40- and 100-Gbps Ethernet MAC and PHY

MegaCore Function User Guide

101 Innovation Drive
San Jose, CA 95134
www.altera.com

UG-01088

Document last updated for Altera Complete Design Suite version: 12.0
Document publication date: June 2012

Feedback Subscribe

Contents

Appendix D. Address Map Changes for 11.0 SP1

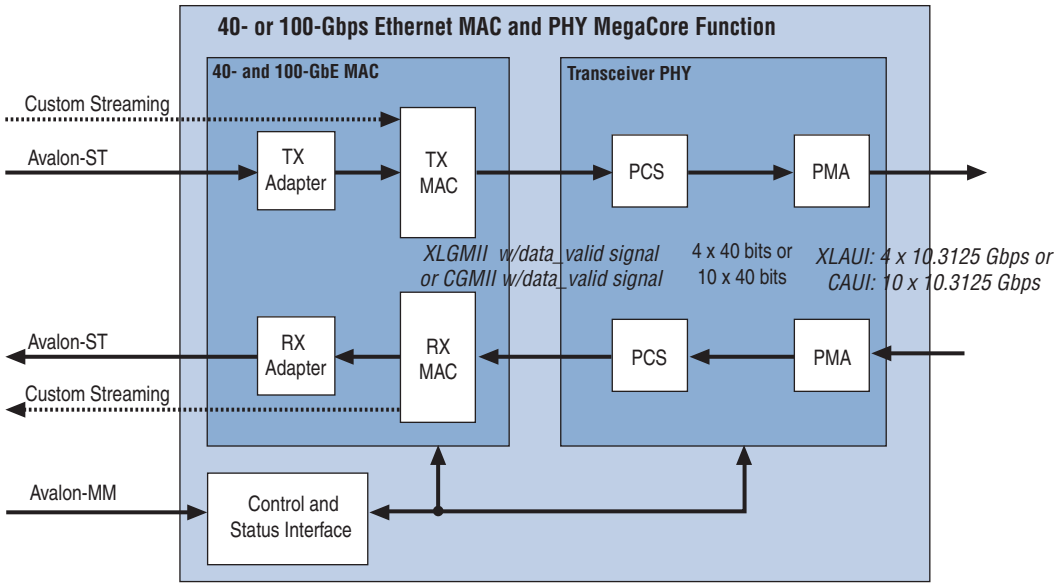
Additional Information

This document describes the Altera® 40- and 100-Gbps Ethernet (40GbE and 100GbE) MAC and PHY (PCS and PMA) MegaCore® functions which implement the *IEEE 802.3ba 40G and 100G Ethernet Standard*. This product is available for download from the *40-Gbps and 100-Gbps Ethernet MegaCore Function web page* and will be available as part of a future combined Quartus II software and Altera IP release.

This document describes both the 40- and 100-Gbps Ethernet MAC and PHY. In the interest of brevity, the remainder of this document frequently abbreviates the full product name, 40- and 100-Gbps Ethernet MAC and PHY MegaCore Function, to 40-100GbE IP core.

Figure 1–1 illustrates the 40GbE and 100GbE MAC and PHY MegaCore functions.

Figure 1–1. 40GbE and 100GbE MAC and PHY MegaCore Functions



The 40GbE and 100GbE MAC and PHY MegaCore functions provide an interface composed of a single, packet-based channel that is logically compatible with previous generation Ethernet systems. Depending on the variant you chose, the interface is either four or twenty 64b/66b encoded and scrambled virtual lane streams which are multiplexed to create either four or ten 10GbE transceiver PHY links. You can connect the transceiver PHY links which are compliant with IEEE 802.3ba standard XLAUI (40 Gbps) or CAUI (100 Gbps) interface directly to an external PMD optical module or to another device.

Supported Features

- Compliant with the *IEEE 802.3ba-2010 Higher Speed Ethernet Standard* available on the IEEE website (www.ieee.org)
- Soft physical coding sublayer (PCS) logic that interfaces seamlessly to Altera 10.3125 Gbps serial transceivers
- Standard XLAUI or CAUI external interface consisting of serial transceiver lanes each lane operating at 10.3125 Gbps
- Ethernet media access controller (MAC) supporting the 40GbE or 100GbE line rate at wire speed with a flexible and configurable feature set
- Avalon® Memory-Mapped (Avalon-MM) interface to access control and status registers
- Avalon Streaming (Avalon-ST) interface connects to client logic with the start of frame in the most significant byte (MSB) when optional adapters are used.
- Optional custom streaming interface with narrower bus width and a start frame possible on 64-bit boundaries without the optional adapters
- MAC, PHY, or MAC and PHY options configurable upon IP generation
- Transmit (TX) only configuration options
- Receive (RX) only configuration options
- Soft error detection on all internal RAMs for high reliability systems
- RX FIFO in MAC provide pass-through or store and forward frame processing
- Deficit idle counter (DIC) to maintain a 12-byte inter-packet gap average
- Ethernet flow control using the pause registers or pause interface
- Programmable maximum receive frame length up to 9600 bytes (jumbo frame)
- One MAC address register for configurable RX address filtering
- Optional TX source address insertion
- Promiscuous (transparent) and non-promiscuous (filtered) operation modes or received frame address filtering
- Configurable received frame filtering with cyclical redundancy check (CRC), runt, or oversized frame error
- Optional statistics counters
- Additional testbench logic to demonstrate Ethernet IP core behavior and customize the interface
- Statistics real-time output status signals vector
- Fault signaling: detects and reports local fault and generates remote fault
- TX and RX CRC pass-through control
- Hardware and software reset control
- TX MAC address insertion control

For a detailed specification of the Ethernet protocol refer to the IEEE 802.3ba-2010 40G and 100G Ethernet Standard available on the IEEE website.

