGMII Multirate Ethernet Core User Guide

---

UG-CORE-USXGMII-MULTIRATE-ETHERNET-v1.0

February 2026

[www.efinixinc.com](http://www.efinixinc.com)



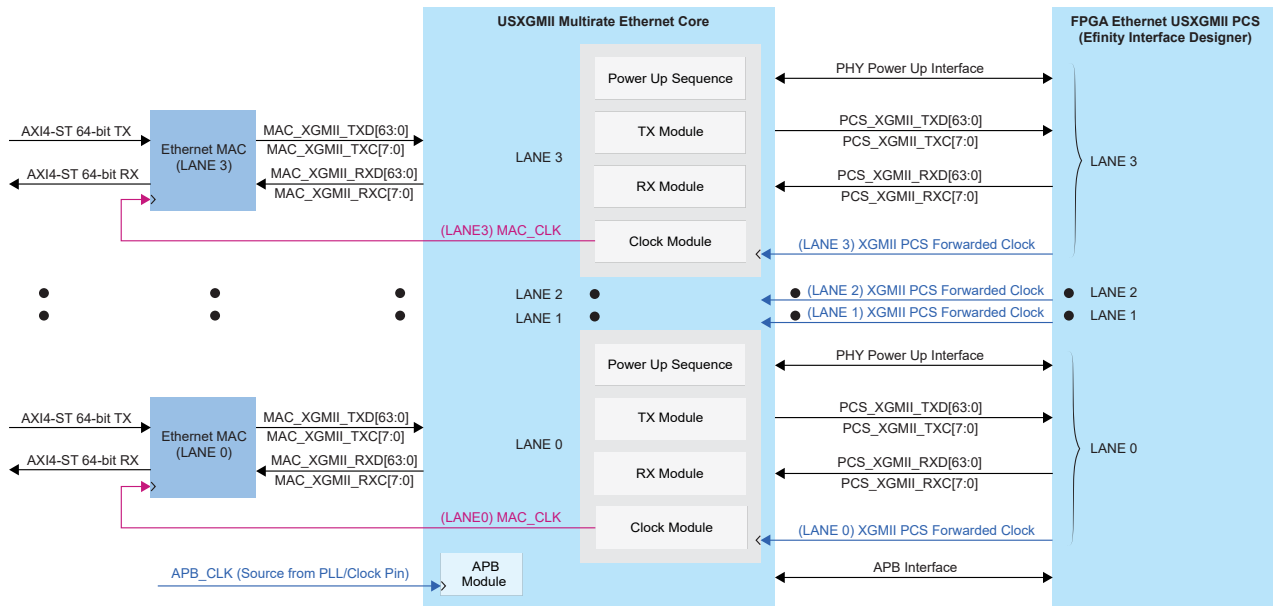
# Contents

<b>Introduction.....</b>	<b>3</b>
<b>Features.....</b>	<b>4</b>
<b>Device Support.....</b>	<b>4</b>
<b>Resource Utilization.....</b>	<b>5</b>
<b>Release Notes.....</b>	<b>6</b>
<b>Functional Description.....</b>	<b>6</b>
Block Diagram.....	6
Clock Sources.....	9
Reset Signals.....	12
Lx_INIT_RST_N Domain.....	14
APB_RST_N Domain.....	14
Lx_PCS_RST_N and Lx_MAC_RST_N Domain.....	15
Power Up Handshake with FPGA Transceiver.....	16
CDR-Locked-to-Data.....	16
Signal_ok.....	17
APB Configuration.....	18
Data Rate Change.....	18
Auto-Negotiation Clause 37 Operation.....	21
Clock Divider and Switching.....	27
Data Transfer.....	27
<b>Latency.....</b>	<b>28</b>
<b>Efinity: IP Catalog, Interface Designer, and Integration.....</b>	<b>29</b>
IP Catalog.....	29
Interface Designer.....	29
Top Level Module.....	29
IP Manager.....	32
Timing Constraints: SDC.....	33
<b>Customizing the USXGMII Multirate Ethernet.....</b>	<b>34</b>
<b>Ports.....</b>	<b>35</b>
<b>USXGMII Multirate Ethernet Example Design.....</b>	<b>37</b>
Generating the Example Design.....	41
Virtual I/O Debugger Settings.....	45
<b>USXGMII Multirate Ethernet Testbench.....</b>	<b>49</b>
<b>Acronyms and Abbreviations.....</b>	<b>50</b>
<b>Revision History.....</b>	<b>51</b>

# Introduction

The USXGMII Multirate Ethernet core , specifically `efx_usxgmii_an_37`, is a programmable soft IP that implements five flexible data rates: 10G, 5G, 2.5G, 1G, and 100M. Additionally, the USXGMII Multirate Ethernet core supports Ethernet Auto Negotiation Clause 37, that allows changes among the 5 data rates.

Figure 1: USXGMII Multirate Ethernet Interacts with Ethernet MAC and FPGA Ethernet USXGMII PCS



The USXGMII Multirate Ethernet core is designed to interact with both Ethernet MAC and the FPGA Ethernet USXGMII PCS. The data path of the USXGMII Multirate Ethernet core operates in the XGMII format, which consists of 64-bit data and 8-bit control. As shown in **Figure 1: USXGMII Multirate Ethernet Interacts with Ethernet MAC and FPGA Ethernet USXGMII PCS** on page 3, the data path interaction among the Ethernet MAC, USXGMII Multirate Ethernet core, and FPGA Ethernet USXGMII PCS is on a per-lane basis. The configuration of the USXGMII Multirate Ethernet core (to change the data rate and/or enable auto-negotiation) operates along the APB interface and is on a per-quad basis.

Use the IP Manager to select IP, customize it, and generate files. The USXGMII Multirate Ethernet core has an interactive wizard to help you set parameters. The wizard also has options to create a testbench and/or example design targeting an Efinix® development board.

## Features

- Compliant with IEEE Std. 802.3-2008 specification
- Supports IEEE Std. 802.3 Clause 37 Auto-negotiation
- Supports flexible data rates of 10G, 5G, 2.5G, 1G, and 100M
- Operates in full-duplex mode only

## Device Support

USXGMII Multirate Ethernet core is supported in Titanium and Topaz FPGA with transceiver. Refer to the [Titanium Selector Guide](#) and [Topaz Selector Guide](#) to get the latest device list that supports transceivers.

# Resource Utilization



**Note:** The resources and performance values provided are based on some of the supported FPGAs. These values are just guidance and may change depending on the device resource utilization, design congestion, and user design.

**Table 1: Resource Utilization**

FPGA	Mode	Logic Elements (Logic, Adders, Flipflops, etc.)	Memory Blocks	DSP Blocks	Efinity® Version
Ti375N1156 C4	APB Module	122/ 362,880 (0.03%)	0/ 2,688 (0%)	0/ 1,344 (0%)	2025.2
	<b>FIFO RAM Mode</b> Synthesize Lane 0 = <b>Yes</b> Synthesize Power Up Module in Lane 0 = <b>Yes</b> Synthesize Tx Pipeline Registers in Lane 0 = <b>Yes</b> Set TX FIFO Type in Lane 0 = <b>RAM</b> Synthesize RX Pipeline Registers in Lane 0 = <b>Yes</b> Set RX FIFO Type in Lane 0 = <b>RAM</b>	1,667/ 362,880 (0.46%)	8/ 2,688 (0.30%)	0/ 1,344 (0%)	
	<b>FIFO Registers Mode</b> Synthesize Lane 0 = <b>Yes</b> Synthesize Power Up Module in Lane 0 = <b>Yes</b> Synthesize TX Pipeline Registers in Lane 0 = <b>Yes</b> Set TX FIFO Type in Lane 0 = <b>Registers</b> Synthesize RX Pipeline Registers in Lane 0 = <b>Yes</b> Set RX FIFO Type in Lane 0 = <b>Registers</b>	4654/ 362,880 (1.28%)	0/ 2,688 (0%)	0/ 1,344 (0%)	

The resource utilization data is derived based on the synthesis of Q1 Lane 0.



**Note:** Refer to the [Customizing the USXGMII Multirate Ethernet](#) on page 34 for the full list of configurations and descriptions.

<sup>(1)</sup> System Verilog 2005.

## Release Notes

You can refer to the IP Core Release Notes for more information about the IP core changes. The IP Core Release Notes are available in the [Efinity page of the Support Center](#) under each Efinity software release version.



**Note:** You must be logged in to the Support Center to view the IP Core Release Notes.

## Functional Description

The USXGMII Multirate Ethernet core does the following functions:

- Interact with Ethernet MAC to operate with up to 5 data rates of 10G, 5G, 2.5G, 1G, and 100M.
- Supports Auto-Negotiation Clause 37.

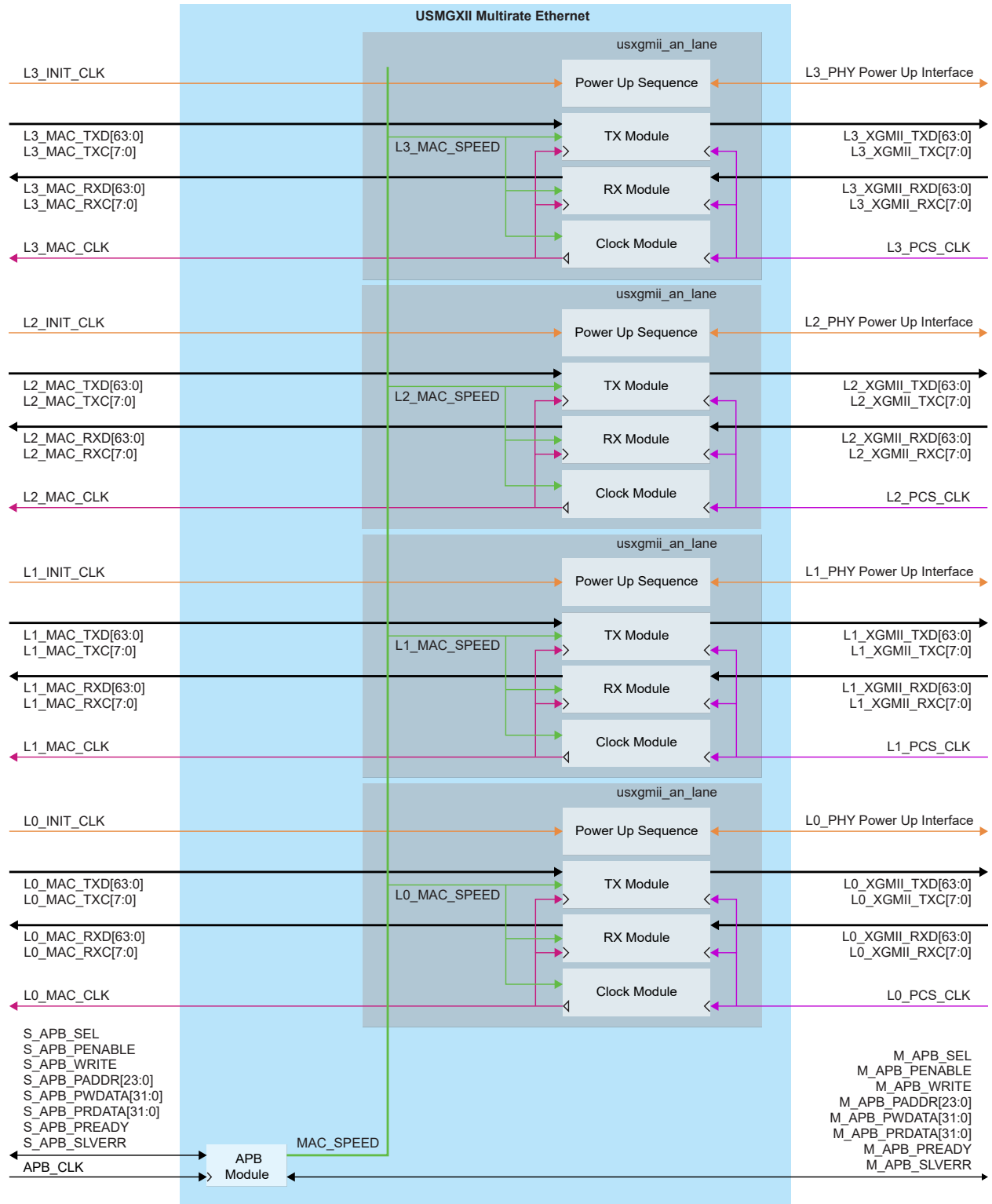
## Block Diagram

The USXGMII Multirate Ethernet core or `efx_usxgmi_an_37` comprises following modules:

- APB module
- USXGMII Multirate Ethernet native (`efx_usxmii_an_lane`)

Figure 2: USXGMII Multirate Ethernet Core Sub-Modules on page 7 depicts the modules inside the USXGMII Multirate Ethernet core.