Release Information

Table 1–1 provides information about this release of the 40-100GbE IP core.

Table 1–1. 40-100GbE IP Core Release Information

Item	Description
Version	12.0
Release Date	May 2012
Ordering Codes	IP-40GEMAC IP-40GEPHY IP-100GEMAC IP-100GEPHY IP-40GEMACPHY IP-100GEMACPHY
Product ID	40Gb Ethernet MAC: 00DF 40Gb Ethernet PHY: 00E0 100Gb Ethernet MAC: 00DD 100Gb Ethernet PHY: 00DE
Vendor ID	6AF7

The 40-100GbE IP core is intended for use with the Quartus II software, version 12.0.

Device Family Support

IP cores provide either final or preliminary support for target Altera device families. These terms have the following definitions:

- *Final support*—Verified with final timing models for this device.
- *Preliminary support*—Verified with preliminary timing models for this device.

Table 1–2 shows the level of support offered by the 40-100GbE IP core for Altera device families

Table 1–2. Device Family Support

MegaCore	Device Family	Support
40GbE	Stratix IV GT I1 or C2 speed grade	Preliminary
	Stratix V GX I2, I3, or C2 speed grade	Preliminary
	Other device families or grades	No support
100GbE	Stratix IV GT I1 or C2 speed grade	Preliminary
	Stratix V GX I2, I3, or C2 speed grade	Preliminary
	Other device families or grades	No support

Performance and Resource Utilization

The 40GbE and 100GbE IP cores generally require additional effort to close timing. This effort may include floor-planning using LogicLock™ to constrain the design into certain locations of your device, making use of advanced fitter settings such as Physical Synthesis, specifying Fitter effort levels, seed sweeping, and so on.

For more information about closing timing, refer to the *Volume2: Design Implementation and Optimization* in the *Quartus II Handbook*.

Table 1–3 shows the RTL hierarchy of the 40GbE IP cores for the Stratix IV and Stratix V devices.Table 1–4 provides the resource utilization for the 40GbE IP cores in a Stratix IV GT device using the C2 speed grade which is required. The recommended frequency for the IP core is 315 MHz, and the minimum frequency is 312.5 MHz. The adaptive look-up table (ALUT) and memory utilization vary based on the options and Quartus II settings that you choose. Resource utilization numbers (except M9Ks) are rounded to the nearest 100.

Table 1–3. 40GbE and 100GbE RTL Hierarchy

Directory	Subdirectory	Subdirectory	Subdirectory
alt_e*_adapter (1)	alt_e*_adapter_adapter_rx:adapter_rx		
	alt_e*_adapter_adapter_tx:adapter_tx		
	alt_e*:e*	alt_e*_mac:mac	alt_e*_mac_csr:mac_csr
			alt_e*_mac_rx:mac_rx
			alt_e*_mac_tx:mac_tx
		alt_e*_phy:phy	alt_e*_pcs_rx:pcs_rx
			alt_e*_pcs_tx:pcs_tx
			alt_e*_phy_csr:phy_csr
			alt_e*_pma:pma

Note to Table 1–3:

(1) In the file names shown, * denotes either 40 for 40GbE or 100 for 100GbE

Resource utilization changes when the statistics counters configuration parameter (`STATS_CNTRS_OPTION`) is enabled or disabled; Table 1–4 provides utilization for the 40GbE IP cores, with the `STATS_CNTRS_OPTION` parameter enabled and disabled, on the Stratix IV GT device.

Table 1–4. Resource Utilization Stratix IV GT Device (Part 1 of 2)

Module	Combinational ALUT	Memory ALUT	ALMs	Registers	Memory Block Bits	M9Ks
STATS_CNTRS_OPTION Enabled (Default)						
alt_e40_adapter	19400	1300	21700	31200	184300	20
alt_e40_adapter_adapter_rx:adapter_rx	600	100	600	100	0	0
alt_e40_adapter_adapter_tx:adapter_tx	200	100	400	800	0	0
alt_e40:e40	18600	1000	20600	29400	184300	20
alt_e40_mac:mac	9600	400	11800	19500	184300	20
alt_e40_mac_csr:mac_csr	4100	0	5300	9300	0	0
alt_e40_mac_rx:mac_rx	2700	200	3800	5800	184300	20
alt_e40_mac_tx:mac_tx	2900	200	3100	4400	0	0
alt_e40_phy:phy	8900	600	9000	9900	0	0
alt_e40_pcs_rx:pcs_rx	3800	300	3800	4400	0	0

Table 1–4. Resource Utilization Stratix IV GT Device (Part 2 of 2)

Module	Combinational ALUT	Memory ALUT	ALMs	Registers	Memory Block Bits	M9Ks
alt_e40_pcs_tx:pcs_tx	3600	300	4000	3900	0	0
alt_e40_phy_csr:phy_csr	600	0	700	1100	0	0
alt_e40_pma:pma	800	0	600	500	0	0
STATS_CNTRS_OPTION Disabled						
alt_e40_adapter	16000	1300	17800	24800	184300	20
alt_e40_adapter_adapter_rx:adapter_rx	600	100	600	100	0	0
alt_e40_adapter_adapter_tx:adapter_tx	200	100	400	800	0	0
alt_e40:e40	15200	1000	16800	23000	184300	20
alt_e40_mac:mac	6300	400	8200	13100	184300	20
alt_e40_mac_csr:mac_csr	700	0	1800	2900	0	0
alt_e40_mac_rx:mac_rx	2700	200	3800	5800	184300	20
alt_e40_mac_tx:mac_tx	2900	200	3100	4400	0	0
alt_e40_phy:phy	8900	600	8800	9900	0	0
alt_e40_pcs_rx:pcs_rx	3800	300	3700	4400	0	0
alt_e40_pcs_tx:pcs_tx	3600	300	3800	3900	0	0
alt_e40_phy_csr:phy_csr	700	0	700	1100	0	0
alt_e40_pma:pma	800	0	600	500	0	0

Some resource utilization numbers decrease when the `STATS_CNTRS_OPTION` parameter is disabled. For example, in Table 1–4, compare the values for `alt_e40_mac_csr:mac_csr`; when counters are enabled, there are 4100 combinational ALUTs, but when the counters are disabled, there are 700 combinational ALUTs; this is a decrease of 3400 combinational ALUTs.

Table 1–5 provides the resource utilization for the 40GbE IP cores in a Stratix V device using the C2 speed grade. The adaptive look-up table (ALUT) and memory utilization vary based on the options and Quartus II settings that you choose. Resource utilization numbers (except M20Ks) are rounded to the nearest 100.

Resource utilization changes when the statistics counters configuration parameter (`STATS_CNTRS_OPTION`) is enabled or disabled; Table 1–5 provides utilization for the 40GbE IP cores, with the `STATS_CNTRS_OPTION` parameter enabled and disabled, on the Stratix V device.

Table 1–5. Resource Utilization Stratix V Device (Part 1 of 2)

Module	Combinational ALUT	ALMs	Registers	Memory Block Bits	M20Ks
STATS_CNTRS_OPTION Enabled (Default)					
alt_e40_adapter	20800	22800	31000	223200	15
alt_e40_adapter_adapter_rx:adapter_rx	600	700	1000	0	0
alt_e40_adapter_adapter_tx:adapter_tx	200	400	700	0	0
alt_e40:e40	20000	21600	29300	223200	15

Table 1–5. Resource Utilization Stratix V Device (Part 2 of 2)

Module	Combinational ALUT	ALMs	Registers	Memory Block Bits	M20Ks
alt_e40_mac:mac	9600	12000	19500	184300	9
alt_e40_mac_csr:mac_csr	4000	5500	9700	0	0
alt_e40_mac_rx:mac_rx	2700	3500	5900	184300	9
alt_e40_mac_tx:mac_tx	2900	3000	4300	0	0
alt_e40_phy:phy	10400	9600	9800	38900	6
alt_e40_pcs_rx:pcs_rx	3700	3700	3900	0	0
alt_e40_pcs_tx:pcs_tx	3600	3500	3400	0	0
alt_e40_phy_csr:phy_csr	800	700	1100	0	0
alt_e40_pma:pma	2200	1700	1400	38900	6
STATS_CNTRS_OPTION Disabled					
alt_e40_adapter	17400	19400	24600	223200	15
alt_e40_adapter_adapter_rx:adapter_rx	600	800	900	0	0
alt_e40_adapter_adapter_tx:adapter_tx	200	400	700	0	0
alt_e40:e40	16600	18200	23000	223200	15
alt_e40_mac:mac	6300	8600	13200	184300	9
alt_e40_mac_csr:mac_csr	700	2000	3000	0	0
alt_e40_mac_rx:mac_rx	2700	3500	5900	184300	9
alt_e40_mac_tx:mac_tx	2900	3100	4300	0	0
alt_e40_phy:phy	10300	9600	9800	38900	6
alt_e40_pcs_rx:pcs_rx	3700	3700	3900	0	0
alt_e40_pcs_tx:pcs_tx	3600	3500	3400	0	0
alt_e40_phy_csr:phy_csr	800	700	1100	0	0
alt_e40_pma:pma	2200	1700	1400	38900	6

Table 1–6 provides the resource utilization for the 100GbE IP cores in a Stratix IV GT device using the C2 speed grade which is required. The minimum frequency for the IP core is 312.5 MHz. The adaptive look-up table (ALUT) and memory utilization vary based on the options and Quartus II settings that you choose. Resource utilization numbers (except M9Ks) are rounded to the nearest 100.

Resource utilization changes when the statistics counters configuration parameter (`STATS_CNTRS_OPTION`) is enabled or disabled; Table 1–6 provides utilization for the 100GbE IP cores, provides utilization for the 100GbE IP cores, with the `STATS_CNTRS_OPTION` parameter enabled and disabled, on the Stratix IV GT device.