Figure 2: USXGMII Multirate Ethernet Core Sub-Modules



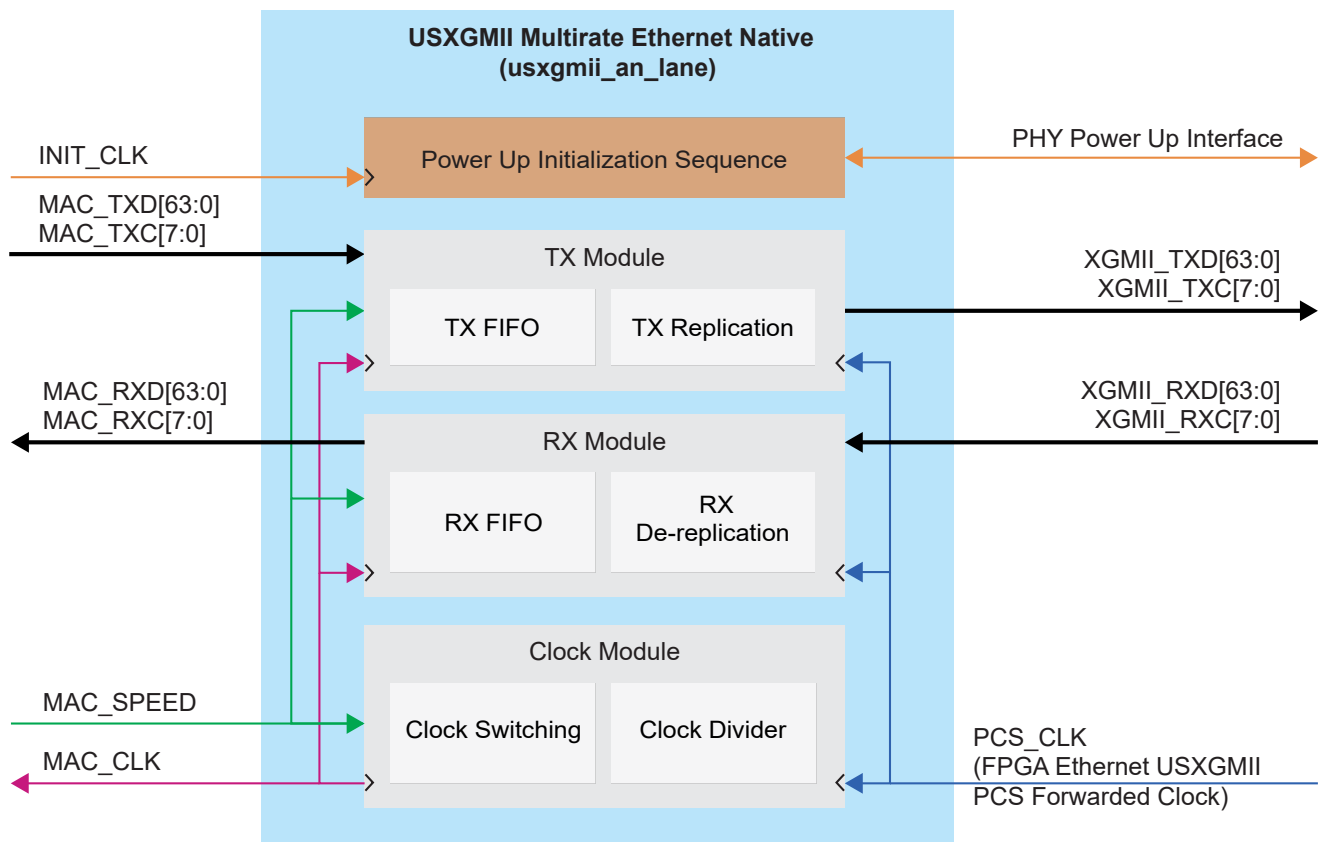
The APB module facilitates the configuration in the FPGA Ethernet USXGMII PCS to enable rate change and Auto-Negotiation Clause 37. The APB module operates on a per-quad basis.

The USXGMII Multirate Ethernet native, `efx_usxgmii_an_lane`, facilitates the necessary replication, de-replication, clock division, and clock switching for data transfer. The synthesis of USXGMII Multirate Ethernet native is optional and configurable on a per lane basis. Each USXGMII Multirate Ethernet native comprises the following modules:

- Optional power-up sequence to handshake with the FPGA transceiver.
- TX module, comprising a TX replication block and a FIFO.
- RX module, comprising a RX de-replication block and a FIFO.
- Clock module, comprising a clock divider and a clock switching block.

**Figure 3: USXGMII Multirate Ethernet Native Modules** on page 8 depicts the sub-modules inside the USXGMII Multirate Ethernet native.

*Figure 3: USXGMII Multirate Ethernet Native Modules*



**Note:** In the subsequent chapters, all descriptions and illustrations are on a per lane basis by default, unless specified otherwise. The prefix '`Lx`' and the term 'Lane X' denote the lane number.

## Clock Sources

The USXGMII Multirate Ethernet core has up to 13 clock sources and domains:

- APB\_CLK (per quad)
- Lx\_INIT\_CLK (per lane)
- Lx\_PCS\_CLK (per lane)
- Lx\_MAC\_CLK (generated, output port)

APB\_CLK facilitates the configuration in the FPGA transceiver to enable rate change and Auto-Negotiation Clause 37. This APB\_CLK operates on a per-quad basis, which can be sourced from the PLL or any clock input pin and has a frequency range of 50 MHz to 200 MHz.

Lx\_INIT\_CLK facilitates the power-up sequence of the FPGA transceiver PHY and can be sourced from the PLL or any clock input pin. It runs at a speed  $\leq 50$  MHz. The synthesis of the power-up sequence is optional; therefore, this clock is only applicable when the **Synthesize Power Up Module in Lane X** is set to **Yes**. This Lx\_INIT\_CLK operates on a per lane basis in the USXGMII Multirate Ethernet core, but you may source them to be common.

Lx\_PCS\_CLK is sourced from the FPGA transceiver's forwarded clock to ensure source synchronicity and migrate the similar clock PPM from the FPGA transceiver. The Lx\_MAC\_CLK is the divided clock generated from the Lx\_PCS\_CLK. Both Lx\_MAC\_CLK and Lx\_PCS\_CLK operate on a per lane basis.

- During 10G, 1G, or 100M data rate, the TX replication and RX de-replication modules are bypassed and Lx\_MAC\_CLK is sourced directly from Lx\_PCS\_CLK.
- During 5G and 2.5G data rates, the Lx\_PCS\_CLK is set to operate at 156.25 MHz. The clock module in the USXGMII Multirate Ethernet core divides down the 156.25 MHz Lx\_PCS\_CLK to output Lx\_MAC\_CLK at 78.125 MHz and 39.0625 MHz respectively.

To ensure timing closure, Lx\_MAC\_CLK domain is separated and is made to be asynchronous from Lx\_PCS\_CLK with a FIFO implementation, each in the TX and RX path as shown in **Figure 4: USXGMII Multirate Ethernet Core Clock Domain** on page 10.

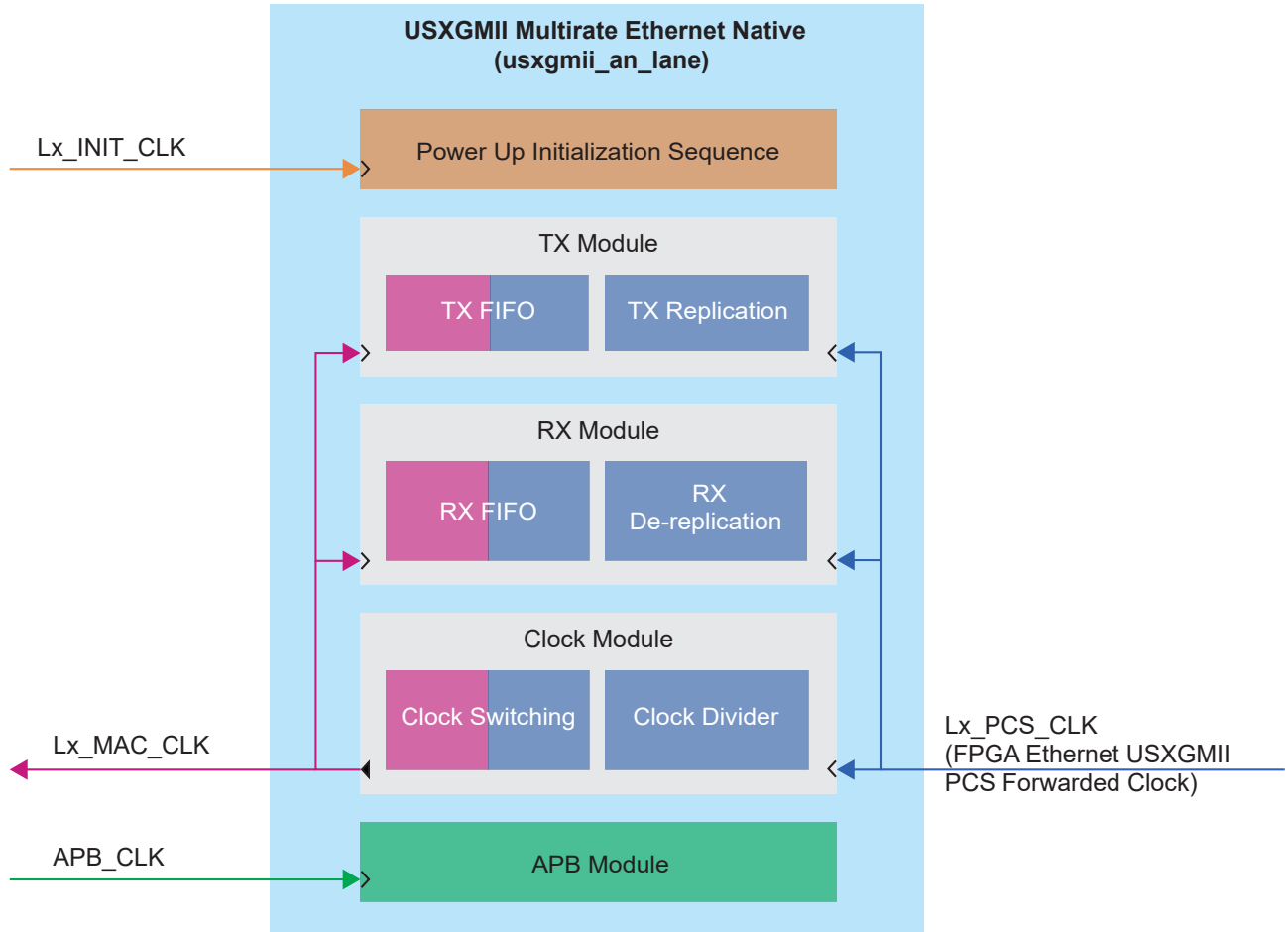
A Lx\_PCS\_CLK is independent on its own lane, meaning that L0\_PCS\_CLK, L1\_PCS\_CLK, L2\_PCS\_CLK, and L3\_PCS\_CLK are asynchronous to one another. The same applies to Lx\_MAC\_CLK, where L0\_MAC\_CLK, L1\_MAC\_CLK, L2\_MAC\_CLK, and L3\_MAC\_CLK are asynchronous to one another.

**Figure 4: USXGMII Multirate Ethernet Core Clock Domain** on page 10 depicts the clock domains in USXGMII Multirate Ethernet core.



**Important:** Lx\_MAC\_CLK is generated from Lx\_PCS\_CLK. Hence, you need to add create\_generated\_clock constraint into the SDC. Refer to the SDC file in the example design.

Figure 4: USXGMII Multirate Ethernet Core Clock Domain



**Notes:** Lx = Lane 0, Lane 1, Lane 2, Lane 3

- (1) ■ `Lx_INIT_CLK` domain (per lane basis, can be simplified to be common among all lanes)
- (2) ■ `Lx_PCS_CLK` domain (per lane basis)
- (3) ■ `Lx_MAC_CLK` (generated from `PCS_CLK`, per lane basis)
- (4) ■ `APB_CLK` domain

**Table 2: USXGMII Multirate Ethernet Core Clock Sources**

<b>Clock Name</b>	<b>Direction</b>	<b>Frequency</b>	<b>Description</b>
Lx_PCS_CLK	Input	156.25 MHz (10G) 15.625 MHz (1G) 1.5625 MHz (100M)	Per lane basis. Sourced from the FPGA transceiver forwarded clock, which is a synchronous source, and to eliminate the PPM clock.  The same forwarded clock is also used to clock the FPGA Ethernet USXGMII PCS.
Lx_MAC_CLK	Output	156.25 MHz (10G) 78.125 MHz (5G) 39.0625 MHz (2.5G) 15.625 MHz (1G) 1.5625 MHz (100M)	Per lane basis. Divided clock generated from Lx_PCS_CLK to support 2.5G and 5G data rate. During 100M, 1G, and 10G data rates, Lx_MAC_CLK is sourced directly from Lx_PCS_CLK.
APB_CLK	Input	50 MHz - 200 MHz	Per quad basis. To facilitate the configuration of the FPGA transceiver to enable rate change and Auto-Negotiation Clause 37.
Lx_INIT_CLK	Input	≤ 50 MHz	To facilitate the power-up sequence of the FPGA transceiver PHY. Can be sourced from PLL or CLKIN pin. Per lane basis but can be connected as common.

## Reset Signals

The USXGMII Multirate Ethernet core has a total of 13 reset domain signals.

- APB\_RST\_N (per quad)
- Lx\_INIT\_RST\_N (per lane)
- Lx\_PCS\_RST\_N (per lane)
- Lx\_MAC\_RST\_N (per lane)

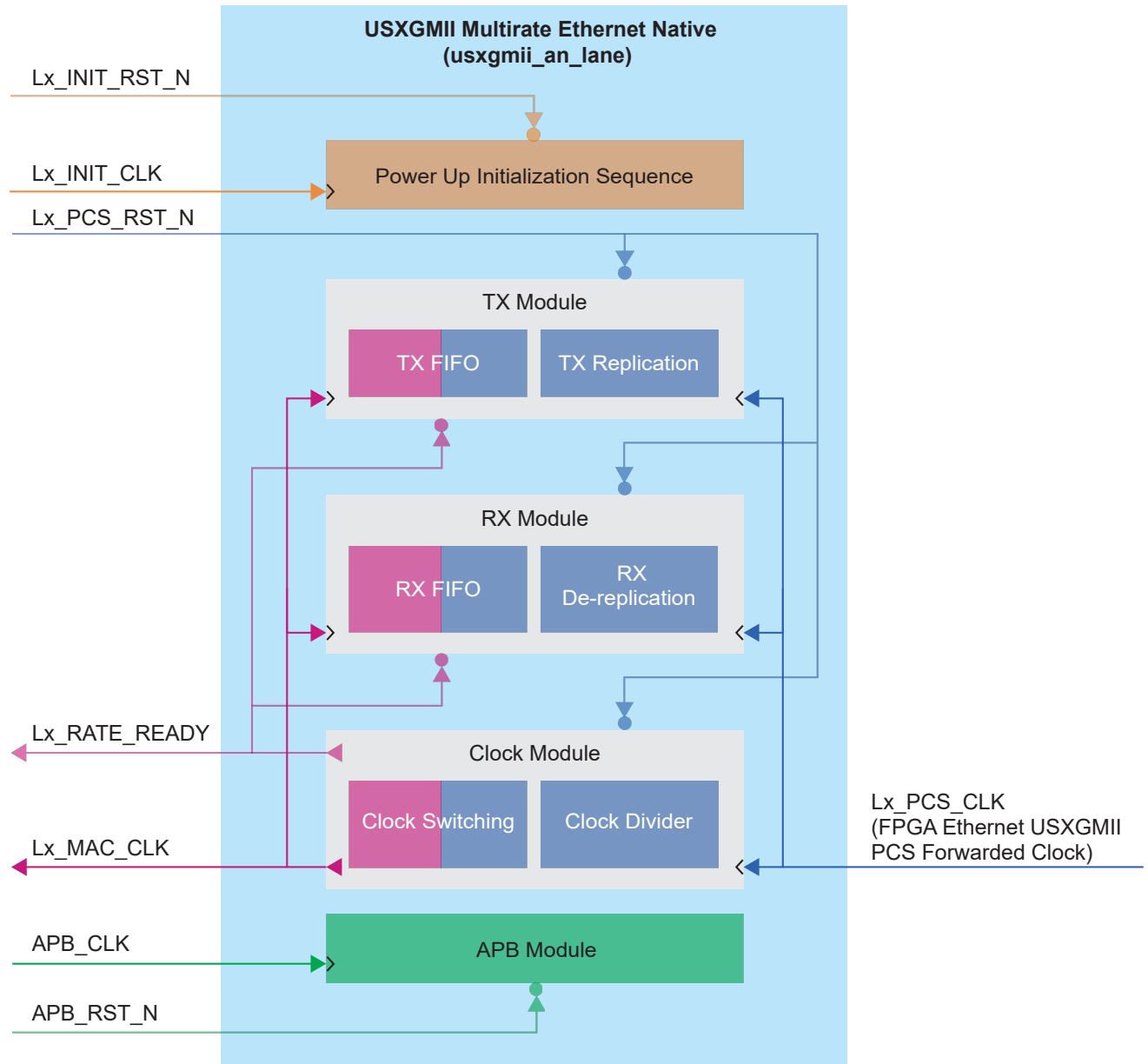
Lx\_INIT\_RST\_N, Lx\_PCS\_RST\_N, and APB\_RST\_N are reset input signals. During the FPGA power-up, all these reset input signals are initialized to 0 to reset the entire USXGMII Multirate Ethernet core to the initial known states.