Table 1–6. Resource Utilization Stratix IV Device (Part 1 of 2)

Module	Combinational ALUT	Memory ALUT	ALMs	Registers	Memory Block Bits	M9Ks
STATS_CNTRS_OPTION Enabled (Default)						
alt_e100_adapter	51300	5400	67400	97900	262100	29
alt_e100_adapter_adapter_rx:adapter_rx	1100	0	4400	6300	151600	17

Table 1–6. Resource Utilization Stratix IV Device (Part 2 of 2)

Module	Combinational ALUT	Memory ALUT	ALMs	Registers	Memory Block Bits	M9Ks
alt_e100_adapter_adapter_tx:adapter_tx	400	900	4100	6400	0	0
alt_e100:e100	49800	4400	59000	85100	110600	12
alt_e100_mac:mac	17700	1200	25000	37600	110600	12
alt_e100_mac_csr:mac_csr	4100	0	5500	9300	0	0
alt_e100_mac_rx:mac_rx	5500	500	8200	12200	0	0
alt_e100_mac_tx:mac_tx	8200	700	11700	16100	0	0
alt_e100_phy:phy	32100	3200	34300	47500	0	0
alt_e100_pcs_rx:pcs_rx	19400	1600	18600	28900	0	0
alt_e100_pcs_tx:pcs_tx	10900	1600	14200	16400	0	0
alt_e100_phy_csr:phy_csr	900	0	1200	1700	0	0
alt_e100_pma:pma	900	0	600	500	0	0
STATS_CNTRS_OPTION Disabled						
alt_e100_adapter	47933	5400	64100	91500	262100	29
alt_e100_adapter_adapter_rx:adapter_rx	1100	0	4600	6300	151600	17
alt_e100_adapter_adapter_tx:adapter_tx	400	900	4300	6400	0	0
alt_e100:e100	46500	4400	55200	78800	110600	12
alt_e100_mac:mac	14300	1200	21400	31300	110600	12
alt_e100_mac_csr:mac_csr	700	0	1900	2900	0	0
alt_e100_mac_rx:mac_rx	5500	500	8300	12200	0	0
alt_e100_mac_tx:mac_tx	8200	700	11600	16100	0	0
alt_e100_phy:phy	32100	3200	34000	47500	0	0
alt_e100_pcs_rx:pcs_rx	19400	1600	18700	28900	0	0
alt_e100_pcs_tx:pcs_tx	10900	1600	13800	16400	0	0
alt_e100_phy_csr:phy_csr	900	0	1200	1700	0	0
alt_e100_pma:pma	900	0	600	500	0	0

Table 1–7 provides the resource utilization for the 100GbE IP cores in a Stratix V device using the C2 speed grade. The adaptive look-up table (ALUT) and memory utilization vary based on the options and Quartus II settings that you choose. Resource utilization numbers (except M20Ks) are rounded to the nearest 100.

Resource utilization changes when the statistics counters configuration parameter (`STATS_CNTRS_OPTION`) is enabled or disabled; Table 1–7 provides utilization for the 100GbE IP cores, with the `STATS_CNTRS_OPTION` parameter enabled and disabled, on the Stratix V device.

Table 1–7. Resource Utilization Stratix V Device (Part 1 of 2)

Module	Combinational ALUT	ALMs	Registers	Memory Block Bits	M20Ks
STATS_CNTRS_OPTION Enabled (Default)					
alt_e100_adapter	53100	60100	91300	301100	34

Table 1–7. Resource Utilization Stratix V Device (Part 2 of 2)

Module	Combinational ALUT	ALMs	Registers	Memory Block Bits	M20Ks
alt_e100_adapter_adapter_rx:adapter_rx	1100	3300	6500	151600	17
alt_e100_adapter_adapter_tx:adapter_tx	400	3200	4900	0	0
alt_e100:e100	51700	53600	80000	149500	17
alt_e100_mac:mac	17400	23300	37200	110600	11
alt_e100_mac_csr:mac_csr	4000	5600	9400	0	0
alt_e100_mac_rx:mac_rx	5400	7400	12400	110600	11
alt_e100_mac_tx:mac_tx	7900	10300	15400	0	0
alt_e100_phy:phy	34200	30300	42800	38900	6
alt_e100_pcs_rx:pcs_rx	19300	15700	25800	0	0
alt_e100_pcs_tx:pcs_tx	10500	11000	13200	0	0
alt_e100_phy_csr:phy_csr	900	1100	1700	0	0
alt_e100_pma:pma	3500	2500	2100	38900	6
STATS_CNTRS_OPTION Disabled					
alt_e100_adapter	49800	57600	84900	301100	34
alt_e100_adapter_adapter_rx:adapter_rx	1100	3400	6500	151600	17
alt_e100_adapter_adapter_tx:adapter_tx	400	3300	4900	0	0
alt_e100:e100	48900	50900	73500	149500	17
alt_e100_mac:mac	14100	20600	30700	110600	11
alt_e100_mac_csr:mac_csr	700	1900	3000	0	0
alt_e100_mac_rx:mac_rx	5400	7400	12400	110600	11
alt_e100_mac_tx:mac_tx	7900	11200	15300	0	0
alt_e100_phy:phy	34200	30300	42800	38900	6
alt_e100_pcs_rx:pcs_rx	19300	15700	25800	0	0
alt_e100_pcs_tx:pcs_tx	10500	11000	13200	0	0
alt_e100_phy_csr:phy_csr	900	1100	1700	0	0
alt_e100_pma:pma	3500	2500	2100	38900	6

The 40-100GbE IP cores can support full wire line speed with a 64-byte frame length and back-to-back or mixed length traffic, up to a 9600-byte frame size, with no drops.

2. Getting Started

This chapter explains how to install, parameterize, and simulate the 40-100GbE IP core. It includes the following sections:

- Installing the IP Core
- Parameterizing the Ethernet IP Core
- Understanding the Testbenches
- Simulating the 40-100GbE IP Core

Installing the IP Core

Complete the following steps to install the core:

1. If you have not already done so, install the following software:
   a. The Quartus II software version 12.0.
   b. The simulation libraries which are part of the standard Quartus II installation. You only need to install the simulation files for the simulator that you plan to use. The Quartus II simulation libraries are located at <*install_dir*>/**eda**/**sim_lib**.
2. Go to the 40- and 100-Gbps Ethernet Example Design web page.
3. Copy the **alt_eth40g_12.0.zip** file or the **alt_eth100g_12.0.zip** file to your workspace. These two files include directories for four Ethernet designs, for 8 Ethernet designs in total. Table 2–1 describes the client and PHY interfaces for each of these designs. Refer to "High Level System Overview" on page 3–1 for additional comparisons between the eight design options.
4. Unzip the file in your working directory, <*project_dir*>. Figure 2–2 shows the directory structure for **alt_eth40g_12.0.zip** and **alt_eth100g_12.0.zip**.

Table 2–1 shows the client and PHY interfaces for the 40-100GbE IP core designs.

Table 2–1. Interface Comparisons for All Testbenches

Directory Name	Subdirectories	Client Side Datapath Interface (1)	PHY Side Interface
alt_eth_40g\sim_verilog\example_testbenches\	**SIV_alt_e40_avalon_tb** (2), includes adapters	Avalon-ST, 256 bits@312.5 MHz	XLAUI
	SIV_alt_e40_custom_tb (3)	Custom streaming, 128 bits@312.5 MHz	XLAUI
	SV_alt_e40_avalon_tb (2), includes adapters	Avalon-ST, 256 bits@312.5 MHz	XLAUI
	SV_alt_e40_custom_tb (3)	Custom streaming, 128 bits@312.5 MHz	XLAUI
alt_eth_100g\sim_verilog\example_testbenches\	**SIV_alt_e100_avalon_tb** (2), includes adapters	Avalon-ST, 512 bits@312.5 MHz	CAUI
	SIV_alt_e100_custom_tb (3)	Custom streaming, 320 bits@312.5 MHz	CAUI
	SV_alt_e100_avalon_tb (2), includes adapters	Avalon-ST, 512 bits@312.5 MHz	CAUI
	SV_alt_e100_custom_tb (3)	Custom streaming, 320 bits@312.5 MHz	CAUI

Note to Table 2–1:

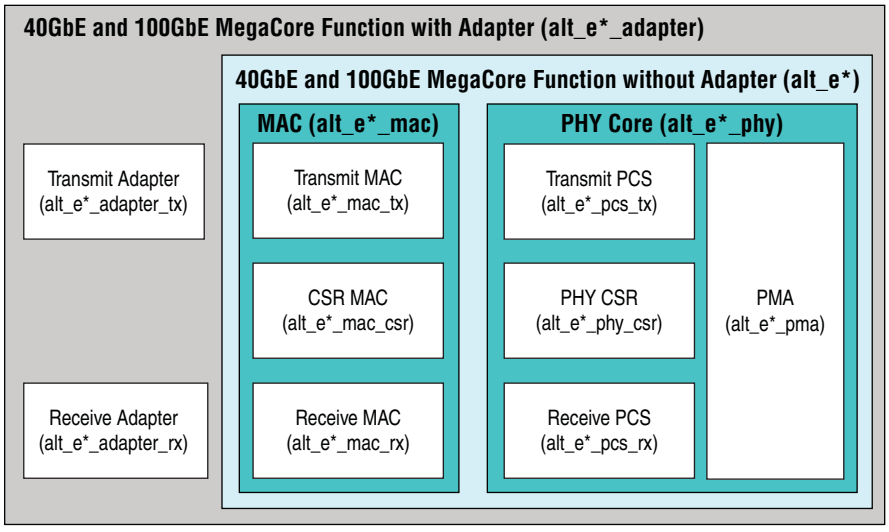
(1) 315 MHz is the recommended clock frequency for all designs; 312.5 MHz is the minimum frequency.

(2) Designs that include the adapters always place the start of packet in the MSB.

(3) In designs that do not include adapters, a packet can start a packet at any 64-bit lane boundary.

Figure 2–1 shows the RTL hierarchy and wrapper files for the IP core directory structure. The wrappers can be parameterized for TX or RX, Stratix IV or Stratix V.

Figure 2–1. RTL Hierarchy and Wrappers (1)



Note to Figure 2–1:

(1) In the file names shown, * denotes either 40 for 40GbE or 100 for 100GbE.

There are no top-level wrapper files provided for the 40GbE or 100GbE MAC IP cores with adapters. Two unencrypted top-level adapter files are provided to accommodate.