These reset input signals are asynchronous reset, but the deassertion of each reset signal is synchronous. For APB\_RST\_N, there is an internal reset synchronizer that aligns the deassertion of APB\_RST\_N to APB\_CLK. For Lx\_INIT\_RST\_N, an internal reset synchronizer aligns the deassertion of Lx\_INIT\_RST\_N to Lx\_INIT\_CLK. For Lx\_PCS\_RST\_N, an internal reset synchronizer aligns the deassertion of Lx\_PCS\_RST\_N to Lx\_PCS\_CLK. Each reset synchronizer has a latency of 2 to 3 clock cycles.

Lx\_RATE\_READY is an output reset signal that indicates the readiness of the clock switching. When paired with Lx\_MAC\_CLK, the Lx\_RATE\_READY serves as a source synchronous reset input signal to the user's MAC.

The **Figure 5: USXGMII Multirate Ethernet Core Reset Domains** on page 13 shows the reset domains in the USXGMII Multirate Ethernet core.

Figure 5: USXGMII Multirate Ethernet Core Reset Domains



- Notes:** Lx = Lane 0, Lane 1, Lane 2, Lane 3
- (1) ■  $Lx\_INIT\_RST\_N$  domain (per lane basis)
  - (2) ■  $Lx\_PCS\_RST\_N$  domain (per lane basis)
  - (3) ■  $Lx\_MAC\_RST\_N$  (triggered by  $Lx\_RATE\_READY$ , per lane basis)
  - (4) ■  $APB\_RST\_N$  domain

## *Lx\_INIT\_RST\_N Domain*

The `Lx_INIT_CLK` must be sourced and actively running during the power-up stage. Upon the assertion of `IN_USER`, you need to deassert `Lx_INIT_RST_N` to start the power-up initialization sequence with the FPGA PHY. The `Lx_INIT_RST_N` signal remains de-asserted throughout the USXGMII Multirate Ethernet core operations. Details of the power-up sequence are being described in [Power Up Handshake with FPGA Transceiver](#) on page 16.

The synthesis of the power-up initialization sequence module is optional, governed by the user-defined **Synthesize Power Up Module in Lane X**. When you configure the **Synthesize Power Up Module in Lane X** to **No**, you need to create your module to perform the power-up handshake with the FPGA transceiver. When the **Synthesize Power Up Module in Lane X** is set to **No**, no port interface is associated with the power-up initialization sequence module in the generated USXGMII Multirate Ethernet core.

Although the `Lx_INIT_CLK` may be common among all lanes, the `Lx_INIT_RST_N` reset input signal is on a per lane basis and asynchronous among all lanes. This asynchronous property allows independent per lane reset in USXGMII Multirate Ethernet core and USXGMII Multirate Ethernet Native.

## *APB\_RST\_N Domain*

Upon the completion of the power-up initialization sequence with the FPGA PHY (indicated by the assertion of signal `Lx_phy_init_done`), you must deassert `APB_RST_N`. It must remain deasserted onwards, so that the APB module in the USXGMII Multirate Ethernet core stays operational to facilitate rate change and Auto-Negotiation Clause 37.



**Note:** When the **Synthesize Power Up Module in Lane X** is set to **No**, the port `Lx_phy_init_done` becomes unavailable. You must create your signal to indicate the completion of the power-up initialization sequence with the FPGA PHY.

When `APB_RST_N` is asserted, it resets the `MAC_Speed` register to 10G, causing `Lx_MAC_CLK` to revert to running at 156.25 MHz. The assertion of `APB_RST_N` only resets the APB module in the USXGMII Multirate Ethernet core, but not the APB Interconnect in the FPGA Ethernet USXGMII PCS. Hence, to prevent misalignment of the MAC speed between the FPGA core and the FPGA Ethernet USXGMII PCS, `APB_RST_N` must remain deasserted after the initial power-up.



**Important:** If the MAC speed misaligns between the USXGMII Multirate Ethernet core and the FPGA Ethernet USXGMII PCS, you need to explicitly configure the MAC speed to the desired speed to realign the MAC speed between the USXGMII Multirate Ethernet core and the FPGA Ethernet USXGMII PCS.



**Warning:** Any assertion of `APB_RST_N` amid APB handshake may jeopardize the assertion of `PREADY` and may cause a deadlock to occur in the APB Interconnect. When a deadlock (due to APB) occurs, you need to restart the FPGA.

## *Lx\_PCS\_RST\_N and Lx\_MAC\_RST\_N Domain*

$Lx\_PCS\_RST\_N$  must be initialized to 0 to reset the TX and RX modules of the USXGMII Multirate Ethernet core to known initial states. When  $Lx\_PCS\_RST\_N$  is asserted to 0, the USXGMII Multirate Ethernet core is in reset mode and transmits continuous IDLE packets.

The output reset signal,  $Lx\_RATE\_READY$  denotes the readiness of the  $Lx\_MAC\_CLK$ . When the  $Lx\_RATE\_READY$  signal deasserts to 0, it signals that the USXGMII Multirate Ethernet core is undergoing a clock switching, and the  $Lx\_MAC\_CLK$  is not ready for use. When the clock switching completes and  $Lx\_MAC\_CLK$  stabilizes, the  $Lx\_RATE\_READY$  signal asserts to 1.

The  $Lx\_RATE\_READY$  signal operates in  $Lx\_MAC\_CLK$  domain and on a per lane basis. The  $Lx\_MAC\_RST\_N$  domain is triggered by the  $Lx\_RATE\_READY$  signal, and is asynchronous for all lanes. Paired with  $Lx\_MAC\_CLK$ , the  $Lx\_RATE\_READY$  signal serves as a synchronous source reset input signal to the user's MAC.

When  $Lx\_PCS\_RST\_N$  is asserted, it resets the data path modules. Additionally, it resets the clock divider and switching module. When  $Lx\_PCS\_RST\_N$  is asserted,  $Lx\_MAC\_CLK$  is sourced directly from a forwarded clock, which may run at 156.25 MHz, 15.625 MHz, or 1.5625 MHz.

## Power Up Handshake with FPGA Transceiver

Before starting the USXGMII Multirate Ethernet core, a power-up handshake must be performed with the FPGA transceiver. The USXGMII Multirate Ethernet core includes an optional built-in power-up initialization sequence module to handshake with the FPGA transceiver PHY.

The synthesis of the power-up initialization sequence module is optional and controlled by the user-defined **Synthesize Power Up Module in Lane X**. When you configure **Synthesize Power Up Module in Lane X** to **No**, you must create your module to perform the power-up handshake with the FPGA transceiver.

For the details on the power-up sequence in the FPGA transceiver, refer to the power-up sequence chapter of the [Titanium Ethernet 10GBase-KR User Guide](#). **Figure 6: Power Up Handshake with the FPGA Transceiver PHY and PCS Per Lane** on page 17 shows the power-up handshake between the USXGMII Multirate Ethernet core and the FPGA transceiver.

The power-up sequence between the USXGMII Multirate Ethernet core and the FPGA transceiver PHY is clocked by `Lx_INIT_CLK`. The assertion of `Lx_phy_init_done` indicates that the FPGA transceiver PHY has achieved CDR-locked-to-data stage.

### *CDR-Locked-to-Data*

To achieve the CDR-locked-to-data stage, the RX path of the FPGA transceiver PHY must receive continuous IDLE/DATA from its Link Partner. Hence, upon the assertion of `PMA_CMN_READY`, you must ensure that the Link Partner sends continuous IDLE/DATA to the RX path of the FPGA transceiver PHY.

During the power-up, the FPGA Ethernet USXGMII PCS's `Lx_PCS_RST_N_TX` must deassert to allow TX path to continuously send IDLE packets to the Link Partner's RX path.

If the 3 PCS reset signals are independently driven, upon the assertion of `PMA_CMN_READY`, you may deassert the FPGA Ethernet USXGMII PCS's `Lx_PCS_RST_N_TX` to 1, while maintaining the USXGMII Multirate Ethernet core's `Lx_PCS_RST_N` to 0. Once the `Lx_phy_init_done` is asserted, you must deassert both USXGMII Multirate Ethernet core's `Lx_PCS_RST_N` and the FPGA Ethernet USXGMII PCS's `Lx_PCS_RST_N_RX` to 1.



**Note:** The 3 PCS reset signals refer to:

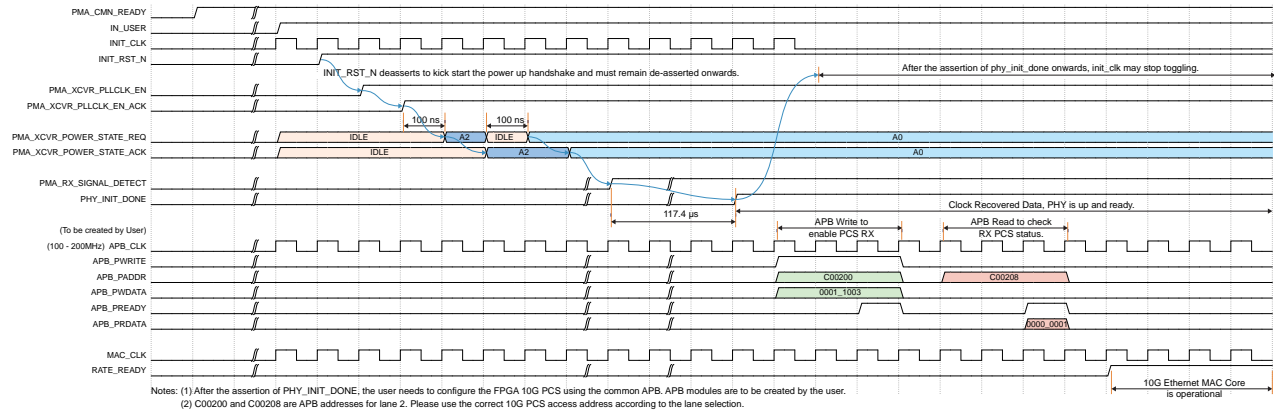
- USXGMII Multirate Ethernet core's `Lx_PCS_RST_N`.
- FPGA Ethernet USXGMII PCS's `Lx_PCS_RST_N_TX`.
- FPGA Ethernet USXGMII PCS's `Lx_PCS_RST_N_RX`.

If the 3 PCS reset signals are driven from the same source, `Lx_PCS_RST_N` must deassert to 1 upon the assertion of `PMA_CMN_READY`.

## Signal\_ok

After the assertion of `Lx_phy_init_done`, you must create an APB driver to configure `signal_ok`, based on the register mapping in the Register Map chapter (refer to `control_register` table, `status_register` table, and `interrupt_status_register` table) of the **Titanium Ethernet 10GBase-KR User Guide**. The APB interface is a common interface that is shared across all 4 lanes within a quad. The `signal_ok` enables the RX path in the FPGA Ethernet USXGMII PCS. **Figure 6: Power Up Handshake with the FPGA Transceiver PHY and PCS Per Lane** on page 17 includes an illustration of APB Write to enable PCS RX in Lane 2.

**Figure 6: Power Up Handshake with the FPGA Transceiver PHY and PCS Per Lane**



Any deassertion of `Lx_phy_init_done` signals that the current CDR-locked-to-data condition has lost lock. When this occurs, you must set the `signal_ok` to 0 to disable the RX path in the FPGA Ethernet USXGMII PCS. Upon the recovery of `Lx_phy_init_done`, you must reconfigure the `signal_ok` to 1 to re-enable the RX path in the FPGA Ethernet USXGMII PCS.

When the `PMA_CMN_READY` signal deasserts, it indicates that the FPGA transceiver PHY is no longer functional. If this occurs, all the per-lane reset input signals (i.e., `Lx_INIT_RST_N` and `Lx_PCS_RST_N`) must be asserted. You must set the `signal_ok` to 0 to disable the RX path in the FPGA Ethernet USXGMII PCS. Once the FPGA PHY recovers from reset, you must repeat the entire power-up handshake by deasserting the `Lx_INIT_RST_N` to start over the power-up sequence.

The APB interface is unaffected by `Lx_phy_init_done` or `PMA_CMN_READY` signal. Hence, `APB_RST_N` should remain deasserted after the initial power-up.

## APB Configuration

The APB module in the USXGMII Multirate Ethernet core facilitates the APB configuration to enable data rate change and Auto-Negotiation Clause 37. The APB module also contains a register to store the data rate.

The ports that connect to the user's MAC interface are prefixed with 's\_apb\_' while the ports that connect the FPGA APB are prefixed with 'm\_apb'.

Refer to 10G PCS Access table in the [Titanium Ethernet 10GBase-KR User Guide](#) for the APB addresses to access the per-lane Ethernet 10G PCS.

### Data Rate Change

During power-up, when enabling the RX Path with the `signal_ok`, it is mandatory to configure the `usx_speed` register to 10 Gbps as the initial data rate. Refer to the Register Map in the [Titanium Ethernet 10GBase-KR User Guide](#) to configure the data rate change and/or initiate the Auto-Negotiation Clause 37.

With reference to the `control_register` table in the Register Map chapter of the [Titanium Ethernet 10GBase-KR User Guide](#), refer to [Table 3: APB Control Register for Data Rate Configurations](#) on page 18 to configure the `control_register` [16:14] for the extended support of 5G and 2.5G data rates.



#### Note:

- The terms 'MAC speed' and 'data rate' are interchangeable. Both terms denote the data rate at the MAC interface and operate on the `Lx_MAC_CLK`.
- The name `usx_speed` register is also referred as `mac_speed` register.
- The output port `lx_mac_speed` reflects the configuration of the `mac_speed` register.