The unencrypted 40GbE top-level adapter files are located in the **alt_eth_40g_12.0.zip** file at:

- `alt_eth_40g\quartus_synth\rtl_src\adapter\rx\alt_e40_adapter_rx.v` and
- `alt_eth_40g\quartus_synth\rtl_src\adapter\tx\alt_e40_adapter_tx.v`

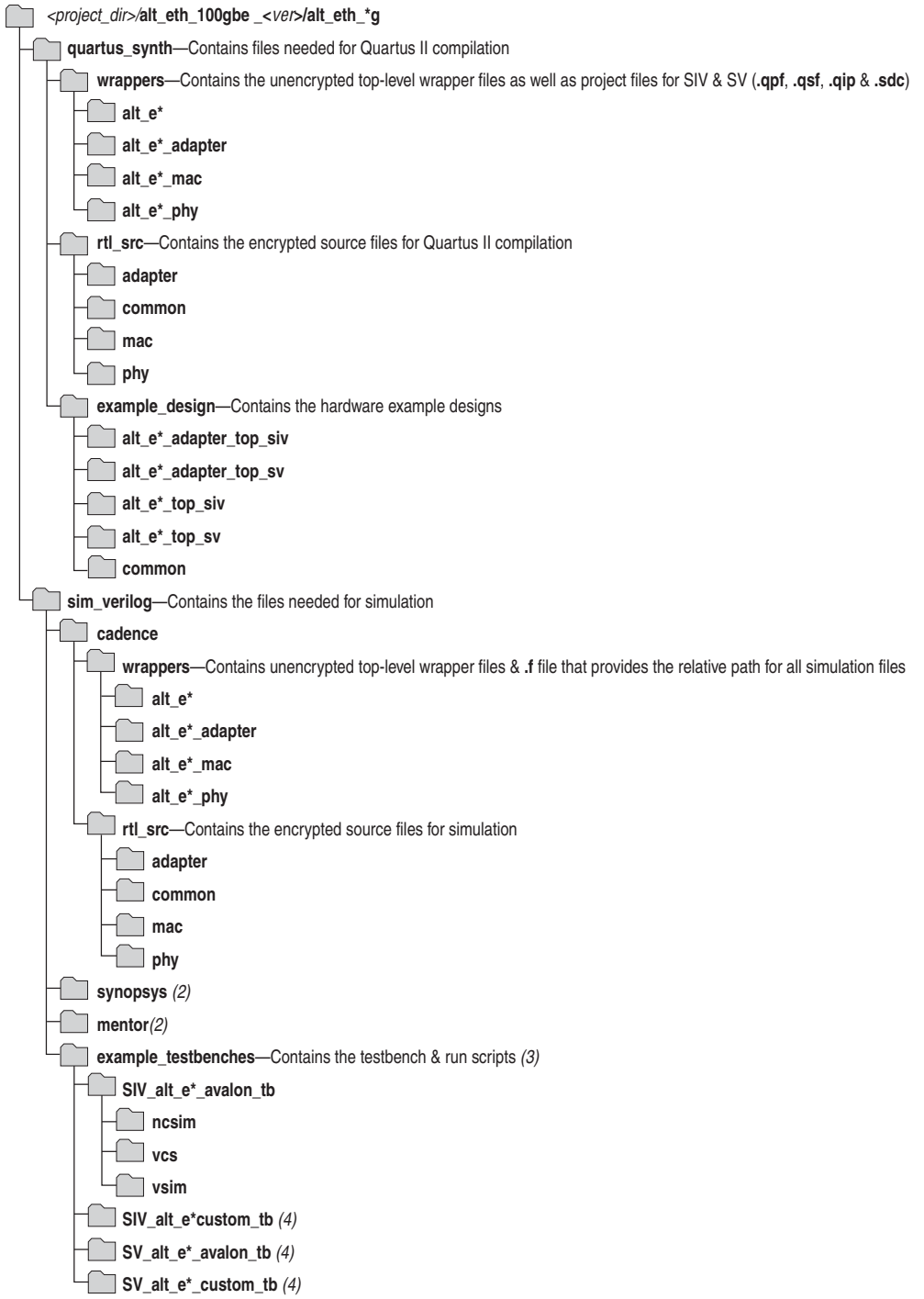
The unencrypted 100GbE top-level adapter files are located in the **alt_eth_100g_12.0.zip** file at:

- `alt_eth_100g\quartus_synth\rtl_src\adapter\rx\alt_e100_adapter_rx.v` and
- `alt_eth_100g\quartus_synth\rtl_src\adapter\tx\alt_e100_adapter_tx.v`

The `alt_e100_adapter.v` wrapper file describes how to hook up these adapter files. For example, in the **alt_e*_adapter** wrapper file (where * represents 40 for 40GbE IP cores and * represents 100 for 100GbE IP cores), the **alt_e*_adapter_rx** and **alt_e*_adapter_tx** modules are instantiated and the outputs are connected to the **alt_e*** module's MAC-level interface signals. Similarly, you can create a top-level wrapper file by instantiating the **alt_e*_adapter_rx** and **alt_e*_adapter_tx** modules and connecting the outputs to the **alt_e*_mac** wrapper.

An example directory structure for the 40-100GbE IP core, including all possible configurations of 40 Gbps and 100 Gbps bandwidth, is shown in Figure 2–2.

Figure 2–2. Directory Structure for the 40GbE and 100GbE Packages



Note to Figure 2–2:

(1) In the file names shown, * denotes either 40 for 40GbE or 100 for 100GbE.

(2) The directories located in the **synopsis** and **mentor** folders are identical to the directory shown in the **cadence** folder.

(3) For more information refer to "Testbench with Adapters File Descriptions" on page 2–8 and "Testbench without Adapters File Descriptions" on page 2–11.

(4) The directories located in the SIV_alt_e*_custom_tb, SV_alt_e*_avalon_tb, and SV_alt_e*_custom_tb folders are identical to the directory shown in the SIV_alt_e*_avalon_tb folder.

Frequency Assignment

There is a 0 parts per million (ppm) frequency assignment.

To accommodate the 0 ppm for Stratix IV device projects, the following lines must be included in the project **.qsf**:

```
    set_instance_assignment -name GXB_0PPM_CORE_CLOCK ON -from tx_dataout* -to
tx_dataout*

    set_instance_assignment -name GXB_0PPM_CORE_CLOCK ON -from rx_datain* -to
rx_datain*
```

To accommodate the 0 ppm for Stratix V device projects, the following lines must be included in the project **.qsf**:

```
    set_instance_assignment -name GXB_0PPM_CORECLK ON -to tx_dataout*

    set_instance_assignment -name GXB_0PPM_CORECLK ON -to rx_datain*
```

For both Stratix IV and Stratix V projects, `tx_dataout` and `rx_datain` are the top-level pin names of the output and input high-speed serial lines from the transceivers, respectively.

Parameterizing the Ethernet IP Core

Table 2–2 describes parameters you can set to customize the Ethernet IP core and simulation testbench.

Table 2–2. Parameters for the Ethernet Modules and Simulation (Part 1 of 2) (1)

Name	Values	Description
alt_e* Wrapper Parameters		
FAST_SIMULATION	1	This parameter configures simulation specific settings to enable faster simulation time. Always set this parameter to 1 when simulating.
VARIANT	Default value: 3 Value range: 1-3	Set to 1 for RX, set to 2 for TX, or set to 3 for duplex (RX_AND_TX).
DEVICE_FAMILY	Default value: Stratix V Value range: Stratix V or Stratix IV	Selects the Stratix IV or Stratix V device family.
ENABLE_STATISTICS_CNTR	Default value: true Value range: true or false	Enables the statistics counters when parameter is set to true.
WORDS	40GbE: 2 100GbE: 5	Do not change this parameter.
alt_e*_adapter Wrapper Parameters		
FAST_SIMULATION	1	This parameter configures simulation specific settings to enable faster simulation time. Always set this parameter to 1 when simulating.
VARIANT	Default value: 3 Value range: 1-3	Set to 1 for RX, set to 2 for TX, or set to 3 for duplex (RX_AND_TX).
DEVICE_FAMILY	Default value: Stratix V Value range: Stratix V or Stratix IV	Selects the Stratix V or Stratix V device family.

Table 2–2. Parameters for the Ethernet Modules and Simulation (Part 2 of 2) (1)

Name	Values	Description
ENABLE_STATISTICS_CNTR	Default value: true Value range: true or false	Enables the statistics counters when parameter is set to true.
alt_e*_mac Wrapper Parameters		
VARIANT	Default value: 3 Value range: 1-3	Set to 1 for RX, set to 2 for TX, or set to 3 for duplex (RX_AND_TX).
DEVICE_FAMILY	Default value: Stratix V Value range: Stratix V or Stratix IV	Selects the Stratix IV or Stratix V device family.
ENABLE_STATISTICS_CNTR	Default value: true Value range: true or false	Enables the statistics counters when parameter is set to true.
WORDS	40GbE: 2 100GbE: 5	Do not change this parameter.
alt_e*_phy Wrapper Parameters		
FAST_SIMULATION	1	This parameter configures simulation specific settings to enable faster simulation time. Always set this parameter to 1 when simulating.
VARIANT	Default value: 3 Value range: 1-3	Set to 1 for RX, set to 2 for TX, or set to 3 for duplex (RX_AND_TX).
DEVICE_FAMILY	Default value: Stratix V Value range: Stratix V or Stratix IV	Selects the Stratix IV or Stratix V device family.
WORDS	40GbE: 2 100GbE: 5	Do not change this parameter.

Note to Table 2–2:

(1) In the file names shown, * denotes either 40 for 40GbE or 100 for 100GbE.

Table 2–3 shows the 40-100GbE IP core wrapper files. Refer to "Port Listing by Transmission Mode Parameter Configuration" on page 3–31 for information on port assignments for the transmission mode (RX, TX, and duplex) parameter settings.

Table 2–3. 40GbE-100GbE IP Core Wrapper Files

Subdirectory	Subdirectory\Wrapper File
alt_eth40g_12.0.zip	
alt_eth_40g\quartus_synth\wrappers	alt_e40\alt_e40.v
	alt_e40_adapter\alt_e40_adapter.v
	alt_e40_mac\alt_e40_mac.v
	alt_e40_phy\alt_e40_phy.v
alt_eth100g_12.0.zip	
alt_eth_100g\quartus_synth\wrappers	alt_e100\alt_e100.v
	alt_e100_adapter\alt_e100_adapter.v
	alt_e100_mac\alt_e100_mac.v
	alt_e100_phy\alt_e100_phy.v

The wrapper subdirectories in the **alt_eth40g_12.0.zip** and **alt_eth100g_12.0.zip** files also include the **.qpf**, **.qip**, **.qsf**, and **.sdc** project files for both Stratix IV and Stratix V.