**Table 3: APB Control Register for Data Rate Configurations**

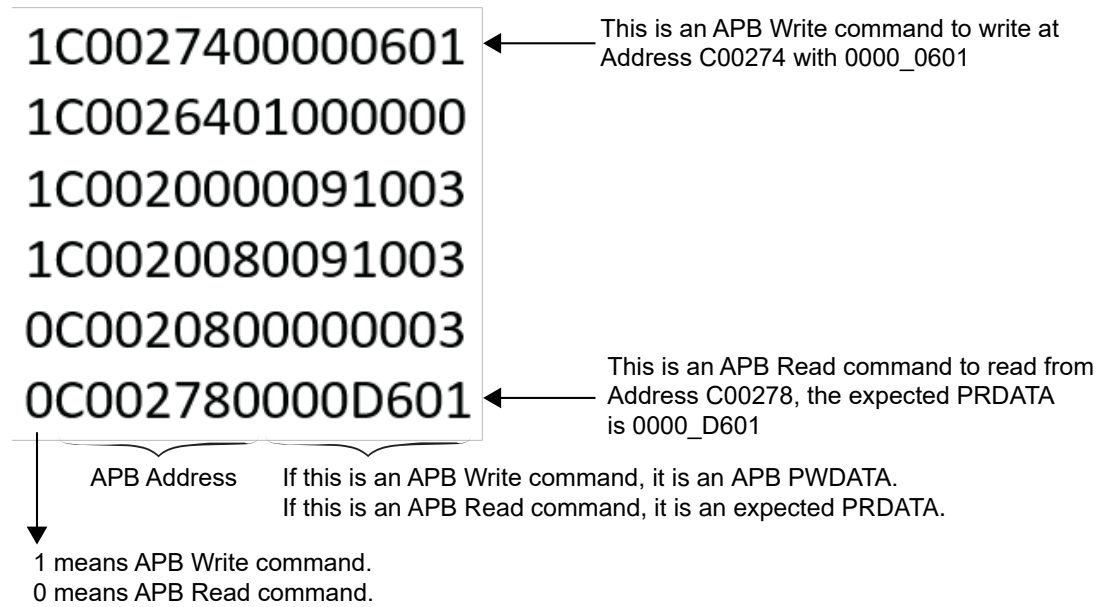
Bit	Name	Description	Type	Reset
16:14	<code>usx_speed</code>	USXGMII speed. The USXGMII speed must match the data rate of the MAC device. 'b100: 10 Gbps 'b011: 5 Gbps 'b010: 2.5 Gbps 'b001: 1 Gbps 'b000: 100 Mbps	RW	0x4



**Note:** In the following chapters, the mechanism of the auto-negotiation to change the data rate revolves between Q1\_L2 as an active device and Q3\_L2 as a link partner.

In the following illustrations of data rate change and auto-negotiation, the APB configuration is represented in a format as shown in **Figure 7: Format of APB Configuration** on page 19.

Figure 7: Format of APB Configuration



**Table 4: Configuring 10G as the initial data rate** on page 19 depicts an example of an active device (Q1\_L2) setting the initial data rate.

Table 4: Configuring 10G as the initial data rate

No.	Active_Device (Q1_L2)
1	Power-Up Initialization Sequence
2	Upon assertion of Q1_L2_PHY_INIT_DONE, Q1_L2 starts APB configuration, mainly to set signal_ok.
3	1_C00200_00011003 Q1_L2 sets signal_ok, sets the usx_speed to 10G as the initial mac_speed.
4	wait(Q1_L2_BLOCK_LOCK) Q1_L2 waits for BLOCK_LOCK (output from Interface Designer)
5	0_C00208_00000003 Upon the assertion of Q1_L2_BLOCK_LOCK, Q1_L2 reads its status register to check for faults/errors.
6	The assertion of Q1_L2_BLOCK_LOCK indicates that the RX Channel has established the RX word boundary and DATA packet starts to transfer.

**Table 5: Changing to new data rate** on page 20 depicts an example of an active device (Q1\_L2) changing data to new 5G data rate.

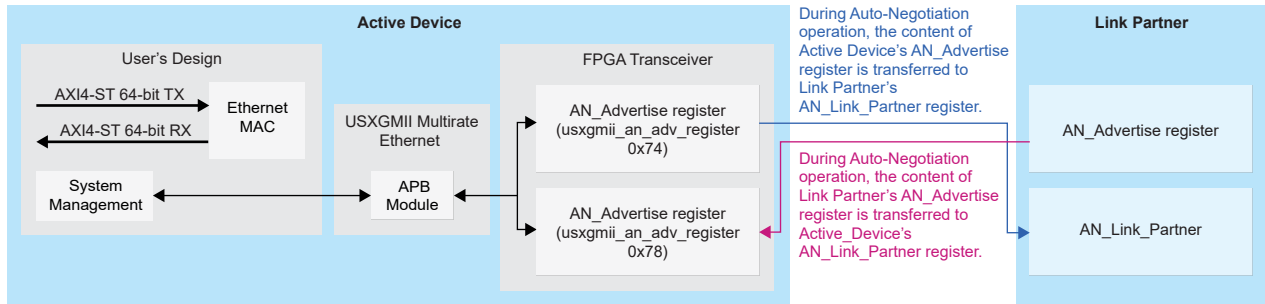
*Table 5: Changing to new data rate*

No.	Active_Device (Q1_L2)
1	Data transfer is ongoing.
2	To ensure no data loss, Q1_L2 stops data transfer. In replacement, Q1_L2 transfers IDLE packets.
3	1_C00200_0000D003 Q1_L2 sets 5G as the new data rate.
4	wait(Q1_L2_RATE_READY) Q1_L2 waits for reassertion of RATE_READY (denotes the readiness of clock switching)
5	The assertion of Q1_L2_RATE_READY indicates that clock switching has been completed and stabilized and DATA packet starts to transfer.

## Auto-Negotiation Clause 37 Operation

The APB module in the USXGMII Multirate Ethernet core facilitates the configuration to enable Auto-Negotiation Clause 37 in the FPGA transceiver. **Figure 8: Auto-Negotiation Clause 37 between an Active Device and a Remote Link Partner** on page 21 shows the operation of the Auto-Negotiation Clause 37 between an active device and a remote link partner.

Figure 8: Auto-Negotiation Clause 37 between an Active Device and a Remote Link Partner



## Auto-Negotiation Message

The active device and the remote link partner advertise their capability based on the auto-negotiation message, as depicted in [Table 6: Auto-Negotiation Message](#) on page 22.

**Table 6: Auto-Negotiation Message**

Bit	Description	Default
15	1 = Link Up 0 = Link Down	0
14	Auto-Negotiation Acknowledge	0
13	Reserved	0
12	1 = Full Duplex 0 = Half Duplex	1
11:9	Speed 000 = 10 Mbps 001 = 100 Mbps 010 = 1 Gbps 011 = 10 Gbps 100 = 2.5 Gbps 101 = 5 Gbps 110 = Reserved 111 = Reserved	010
8	1 = EEE capability is supported 0 = EEE capability is not supported	0
7	1 = EEE clock stop capability is supported 0 = EEE clock stop capability is not supported	0
6:1	Reserved	0
0	1 = XGMII 0 = SGMII	1

## Auto-Negotiation Handshake

**Table 7: Active Device enables Auto-Negotiation upon Power-Up** on page 23 illustrates an example of an Active Device (Q1\_L2) that enables the auto-negotiation handshake after completing the power-up initialization sequence, to obtain the Link Partner's capabilities.

**Table 7: Active Device enables Auto-Negotiation upon Power-Up**

Active_Device (Q1_L2)	Link_Partner (Q3_L2)
Power-Up Initialization Sequence	
Upon assertion of Q1_L2_PHY_INIT_DONE, start APB configuration, mainly to set signal_ok.	
1_C00274_00009601 Q1_L2 advertises its capability: Duplex, 10G, and XGMII.	1_C00274_00009601 Q3_L2 advertises its capability: Duplex, 10G, and XGMII.
1_C00264_01000000 Q1_L2 sets to enable interrupt upon link status update (i.e., auto nego completes).	1_C00264_010000 Q3_L2 sets to enable interrupt upon link status update (i.e., auto nego completes).
1_C00200_80091003 Q1_L2 sets signal_ok, enables Auto-Negotiation, sets the usx_speed to 10G as initial mac_speed.	-
1_C00260_03000000 Q1_L2 writes 'b1 to clear the interrupt status registers, namely status_new_link and link_sts_upd registers.	-
Auto-Negotiation starts. During Auto-Negotiation, <ol style="list-style-type: none"> <li>At this point, Q1_L2_BLOCK_LOCK and Q3_L2_BLOCK_LOCK are deasserted. Any data transfer will be lost.</li> <li>Q1_L2_IRQ and Q3_L2_IRQ are deasserted, meaning the internal an_complete in both devices are also deasserted.</li> <li>Q1_L2's AN_Advertise register (usxgmii_an_adv_register) contents transfer to the Q3_L2's AN_Link_Partner register.</li> <li>Q3_L2's AN_Link_Partner register contents transfer to the Q1_L2's AN_Link_Partner register (usxgmii_an_lp_register).</li> </ol>	
wait(Q1_L2_BLOCK_LOCK && Q1_L2_IRQ) Q1_L2 waits for an_complete (denoted by Q1_L2_IRQ).	-
Q1_L2_BLOCK_LOCK asserts, Q1_L2_IRQ asserts.	-
0_C00208_00000003 Upon the assertion of Q1_L2_IRQ, Q1_L2 reads its status_register to check for faults or errors.	-
0_C00278_0000D601 Upon the assertion of Q1_L2_IRQ, Q1_L2 reads the usxgmii_an_lp_register to discern Q3_L2's capability.	-
0_C00200_80091003 For comparison, Q1_L2 reads its own speed from control_register.	-
Q1_L2's an_lp register[11:9] = 'b011 means that Q3_L2's speed is 10G. Q1_L2's control_register[16:14] = 'b100 means that the Q1_L2's speed 10G. At this point, Q1_L2's speed is the same as Q3_L2's speed, which is 10G.	

Active_Device (Q1_L2)	Link_Partner (Q3_L2)
DATA packet transfer starts after establishing the same speed between Q1_L2 Active Device and Q3_L2 Link_Partner.	

**Table 8: Changing to a New Data Rate through Auto-Negotiation during Data Transfer** on page 24 illustrates an example of an active device (Q1\_L2) changing to a new data rate through Auto-Negotiation handshake.

**Table 8: Changing to a New Data Rate through Auto-Negotiation during Data Transfer**

Active_Device (Q1_L2)	Link_Partner (Q3_L2)
During data packet transfer, Link Partner Q3_L2 decides to change <code>mac_speed</code> from 10G to 5G. Hence, the Link Partner Q3_L2 advertises a new capability.	
-	1_C00274_00009A01 Q3_L2 advertises new capability: Duplex, 5G, and XGMII.
-	1_C00200_C000D003 Q3_L2 enables and restarts Auto-Negotiation, sets the <code>usx_speed</code> to 5G.
-	1_C00200_8000D003 <code>usx_an_restart</code> is rising edge detect, hence Q3_L2 sets <code>usx_an_restart</code> to 0 after writing.
-	1_C00260_03000000 At the start of Q3_L2's Auto-Negotiation, Q3_L2 writes 'b1 to clear the interrupt status registers, namely <code>status_new_link</code> and <code>link_sts_upd</code> registers.
wait(~Q3_L2_IRQ)	
Auto-Negotiation restarts and IRQ deasserts.	
<ol style="list-style-type: none"> <li>In Link Partner Q3_L2, although <code>Q3_L2_BLOCK_LOCK</code> remains asserted, the DATA packet transfer will be lost to give way to Auto-Negotiation.</li> <li>Once the rising-edge of <code>usx_an_restart</code> in Link Partner Q3_L2 is successfully detected, Auto-Negotiation starts, causing Link Partner Q3_L2's internal <code>an_complete</code> and <code>Q3_L2_IRQ</code> to deassert.</li> <li><code>Q3_L2_IRQ</code> deasserts, indicating that the Auto-Negotiation has started in Link Partner Q3_L2.</li> </ol>	
1_C00260_03000000 At the deassertion of <code>Q3_L2_IRQ</code> , Q1_L2 writes 'b1 to clear the interrupt status registers, namely <code>status_new_link</code> and <code>link_sts_upd</code> registers, causing <code>Q1_L2_IRQ</code> to deassert.	0_C00260_00000000
<code>Q1_L2_IRQ</code> deasserts	-
Auto-Negotiation starts. During Auto-Negotiation,	
<ol style="list-style-type: none"> <li>Auto-Negotiation in Link Partner Q3_L2 causes the deassertion of Active Device Q1_L2's internal <code>an_complete</code> (despite Active Device Q1_L2 has not triggered <code>Q1_L2_usx_an_restart</code>).</li> <li><code>Q1_L2's AN_Advertise</code> register (<code>usxgmii_an_adv_register</code>) contents transfer to the Q3_L2's <code>AN_Link_Partner</code> register.</li> <li><code>Q3_L2's AN_Link_Partner</code> register contents transfer to the Q1_L2's <code>AN_Link_Partner</code> register (<code>usxgmii_an_lp_register</code>).</li> </ol>	
wait(Q1_L2_IRQ) <code>Q1_L2</code> waits for internal <code>an_complete</code> (denoted by <code>Q1_L2_IRQ</code> ), then reads its <code>usxgmii_an_lp</code> register.	wait(Q3_L2_IRQ) <code>Q3_L2</code> waits for internal <code>an_complete</code> (denoted by <code>Q3_L2_IRQ</code> ), then reads its <code>usxgmii_an_lp</code> register.
<code>Q1_L2_IRQ</code> asserts	<code>Q3_L2_IRQ</code> asserts

Active_Device (Q1_L2)	Link_Partner (Q3_L2)
<p>0_C00278_0000DA01</p> <p>Upon the assertion of Q1_L2_IRQ, Q1_L2 reads the usxgmii_an_lp_register to discern Q3_L2's capabilities.</p> <p>Q1_L2's usxgmii_an_lp_register[11:9] = 'b101 signals Q3_L2's speed is 5G.</p>	<p>0_C00278_0000DA01</p> <p>Upon assertion of Q3_L2_IRQ, Q3_L2 reads the usxgmii_an_lp_register to discern the Q1_L2 Active Device's capabilities.</p> <p>Q3_L2's usxgmii_an_lp_register[11:9] = 'b101 signals Q1_L2 has acknowledged the Q3_L2's new 5G speed.</p>
<p>0_C00200_80091003</p> <p>Q1_L2's control_register[16:14] = 'b100 signals that the Q1_L2's speed is 10G.</p>	-
<p>1_C00274_0000DA01</p> <p>At this point, Q1_L2 decides to update its speed to match that of Q3_L2.</p> <p>Q1_L2 advertises its new capabilities: Duplex, 5G, and XGMII.</p>	-
<p>1_C00200_C008D003</p> <p>Q1_L2 sets the usx_speed to 5G, enables and restarts Auto-Negotiation.</p>	-
<p>1_C00200_8008D003</p> <p>usx_an_restart is rising edge detect, hence Q1_L2 sets usx_an_restart to 0 after writing.</p>	-
<p>1_C00260_03000000</p> <p>At the start of Active Device Q1_L2's Auto-Negotiation, Q1_L2 writes 'b1 to clear the interrupt status registers, namely status_new_link and link_sts_upd registers.</p>	-
wait(~Q1_L2_IRQ)	
<ol style="list-style-type: none"> <li>1. In Active Device, although Q1_L2_BLOCK_LOCK remains asserted, DATA packet transfer will be lost to give ways to Auto-Negotiation.</li> <li>2. Once the rising-edge of usx_an_restart in Active Device Q1_L2 is successfully detected, Auto-Negotiation starts, causing Active Device Q1_L2's internal an_complete and Q1_L2_IRQ to deassert.</li> <li>3. Q1_L2_IRQ deasserts, indicating that the Auto-Negotiation has started in Active Device Q1_L2.</li> </ol>	
<p>Auto-Negotiation starts. During Auto-Negotiation,</p> <ol style="list-style-type: none"> <li>1. Auto-Negotiation in Active Device Q1_L2 causes the deassertion of Link Partner Q3_L2's internal an_complete (despite Active Device Q3_L2 has not triggered Q3_L2_usx_an_restart).</li> <li>2. Q1_L2's AN_Advertise register (usxgmii_an_adv_register) contents transfer to the Q3_L2's AN_Link_Partner register.</li> <li>3. Q3_L2's AN_Link_Partner register contents transfer to the Q1_L2's AN_Link_Partner register (usxgmii_an_lp_register).</li> </ol>	
<p>0_C00260_00000000</p> <p>Q1_L2's interrupt status registers, which are status_new_link and link_sts_upd registers, are cleared.</p>	<p>1_C00260_03000000</p> <p>At the deassertion of Q1_L2_IRQ (at the start of Active Device Q1_L2's Auto-Negotiation), Q3_L2 writes 1 to clear the interrupt status registers, namely status_new_link and link_sts_upd registers, causing Q3_L2_IRQ to deassert.</p>
<p>wait(Q1_L2_IRQ)</p> <p>Q1_L2 waits for internal an_complete (denoted by Q1_L2_IRQ), reads its an_lp register.</p>	<p>wait(Q3_L2_IRQ)</p> <p>Q3_L2 waits for internal an_complete (denoted by Q3_L2_IRQ), reads its usxgmii_an_lp register.</p>
<p>Q1_L2_IRQ asserts</p>	<p>Q3_L2_IRQ asserts</p>

Active_Device (Q1_L2)	Link_Partner (Q3_L2)
<p>0_C00208_00000003</p> <p>Upon the assertion of Q1_L2_IRQ, Q1_L2 reads its status_register to check for faults or errors.</p>	<p>0_C00278_0000DA01</p> <p>Upon assertion of Q3_L2_IRQ, Q3_L2 reads the usxgmii_an_lp_register to discern the Active Device's capabilities.</p> <p>Q3_L2's usxgmii_an_lp_register[11:9] = 'b101 means Q1_L2's speed is 5G</p>
<p>0_C00278_0000DA01</p> <p>Upon the assertion of Q1_L2_IRQ, Q1_L2 reads the usxgmii_an_lp_register to discern Q3_L2's capabilities.</p>	<p>-</p>
<p>0_C00200_8008D003</p> <p>For comparison, Q1_L2 reads its own speed from control_register.</p>	<p>-</p>
<p>Q1_L2's an_lp_register[11:9] = 'b101 means that Q3_L2's speed is 5G.</p> <p>Q1_L2's control_register[16:14] = 'b011 means that the Q1_L2's speed 5G.</p> <p>At this point, Q1_L2's speed is the same as that of Q3_L2, which is 5G.</p>	
<p>DATA packet starts transfer after establishing the same speed between Q1_L2 and Link_Partner (i.e., Q3_L2)</p>	

## Clock Divider and Switching

The clock divider and switching module controls the data rate in the `Lx_MAC_CLK` domain, and also the `Lx_RATE_READY` signal in the `Lx_MAC_RST_N` domain. The data rate setting comes from the `usx_speed` register, which is configured through the APB module.

When the data rate switches, the `Lx_RATE_READY` signal deasserts to 0, indicating that the `Lx_MAC_CLK` is undergoing a speed change operation. During the deassertion of the `Lx_RATE_READY` signal, the data transfer becomes unpredictable. Hence, when `Lx_RATE_READY = 0`, only IDLE packets are transferred.

When the new data rate becomes effective and stabilizes, the `Lx_RATE_READY` signal asserts to 1, marking the completion of the speed change operation, and `Lx_MAC_CLK` is ready for use. The data packet transfer is resumed when the `Lx_RATE_READY` signal asserts to 1.

## Data Transfer

TX replication and RX de-replication modules transfer data between the Ethernet MAC and FPGA Ethernet USXGMII PCS. The input and output of these 2 modules operate in XGMII format, which comprises 64-bit data and 8-bit control. As both interfaces operate in XGMII format, a prefix is added to the port names to distinguish between the port interfaces. On the user's MAC interface, the ports are prefixed with "mac\_", whereas at the PCS interface, the ports have the prefix "XGMII\_".

At the TX channel, the data from the Ethernet MAC undergoes an internal TX FIFO to cross from `Lx_MAC_CLK` domain to the `Lx_PCS_CLK` domain. Then, it is replicated for the FPGA Ethernet USXGMII PCS.

At the RX channel, the data from the FPGA Ethernet USXGMII PCS is de-replicated and undergoes an internal RX FIFO, and then channeled to the Ethernet MAC. The internal RX FIFO has 2 major functions:

- Synchronizes the RX data path from `Lx_PCS_CLK` domain to `Lx_MAC_CLK` domain.
- Cushions any irregularities in the de-replication due to IDLE deletion by the clock tolerance compensation (CTC) block in the FPGA Ethernet USXGMII PCS.

The maximum Ethernet (jumbo) frame size supported is as follows:

**Table 9: Maximum Frame Size for Each Data Rate**

Mac Speed	Maximum Jumbo Frame size
10 G	16 kByte
5 G	8 kByte
2.5	4 kByte
1 G	16 kByte
100 M	16 kByte

The `lx_rx_fifo_full` status port indicates the status of the internal RX FIFO. If `lx_rx_fifo_full` asserts, it signals that there is a potential data loss due to an overflow in the RX FIFO. Hence, you need to monitor the `lx_rx_fifo` port. If `lx_rx_fifo_full` asserts, you should deploy priority flow control at the system level to stall the incoming RX stream from the FPGA Ethernet USXGMII PCS.

# Latency

Depending on the data rate, the latency in the TX path ranges from 7 to 14 clock cycles of the `Lx_PCS_CLK` signal, while the latency in the RX path ranges from 7 to 24 clock cycles.

To resolve any timing issue, you may set **Synthesize TX / RX Pipeline Registers in Lane X** to **Yes** to improve the timing between the FPGA Ethernet USXGMII PCS and the USXGMII Multirate Ethernet core. When you set **Synthesize TX / RX Pipeline Registers in Lane X** to **Yes**, it incurs an additional clock cycle of `Lx_PCS_CLK` in the TX and/or RX path.

# Efinity: IP Catalog, Interface Designer, and Integration

The USXGMII Multirate Ethernet core is designed to interact with the user's Ethernet MAC and the FPGA Ethernet USXGMII PCS, as shown in [Figure 1: USXGMII Multirate Ethernet Interacts with Ethernet MAC and FPGA Ethernet USXGMII PCS](#) on page 3. To combine the USXGMII Multirate Ethernet core, FPGA Ethernet USXGMII PCS, and the user's Ethernet MAC together as a whole design, you need to perform the following steps:

1. Configure and generate the USXGMII Multirate Ethernet core from the IP Catalog.
2. Configure and generate the Ethernet 10G MAC core from the IP Catalog (optional).
3. Configure and generate the FPGA Ethernet USXGMII PCS using the Interface Designer.
4. Create a top-level module to integrate the USXGMII Multirate Ethernet core (from step 1), the Ethernet 10G MAC core (from step 2), the FPGA Ethernet USXGMII PCS (from step 3), and the user's logic.

## IP Catalog

The USXGMII Multirate Ethernet core is ready to use from IP Catalog. For steps to customize an IP core with the IP Configuration wizard, refer to [Generating the USXGMII Multirate Ethernet Core with the IP Manager](#).

## Interface Designer

Refer to *Chapter 16 10Gbase-KR Interface* of [Titanium Interfaces User Guide](#).

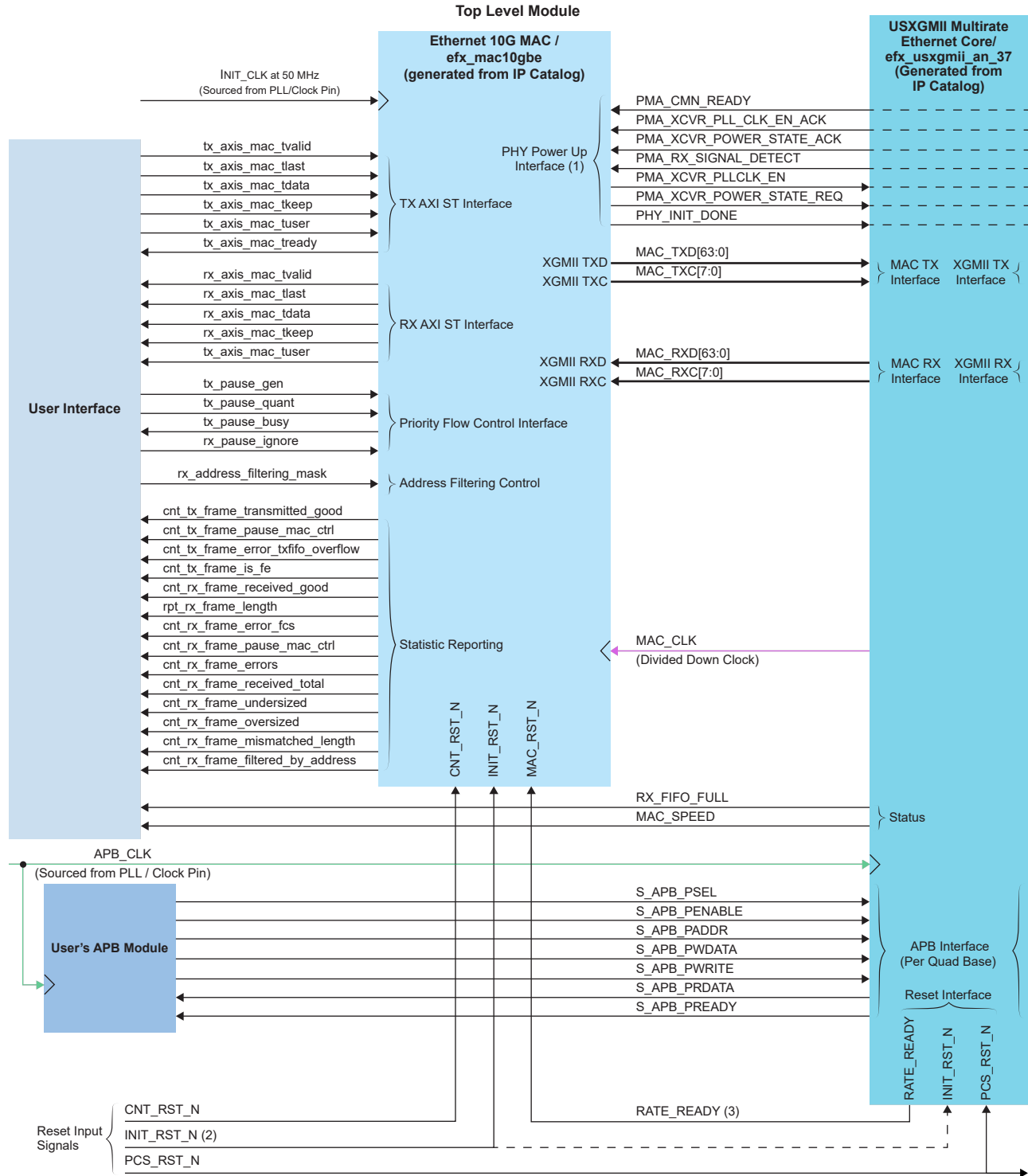
## Top Level Module

You need to create a top-level module to integrate the USXGMII Multirate Ethernet core, Ethernet 10G MAC core, and the FPGA Ethernet USXGMII PCS. Refer to [Figure 9: Top-Level Module for Efinity Compilation \(Part 1\)](#) on page 30, which illustrates the integration of one lane of the USXGMII Multirate Ethernet core, Ethernet 10G MAC core, and the FPGA Ethernet USXGMII PCS for Efinity compilation.



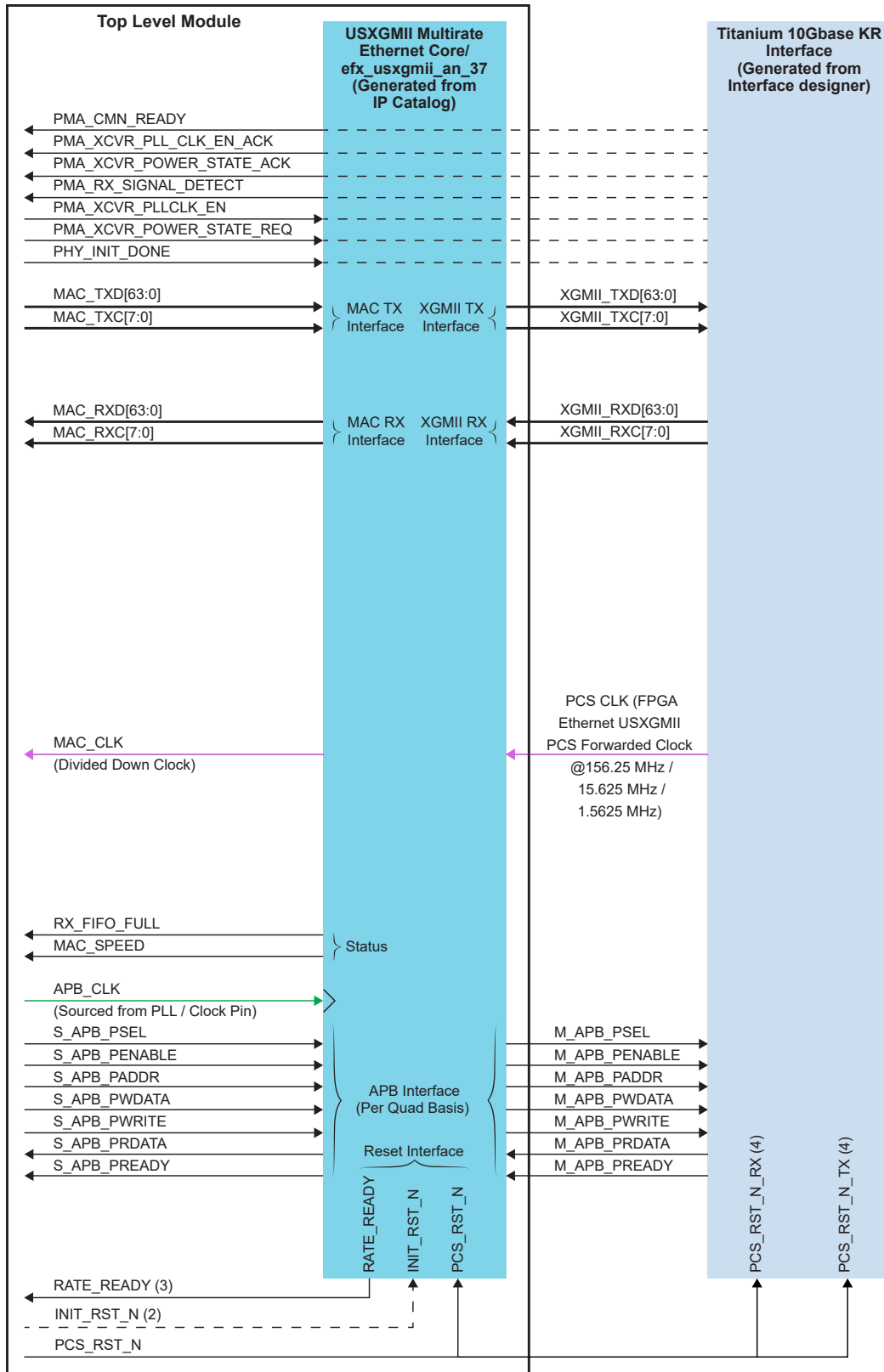
**Note:** In the Efinity software and [Figure 9: Top-Level Module for Efinity Compilation \(Part 1\)](#) on page 30, the FPGA Ethernet USXGMII PCS is represented by the Titanium 10Gbase KR Interface. The Titanium 10Gbase KR Interface is configured and generated from the Interface Designer.

Figure 9: Top-Level Module for Efinix Compilation (Part 1)



- Notes:**
- 1) Power-up initialization sequence module is available in the Ethernet 10G MAC core. Hence, in the USXGMII Multirate Ethernet core, you should set **Synthesize Power Up Module in Lane X** to **No** to prevent duplication. Alternatively, you can tie Ethernet 10G MAC core `INIT_RST_N` to 0 to disable the power-up initialization sequence module in the Ethernet 10G MAC core, and set **Synthesize Power Up Module in Lane X** to **Yes**.
  - 2) You should drive the `INIT_RST_N` of the USXGMII Multirate Ethernet core if you set **Synthesize Power Up Module in Lane X** to **Yes**.
  - 3) Efinix recommends to connect the output `RATE_READY` from the USXGMII Multirate Ethernet Core to the input `MAC_RST_N` of the Ethernet 10G MAC core.

Figure 10: Top-Level Module for Efinity Compilation (Part 2)



- Notes:**
- 2) You should drive the `INIT_RST_N` of the USXGMII Multirate Ethernet core if you set **Synthesize Power Up Module in Lane X to Yes**.
  - 3) Efinix recommends to connect the output `RATE_READY` from the USXGMII Multirate Ethernet Core to the input `MAC_RST_N` of the Ethernet 10G MAC core.
  - 4) The 3 PCS reset signals, the USXGMII Multirate Ethernet core's `PCS_RST_N` and the FPGA Ethernet USXGMII PCS's `PCS_RST_N_TX` and `PCS_RST_N_RX`, may be driven from the same source or be driven independently.

## IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate Efinity® IP cores. The IP Manager performs validation checks on the parameters you set to ensure that your selections are valid. When you generate the IP core, you can optionally generate an example design targeting an Efinity development board and/or a testbench. This wizard is helpful in situations in which you use several IP cores, multiple instances of an IP core with different parameters, or the same IP core for different projects.



**Note:** Not all Efinity IP cores include an example design or a testbench.

### Generating the USXGMII Multirate Ethernet Core with the IP Manager

The following steps explain how to customize an IP core with the IP Configuration wizard.

1. Open the IP Catalog.
2. Choose **Ethernet > USXGMII Multirate Ethernet** core and click **Next**. The **IP Configuration** wizard opens.
3. Enter the module name in the **Module Name** box.



**Note:** You cannot generate the core without a module name.

4. Customize the IP core using the options shown in the wizard. For detailed information on the options, refer to the [Customizing the USXGMII Multirate Ethernet](#) on page 34 section.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an Efinity® development board and/or testbench. These options are turned on by default.
6. (Optional) In the **Summary** tab, review your selections.
7. Click **Generate** to generate the IP core and other selected deliverables.
8. In the **Review configuration generation** dialog box, click **Generate**. The Console in the **Summary** tab shows the generation status.



**Note:** You can disable the **Review configuration generation** dialog box by turning off the **Show Confirmation Box** option in the wizard.

9. When generation finishes, the wizard displays the **Generation Success** dialog box. Click **OK** to close the wizard.

The wizard adds the IP to your project and displays it under **IP** in the Project pane.

## Generated Files

The IP Manager generates these files and directories:

- **<module name>\_define.svh**—Contains the customized parameters.
- **<module name>\_tpl.sv**—Verilog HDL instantiation template.
- **<module name>\_tpl.vhd**—VHDL instantiation template.
- **<module name>.sv**—IP source code.
- **settings.json**—Configuration file.
- **efx\_usxgmii\_an\_37\_exp**—Has generated RTL, example design, and Efinity® project targeting a specific development board.
- **Testbench**—Contains simulation models. Testbench is not available.

## Timing Constraints: SDC

The USXGMII Multirate Ethernet core comprises clock division module, TX FIFO and RX FIFO, which require correct timing constraints to ensure timing closure during static timing analysis (STA). Refer to the SDC in the example design for the “create\_generated\_clock” for clock division modules, and the “set\_false\_path” for the FIFOs.

There are 4 lanes in the USXGMII Multirate Ethernet core. To correctly constrain the instance in the SDC, refer to the post synthesis netlist in the <user design>.map.v in the outflow directory. **Table 10: Instance Name for SDC** on page 33 provides a guideline of the instances name in the USXGMII Multirate Ethernet core.

**Table 10: Instance Name for SDC**

Lane Number	Instance Name
Lane 0	<USXGMII Multirate Ethernet Core instance name>/u_efx_usxgmii_an_37/ <b>genblk4.inst_I0</b>
Lane 1	<USXGMII Multirate Ethernet Core instance name>/u_efx_usxgmii_an_37/ <b>genblk3.inst_I1</b>
Lane 2	<USXGMII Multirate Ethernet Core instance name>/u_efx_usxgmii_an_37/ <b>genblk2.inst_I2</b>
Lane 3	<USXGMII Multirate Ethernet Core instance name>/u_efx_usxgmii_an_37/ <b>genblk1.inst_I3</b>



**Note:**

- To prevent unintentional or accidental constraints, set\_false\_path constraints should only be used if necessary.
- Wildcards should be used cautiously.

# Customizing the USXGMII Multirate Ethernet

The core has parameters, so that you can customize its function. You set the parameters in any/all of the **Lane 3**, **Lane 2**, **Lane 1**, and **Lane 0** tabs of the core's IP Configuration window.

**Table 11: USXGMII Multirate Ethernet Core Parameters**

Name	Options	Description
Synthesize Lane X	Yes No	<p>Default: Yes</p> <p>To enable or disable the synthesis of the USXGMII Multirate Ethernet core in Lane X.</p> <hr/> <p> <b>Important:</b> You must configure the correct lane. The lane setting defines the APB address and configuration during the rate change and auto-negotiation.</p> <hr/> <p>By setting the configuration to <b>No</b> for Lane X, there is no synthesis for the USXGMII Multirate Ethernet core IP in Lane X. Hence, all corresponding configurations related to Lane X are set to disable mode.</p>
Synthesize Power Up Module in Lane X	Yes No	<p>Default: Yes</p> <p>To enable or disable the synthesis of the power-up module in Lane X.</p> <p>By setting the configuration to <b>No</b>, you must create your own module to perform the power-up handshake with the FPGA transceiver.</p>
Synthesize TX Pipeline Register in Lane X	Yes No	<p>Default: Yes</p> <p>To enable or disable the synthesis of the TX pipeline registers in Lane X to assist STA closure.</p> <p>You may set the configuration to <b>No</b> if the timing closure is good for the PCS_CLK and MAC_CLK in the respective lanes.</p>
Set TX FIFO Type in Lane X	RAM Registers	<p>Default: RAM.</p> <p>By selecting the default, this allows the synthesis of the internal TX FIFO with memory blocks.</p> <p>By setting the configuration to <b>Registers</b>, Efinity synthesizes the internal TX FIFO with registers.</p>
Synthesize RX Pipeline Register in Lane X	Yes No	<p>Default: Yes</p> <p>To enable or disable the synthesis of the RX pipeline registers in Lane X to assist STA closure.</p> <p>You may set the configuration to <b>No</b> if the timing closure is good for the PCS_CLK and MAC_CLK in the respective lanes.</p>
Set RX FIFO Type in Lane X	RAM Registers	<p>Default: RAM.</p> <p>By selecting the default, this allows the synthesis of the internal RX FIFO with memory blocks.</p> <p>By setting the configuration to <b>Registers</b>, Efinity synthesizes the internal RX FIFO with registers.</p>

# Ports

**Table 12: USXGMII Multirate Ethernet Core Ports Interface**

Port Name	Direction	Bus Width	Clock Domain	Description
<b>Clock and Reset<sup>(2)</sup></b>				
apb_clk	Input	1	-	Freq = 50 to 200 MHz
apb_rst_n	Input	1	Async	Reset the APB module in the USXGMII Multirate Ethernet core, which affects the mac_speed register.
Lx_pcs_clk	Input	1	-	Freq = 156.25 MHz, 15.625 MHz, and 1.5625 MHz
Lx_pcs_rst_n	Input	1	Async	Reset majority functions and clock module in the USXGMII Multirate Ethernet native.
Lx_mac_clk	Output	1	-	Freq = 156.25 MHz for 10G, Freq = 78.125 MHz for 5G, Freq = 39.0625 MHz for 2.5G, Freq = 15.625 MHz for 1G, Freq = 1.5625 MHz for 100M
Lx_rate_ready	Output	1	Lx_mac_clk	The assertion of Lx_rate_ready indicates that the clock switching has completed and the Lx_mac_clk is stable. Reset sync to Lx_mac_clk domain.
Lx_init_clk <sup>2</sup>	Input	1	-	Freq = 50 MHz or below
Lx_init_rst_n <sup>2</sup>	Input	1	Async	Reset the power-up module between the soft IP and the FPGA PHY.
<b>APB Interface</b>				
s_apb_psel	Input	1	apb_clk	Connects to the user interface. Refer to <b>APB Configuration</b> on page 18.
s_apb_penable	Input	1	apb_clk	
s_apb_pwrite	Input	1	apb_clk	
s_apb_paddr	Input	24	apb_clk	
s_apb_pwdata	Input	32	apb_clk	
s_apb_prdata	Output	32	apb_clk	
s_apb_pready	Output	1	apb_clk	
s_apb_slvrr	Output	1	apb_clk	
m_apb_psel	Output	1	apb_clk	Connects to Interface Designer. Refer to <b>APB Configuration</b> on page 18.
m_apb_penable	Output	1	apb_clk	
m_apb_pwrite	Output	1	apb_clk	
m_apb_paddr	Output	24	apb_clk	
m_apb_pwdata	Output	32	apb_clk	
	Output			

<sup>(2)</sup> For details on Clock and Reset, refer to **Clock Sources** on page 9 and **Reset Signals** on page 12.

Port Name	Direction	Bus Width	Clock Domain	Description
m_apb_prdata	Input	32	apb_clk	
m_apb_pready	Input	1	apb_clk	
m_apb_slvrr	Input	1	apb_clk	
<b>Data Transfer: MAC_TX, MAC_RX</b>				
Lx_mac_txd	Input	64	Lx_mac_clk	TX Data from the user's Ethernet MAC.
Lx_mac_txc	Input	8	Lx_mac_clk	TX Control from the user's Ethernet MAC.
Lx_mac_rxd	Output	64	Lx_mac_clk	RX Data to the user's Ethernet MAC.
Lx_mac_rxc	Output	8	Lx_mac_clk	RX Control to the user's Ethernet MAC.
<b>XGMII Interface: Connect with Interface Designer</b>				
Lx_XGMII_TXD	Output	64	Lx_pcs_clk	XGMII TX Data.
Lx_XGMII_TXC	Output	8	Lx_pcs_clk	XGMII TX Control.
Lx_XGMII_RXD	Input	64	Lx_pcs_clk	XGMII RX Data.
Lx_XGMII_RXC	Input	8	Lx_pcs_clk	XGMII RX Control.
<b>Status</b>				
lx_mac_speed	output	3	apb_clk	MAC Speed in Lane X.
lx_rx_fifo_full	output	1	Lx_pcs_clk	Indicator of the internal RX FIFO becoming full due to irregularities during the de-replication.
<b>Power Up Initialization Sequence<sup>(3)</sup></b>				
Lx_PMA_CMN_READY	Input	1	Async, Pseudo-Static	Output from PHY to indicate the readiness of PHY calibration.
Lx_PMA_XCVR_PLLCLK_EN	Output	1	Lx_init_clk	Refer to <b>Power Up Handshake with FPGA Transceiver</b> on page 16.
Lx_PMA_XCVR_PLLCLK_EN_ACK	Input	1	Async, Pseudo-Static	
Lx_PMA_XCVR_POWER_STATE_REQ	Output	4	Lx_init_clk	
Lx_PMA_XCVR_POWER_STATE_ACK	Input	4	Async, Pseudo-Static	
Lx_PMA_RX_SIGNAL_DETECT	Input	1	Async	
Lx_phy_init_done	Output	1	Async, Pseudo-Static	



**Note:** 'Lx' denotes Lane 0/1/2/3.

<sup>(3)</sup> The synthesis of the power-up initialization sequence module is optional and controlled by the user-defined **Synthesize Power Up Module in Lane X**. When you select **Synthesize Power Up Module in Lane X = No**, these ports become unavailable.

# USXGMII Multirate Ethernet Example Design

You can choose to generate the example design when generating the core in the IP Manager Configuration window. Compile the example design project and download the **.hex** or **.bit** file to your board. To generate example design, the **Example Design Deliverables Option** signal must be enabled.

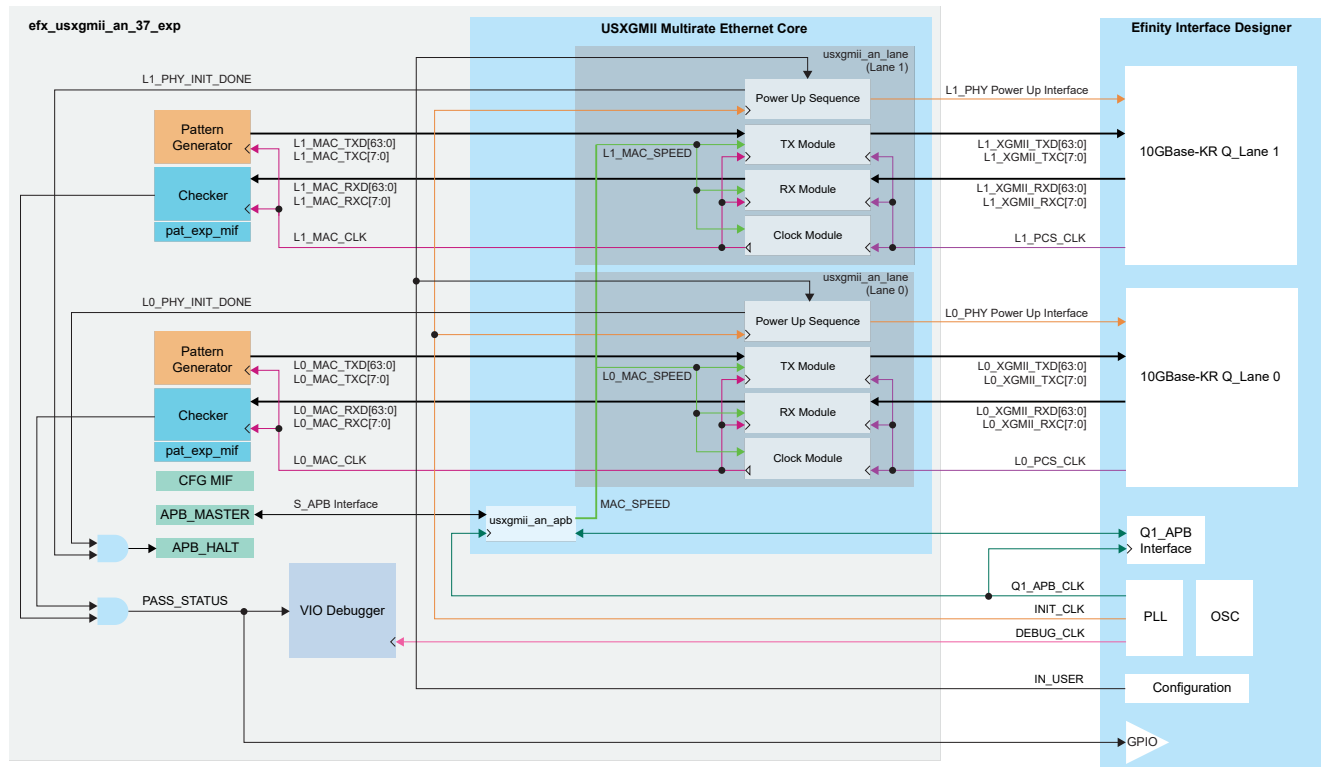


**Important:**

- The USXGMII Multirate Ethernet example design instantiates Q1 Lane 0 and Q1 Lane 1. Hence, you need to configure the USXGMII Multirate Ethernet core with the following:
  - Set **Synthesize Lane 0** to **Yes**.
  - Set **Synthesize Lane 1** to **Yes**.
  - Set **Synthesize Lane 2** to **No**.
  - Set **Synthesize Lane 3** to **No**.
- Efinix tested the example design generated with the default parameter options only.

The example design demonstrates the connectivity of the USXGMII Multirate Ethernet core, FPGA Ethernet USXGMII PCS, and the user's logic, as shown in **Figure 11: USXGMII Multirate Ethernet Core Example Design (efx\_usxgmii\_an\_37\_exp.sv)** on page 37.

Figure 11: USXGMII Multirate Ethernet Core Example Design (efx\_usxgmii\_an\_37\_exp.sv)



In the USXGMII Multirate Ethernet core example design, the user's logic is represented by the following modules:

- *Pattern Generators*—Denoted by `efx_usxgmii_an_exp_pat_gen.sv`.
- *Checkers*—Denoted by `efx_usxgmii_an_exp_checker.sv` and `efx_usxgmii_an_exp_pat_exp.mif`.
- *APB Modules*—Denoted by `efx_usxgmii_an_exp_apb_halt.sv`, `efx_usxgmii_an_exp_apb_master.sv`, and `efx_usxgmii_an_exp_cfg.mif`.
- *VIO Debugger*—Denoted by `efx_debug_edb_top.sv`.

The example design further demonstrates the quintessential APB commands and handshakes to perform the following functions:

- Enable the RX path during the CDR-locked-to-data stage, as described in [Figure 6: Power Up Handshake with the FPGA Transceiver PHY and PCS Per Lane](#) on page 17.
- Auto-negotiation to change the MAC speed from 10G → 5G → 2.5G → 1G, as described in [Table 8: Changing to a New Data Rate through Auto-Negotiation during Data Transfer](#) on page 24.



**Note:** The APB configuration is on a per quad basis. [Table 8: Changing to a New Data Rate through Auto-Negotiation during Data Transfer](#) on page 24 describes the auto-negotiation mechanism between 2 lanes from different quads, unlike the example design, which demonstrates the same mechanism between 2 lanes from the same quad.

In the example design, the APB configuration controller (`efx_usxgmii_an_exp_apb_halt`) controls the sequence of APB programming based on events such as `IRQ` and `BLOCK_LOCK`. The APB master controller (`efx_usxgmii_an_exp_apb_master`) has a built-in ROM and RAM. The built-in ROM stores the configuration settings (denoted by `efx_usxgmii_an_exp_cfg.mif`) using the format as described in [Figure 7: Format of APB Configuration](#) on page 19. Concurrently, the built-in RAM captures the `APB_PRDATA` and `APB_PADDRESS` of all the APB Read commands. By using the VIO Debugging to input the value to `ram_usr_addr_i`, you may access the built-in RAM to read out the `APB_PRDATA` and the corresponding `APB_PADDRESS` from the `ram_dout_d` and `ram_dout_a` ports.



**Note:**

- The built-in RAM in the APB master controller stores the `APB_PRDATA` and the corresponding `APB_ADDR` of all the APB Read requests, in ascending order.
- The results of the 1<sup>st</sup> APB Read request are stored in the Address 00, the 2<sup>nd</sup> request is stored in the Address 01, the 3<sup>rd</sup> request is stored in the Address 02, and so forth.

Upon entering the user mode, the `IN_USER` signal releases the `LX_INIT_RST_N` to kick start the PHY power-up sequence handshake, resulting in the assertion of `PHY_INIT_DONE` in Lane 0 and Lane 1. At the same time, the assertion of `PLL_LOCK` releases the resets in the USXGMII Multirate Ethernet core and the FPGA Ethernet USXGMII PCS.



**Note:** In the example design, the 3 PCS reset signals are driven by the `PLL_LOCK` signal.

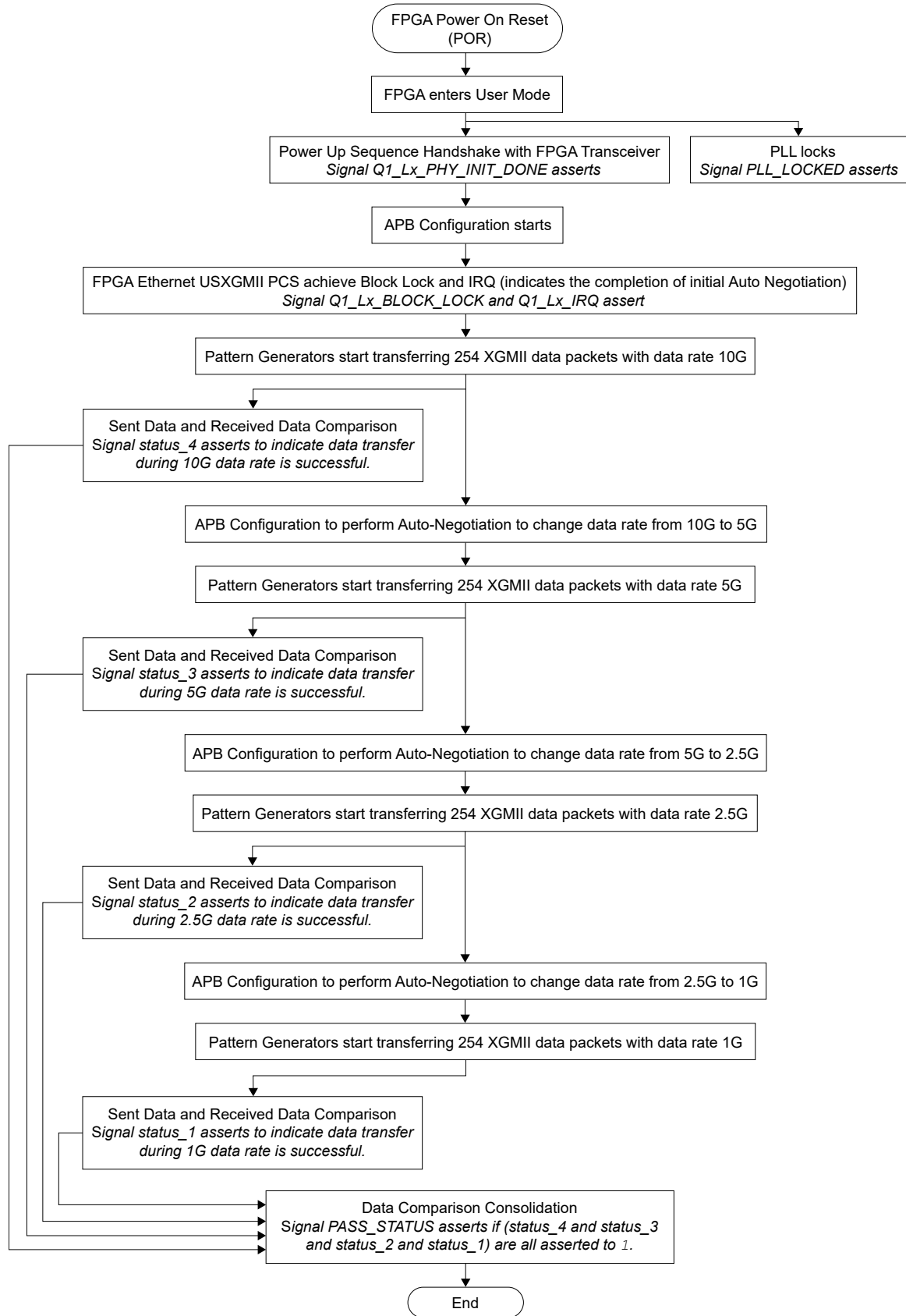
Upon the assertion of `PHY_INIT_DONE`, the APB modules become operational and start the APB configuration based on the built-in ROM content. There are several phases of APB configurations. The 1<sup>st</sup> phase of APB configuration involve the configuration of the 'signal\_ok'. After the 1<sup>st</sup> phase of APB configuration, the RX path in the FPGA becomes functional and achieves BLOCK LOCK status. Upon achieving BLOCK LOCK status, the pattern generator starts transmitting data frames into the {`LX_MAC_TXC`, `LX_MAC_TXD`} interface of the USXGMII Multirate Ethernet core.

The pattern generator halts the data transfer after transmitting 254 packets to perform Auto-Negotiation, eventually changing the MAC speed. The APB modules (`efx_usxgmii_an_exp_apb_halt` and `efx_usxgmii_an_exp_apb_master`) configure through the APB interface to perform the Auto-Negotiation.

For the entire operation, the checker actively matches the {`LX_MAC_RXC`, `LX_MAC_RXD`} of the USXGMII Multirate Ethernet core against the expected RX streams (denoted by `efx_usxgmii_an_pat_exp.mif`). Finally, the `PASS_STATUS` is asserted if the {`LX_MAC_RXC`, `LX_MAC_RXD`} matches the `efx_usxgmii_an_pat_exp.mif`.

**Figure 12: USXGMII Multirate Ethernet Core Example Design Flow** on page 40 describes the flow of the example design.

*Figure 12: USXGMII Multirate Ethernet Core Example Design Flow*



## Generating the Example Design

Use the following steps to generate the example design from the IP Catalog.

1. Create a new project.
2. Select the **USXGMII Multirate Ethernet** core from the IP Catalog and set your configurations.
3. In the **Deliverables** tab, turn on the **Example Design (efx\_usxgmii\_an\_37\_exp)**.
4. Click **Generate**.

The example design is available in `<project path>\ip\<ip-module-name>\efx_usxgmii_an_37_exp` directory.

To load the example design into the FPGA development board:

1. Close the existing project.
2. To open the example design, select `<project path>\ip\<ip-module-name>\efx_usxgmii_an_37_exp\efx_usxgmii_an_37_exp.xml`
3. Compile the design.
4. Download the design to your Titanium Ti375 N1156 Development Board.



**Note:** In the following tables, the signal names are prefixed with Q1\_, Q1\_L0, Q1\_L1 to indicate and distinguish instances of modules in the example design. For detailed signal description, refer to [Titanium Ethernet 10GBase-KR User Guide, Ports](#) on page 35, and tables regarding the settings in [Virtual I/O Debugger Settings](#) on page 45.

Table 13: Example Design Input and Outputs

Signal Name	Direction	Width	Description	
INIT_CLK	input	1	Connect from PLL_TR1 in FPGA peripheral.	
DEBUG_CLK	input	1		
Q1_APB_CLK	input	1		
PLL_LOCKED	input	1		
IN_USER	input	1	Connect from Configuration Block in the FPGA.	
PASS_STATUS	output	1	Connect to GPIO to indicate the PASS status of the transmitted and received XGMII packets.	
Q1_PMA_CMN_READY	input	1	Connect from the FPGA Transceiver Q1 Common Interface.	
Q1_L1_PMA_XCVR_PLLCLK_EN	output	1	Connect with the Power Up Interface of FPGA Transceiver of Q1 Lane 1.	
Q1_L1_PMA_XCVR_PLLCLK_EN_ACK	input	1		
Q1_L1_PMA_XCVR_POWER_STATE_REQ	output	4		
Q1_L1_PMA_XCVR_POWER_STATE_ACK	input	4		
Q1_L1_PMA_RX_SIGNAL_DETECT	input	1		
Q1_L0_PMA_XCVR_PLLCLK_EN	output	1	Connect with the Power Up Interface of FPGA Transceiver of Q1 Lane 0.	
Q1_L0_PMA_XCVR_PLLCLK_EN_ACK	input	1		
Q1_L0_PMA_XCVR_POWER_STATE_REQ	output	4		
Q1_L0_PMA_XCVR_POWER_STATE_ACK	input	4		
Q1_L0_PMA_RX_SIGNAL_DETECT	input	1		
Q1_L1_PCS_CLK	input	1	Connect with the FPGA Ethernet USXGMII PCS of Q1 Lane 1.	
Q1_L1_PCS_RST_N_RX	output	1		
Q1_L1_PCS_RST_N_TX	output	1		
Q1_L1_BLOCK_LOCK	input	1		
Q1_L1_HI_BER	input	1		
Q1_L1_IRQ	input	1		
Q1_L1_PCS_STATUS	input	1		
Q1_L1_PHY_INTERRUPT	input	1		
Q1_L1_PMA_TX_ELEC_IDLE	output	1		
Q1_L1_ETH_EEE_ALERT_EN	output	1		
Q1_L1_TXD	output	64		
Q1_L1_TXC	output	8		
Q1_L1_RXD	input	64		
Q1_L1_RXC	input	8		
Q1_L0_PCS_CLK	input	1		Connect with the FPGA Ethernet USXGMII PCS of Q1 Lane 0.
Q1_L0_PCS_RST_N_RX	output	1		
Q1_L0_PCS_RST_N_TX	output	1		
Q1_L0_BLOCK_LOCK	input	1		

Signal Name	Direction	Width	Description
Q1_L0_HI_BER	input	1	
Q1_L0_IRQ	input	1	
Q1_L0_PCS_STATUS	input	1	
Q1_L0_PHY_INTERRUPT	input	1	
Q1_L0_PMA_TX_ELEC_IDLE	output	1	
Q1_L0_ETH_EEE_ALERT_EN	output	1	
Q1_L0_TXD	output	64	
Q1_L0_TXC	output	8	
Q1_L0_RXD	input	64	
Q1_L0_RXC	input	8	
Q1_USER_APB_PSEL	output	1	Connect with Q1 APB Interface.
Q1_USER_APB_PENABLE	output	1	
Q1_USER_APB_PWRITE	output	1	
Q1_USER_APB_PADDR	output	24	
Q1_USER_APB_PWDATA	output	32	
Q1_USER_APB_PRDATA	input	1	
Q1_USER_APB_PREADY	input	1	
Q1_USER_APB_PSLVERR	input	1	
jtag_vio_CAPTURE	input	1	JTAG Interface to operate the Virtual I/O Debugger.
jtag_vio_DRCK	input	1	
jtag_vio_RESET	input	1	
jtag_vio_RUNTEST	input	1	
jtag_vio_SEL	input	1	
jtag_vio_SHIFT	input	1	
jtag_vio_TCK	input	1	
jtag_vio_TDI	input	1	
jtag_vio_TMS	input	1	
jtag_vio_UPDATE	input	1	
jtag_vio_TDO	output	1	

Table 14: Example Design Project Files

File Name	Description
<code>efx_usxgmii_an_37_exp.sv</code>	Top level wrapper of the USXGMII Multirate Ethernet core example design.
<code>&lt;user_defined_ip_name&gt;.sv<sup>(4)</sup></code>	The generated USXGMII Multirate Ethernet core file based on the user configuration in Efinity IP Manager.
<code>efx_usxgmii_an_37_exp.sdc</code>	Timing constraint file for the example design.
<code>efx_usxgmii_an_exp_cfg.mif</code>	Initialization Hex file for APB configuration.
<code>efx_usxgmii_an_exp_apb_master.sv<sup>(4)</sup></code>	APB master controller module.
<code>efx_usxgmii_an_exp_apb_halt.sv<sup>(4)</sup></code>	APB configuration controller module.
<code>efx_usxgmii_an_exp_pat_gen.sv</code>	Pattern generator module.
<code>efx_usxgmii_an_exp_checker.sv</code> <code>efx_usxgmii_an_pat_exp.mif</code>	Checker module to validate the data received.
<code>efx_asyncreg.v<sup>(4)</sup></code>	Bit synchronizer register module.
<code>efx_resetsync.v<sup>(4)</sup></code>	Reset the synchronizer module.
<code>efx_debug_edb_top.v</code>	Verilog file for EFX debug module.
<code>debug_profile.json</code>	Virtual I/O Debugger core file. Load this file in the Efinity Virtual I/O Debugger to customize the example design. See <a href="#">Virtual I/O Debugger Settings</a> on page 45.

<sup>(4)</sup> This module is encrypted.


## Virtual I/O Debugger Settings

This example design has a plug-in interface known as Virtual I/O Debugger. This interface allows you to download the bitstream to your device. After the downloads are completed, you can use the Virtual I/O Debugger interface to probe and control the signals of the example design. To control the settings, refer to the Virtual I/O Debugger settings.

**Table 15: Efinity Virtual I/O Debugger Settings - Power Up Handshake Interface and FPGA Ethernet USXGMII PCS**

Signal Name	Width	Probe / Source	Description
Q1_PMA_CMN_READY	1	Probe	Refer to the signal description in the Signals per Lane table in the <i>Signal</i> chapter of the <a href="#">Titanium Ethernet 10GBase-KR User Guide</a>
Q1_L0_PMA_RX_SIGNAL_DETECT	1	Probe	
Q1_L1_PMA_RX_SIGNAL_DETECT	1	Probe	
Q1_L0_init_done	1	Probe	
Q1_L1_init_done	1	Probe	
Q1_L0_BLOCK_LOCK	1	Probe	
Q1_L1_BLOCK_LOCK	1	Probe	
Q1_L0_IRQ	1	Probe	
Q1_L1_IRQ	1	Probe	

Table 16: Efinity Virtual I/O Debugger Settings - APB Configuration

Signal Name	Width	Probe / Source	Description
Q1_cfg_cnt	7	Probe	Configuration count, corresponds to line number in <b>efx_usxgmii_an_exp_cfg.mif</b> . For instance, if the Q1_cfg_cnt displays 5, it shows the example design has completed the 5 <sup>th</sup> APB configuration, which corresponds to the 5 <sup>th</sup> line of <b>efx_usxgmii_an_exp_cfg.mif</b> .
Q1_apb_rom_end_vio_o	1	Probe	The assertion of this signal indicates that all the configuration settings stored in the built-in APB ROM are executed. If this signal stays 0, it signals that the APB master controller is still operating (or waiting) with configurations from the built-in ROM.
Q1_apb_done_vio_o	1	Probe	Indicates that the APB request is completed and ready to accept new APB requests. This signal indicates the readiness of the APB Master to accept the manual APB request. When this signal is 0, it signals that there is an active APB command on-going and is waiting for the APB_PREADY assertion. You can only trigger the manual APB request when this signal is 1.   <b>Note:</b> This signal is 0 when Q1_apb_rom_end is 0, signalling that the APB master controller is still operating (or waiting) with configurations from the built-in ROM.
Q1_usr_apb_start_vio_i	1	Source	This signal allows you to trigger the APB requests manually. Pre-requisite: Before asserting this signal, you need to ensure that Q1_apb_rom_end and Q1_apb_done_vio_o signals are asserted, indicating that the APB master is available to grant the APB commands from this signal. This signal is ineffective if Q1_apb_rom_end or Q1_apb_done_vio_o is 0. Efinix recommends that you control this signal column by selecting <b>Active-High</b> from a drop-down list in the control column. This signal operates based on its rising edge, i.e., the assertion of this signal triggers 1 APB request. Prior to the assertion of this signal, APB signals (usr_apb_write, usr_apb_addr, and usr_apb_pwdata) need to be assigned and stable. To trigger new APB request, this signal must return to 0 first, then reassert to trigger a new APB request.
Q1_usr_apb_write_vio_i	1	Source	APB write for manual APB request. Must assign this signal prior to the triggering of Q1_usr_apb_start_vio_i.
Q1_usr_apb_addr_vio_i	24	Source	APB address for manual APB request. Must assign this signal prior to the triggering of Q1_usr_apb_start_vio_i.
Q1_usr_apb_pwdata_vio_i	32	Source	APB write data for manual APB request. Must assign this signal prior to the triggering of Q1_usr_apb_start_vio_i.
Q1_ram_usr_wren_vio_i	1	Source	This signal enables you to manually overwrite or clear the built-in RAM in the APB master controller. Pre-requisite: Before asserting this signal, you must set the intended address of the RAM through Q1_ram_usr_addr_vio_i. Efinix recommends that you control this signal column by selecting <b>Active-High</b> from a drop-down list in the control column.

Signal Name	Width	Probe / Source	Description
Q1_ram_usr_addr_vio_i	7	Source	Address of the built-in RAM. Set this address to read from or to write into the RAM.
Q1_ram_dout_d_vio_o	32	Source	Every APB read operation stores the APB PRDATA and the APB_PADDRESS into the built-in APB RAM in an ascending order.  If you set the <code>ram_usr_addr_vio_i</code> to <code>'h00</code> , the <code>ram_dout_d</code> and <code>ram_dout_a</code> display the APB PRDATA and APB_PADDRESS of the 1 <sup>st</sup> APB read command. Setting the <code>ram_usr_addr</code> to <code>'h01</code> displays the results of the 2 <sup>nd</sup> APB read command, and so forth.
Q1_ram_dout_a_vio_o	24	Source	

**Table 17: Efinity Virtual I/O Debugger Settings - XGMII Packet Transfer**

Signal Name	Width	Probe / Source	Description	
PLL_LOCKED	1	Probe	Lock signal from PLL, indicating that the clock sources ( <code>INIT_CLK</code> , <code>A1_APB_CLK</code> , and <code>DEBUG_CLK</code> ) are stable.  In the example design, the assertion of this signal also releases the resets in the USXGMII Multirate Ethernet core and the FPGA Ethernet USXGMII PCS.	
Q1_L0_transmit_reg	1	Probe	The assertion of this signal indicates that the pattern generator is transmitting data into the USXGMII Multirate Ethernet core through <code>{MAC_TXC, MAC_TXD}</code> .	
Q1_L1_transmit_reg	1	Probe		
Q1_L0_packet_cnt	8	Probe	Display the number of XGMII packet transmitted.	
Q1_L1_packet_cnt	8	Probe		
Q1_L0_mac_txc	8	Probe	XGMII interface which connect to user's (or MAC) logic.	
Q1_L0_mac_txd	64	Probe		
Q1_L0_mac_rxc	8	Probe		
Q1_L0_mac_rxd	64	Probe		
Q1_L1_mac_txc	8	Probe		
Q1_L1_mac_txd	8	Probe		
Q1_L1_mac_rxc	8	Probe		
Q1_L1_mac_rxd	64	Probe		
Q1_I0_mac_speed	3	Probe		Display the <code>mac_speed</code> . Refer to <a href="#">Table 3: APB Control Register for Data Rate Configurations</a> on page 18 for the decoded configuration.
Q1_I0_mac_speed	3	Probe		

**Table 18: Efinity Virtual I/O Debugger Settings - Checkers**

Signal Name	Width	Probe / Source	Description
PASS_STATUS	1	Probe	Ultimate status of the checkers
Q1_L0_status_4	1	Probe	The assertion of this signal indicates that the initial 10G data rate has successfully transferred 254 XGMII data packets.
Q1_L1_status_4	1	Probe	
Q1_L0_status_3	1	Probe	The assertion of this signal indicates that the Auto-Negotiation has successfully changed the data rate from 10G to 5G. Data transfer of 254 XGMII packets at 5G data rate is matching between TX and RX.
Q1_L1_status_3	1	Probe	
Q1_L0_status_2	1	Probe	The assertion of this signal indicates that the Auto-Negotiation has successfully changed the data rate from 5G to 2.5G. Data transfer of 254 XGMII Packets at 2.5G data rate is matching between TX and RX.
Q1_L1_status_2	1	Probe	
Q1_L0_status_1	1	Probe	The assertion of this signal indicates that the Auto-Negotiation has successfully changed the data rate from 2.5G to 1G. Data transfer of 254 XGMII packets at 1G data rate is matching between TX and RX.
Q1_L1_status_1	1	Probe	

# USXGMII Multirate Ethernet Testbench

The IP Manager also generates encrypted source code to allow you to simulate with various simulators for verification.

To generate the testbench deliverables:

1. Create a new project.
2. Select the **USXGMII Multirate Ethernet Core** from the IP Catalog and set your configurations.
3. In the **Deliverable** tab, ensure that you turn on one (or more) of the following:
  - **Testbench**
  - **Testbench/ncsim**
  - **Testbench/synopsys**
  - **Testbench/aldec**
  - **Testbench/modelsim**
4. Click **Generate**.
5. The testbench deliverables are available in **<project path>\ip\<ip-module-name>\Testbench directory**.

Contact Efinix support at the [Support Center](#) to obtain the testbench for the USXGMII Multirate Ethernet core with the Ethernet 10G MAC core and the FPGA Ethernet USXGMII PCS.

# Acronyms and Abbreviations

*Table 19: Acronyms and Abbreviations*

<b>Term</b>	<b>Definition</b>
APB	Advanced Peripheral Bus
AXI ST	Advanced eXtensible Interface Streaming
CTC	Clock Tolerance Compensation
FPGA	Field Programmable Gate Array
MAC	Media Access Control
PCS	Physical Coding Sublayer
PHY	The Physical Layer (the first and lowest layer in the seven-layer OSI model)
PLL	Phase Locked Loop
POR	Power On Reset
RX	Receiver Channel
SDC	Synopsys Design Constraints (defines the timing requirement for STA)
STA	Static Timing Analysis
TX	Transmit Channel
XGMII	10 Gigabit Media Independent Interface

# Revision History

*Table 20: Revision History*

<b>Date</b>	<b>Document Version</b>	<b>IP Version</b>	<b>Description</b>
February 2026	1.0	1.0	Initial release.