The *Quartus II Handbook* provides more information on the Quartus II software project files.

Understanding the Testbenches

The TX and RX interfaces between the client application and the MAC use 64-bit words. The optional adapters guarantee that the start of packet (SOP) is always in the MSB, and they increase the width of the client interfaces as follows:

- 40GbE, both TX and RX buses:
  - IP core without adapters—2 words (128 bits)
  - IP core with adapters—4 words (256 bits)
- 100GbE, both TX and RX buses:
  - IP core without adapters—5 words (320 bits)
  - IP core with adapters—8 words (512 bits)

Altera offers four 40GbE and four 100GbE example testbenches for the following configurations:

- The 40GbE IP core with four- to two-word adapters (256 bits to 128 bits) and the Avalon-MM/Avalon-ST interface on a Stratix IV device (**SIV_alt_e40_avalon_tb**)
- The 40GbE IP core with four- to two-word adapters (256 bits to 128 bits) and the Avalon-MM/Avalon-ST interface on a Stratix V device (**SV_alt_e40_avalon_tb**)
- The 40GbE IP core with no adapters and a custom interface a Stratix IV device (**SIV_alt_e40_custom_tb**)
- The 40GbE IP core with no adapters and a custom interface a Stratix V device (**SV_alt_e40_custom_tb**)
- The 100GbE IP core with eight- to five-word adapters (512 bits to 320 bits) and the Avalon-MM/Avalon-ST interface on a Stratix IV device (**SIV_alt_e100_avalon_tb**)
- The 100GbE IP core with eight- to five-word adapters (512 bits to 320 bits) and the Avalon-MM/Avalon-ST interface on a Stratix V device (**SV_alt_e100_avalon_tb**)
- The 100GbE IP core with no adapters and a custom interface on a Stratix IV device (**SIV_alt_e100_custom_tb**)
- The 100GbE IP core with no adapters and a custom interface on a Stratix V device (**SV_alt_e100_custom_tb**)

Conceptually, the testbenches for the 40-100GbE IP cores with adapters are identical, and the testbenches for the 40-100GbE IP cores without adapters are identical, except for the bandwidth. The following sections first describe the testbenches that include adapters and then describe the testbenches without adapters.

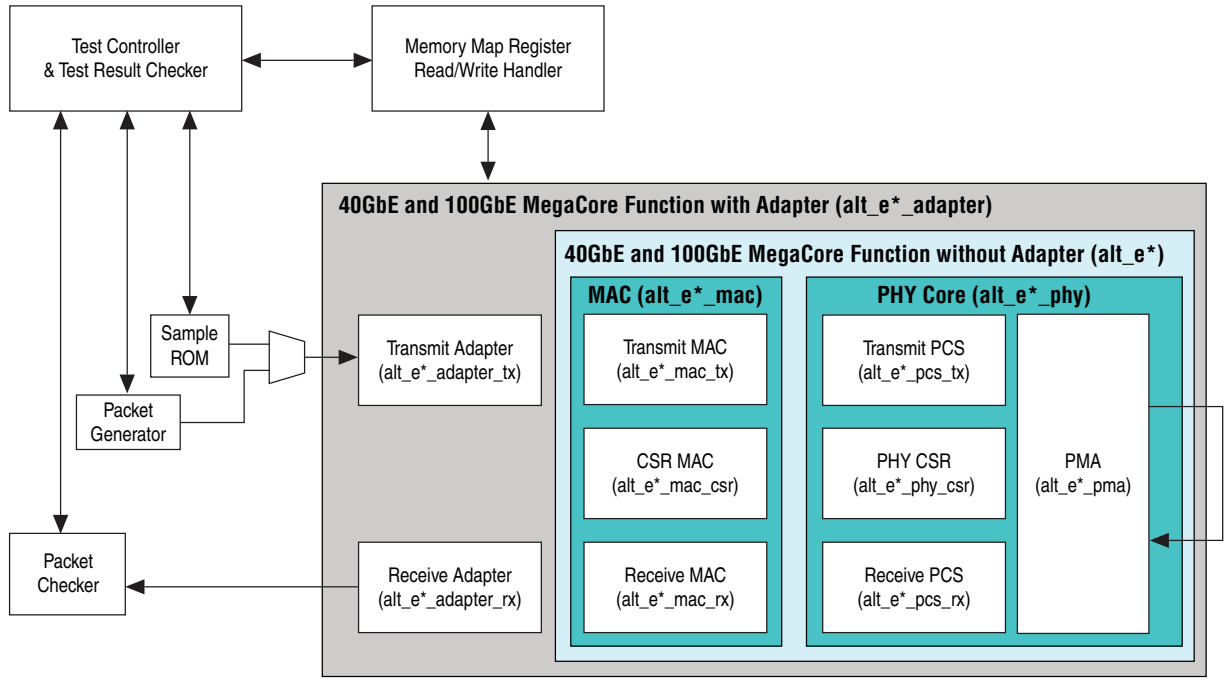
Each of the eight example testbenches illustrate packet traffic, error behavior, and asynchronous reset recovery, in addition to providing information regarding the transceiver PHY.

All of the testbenches support ModelSim® SE, version 10.0d and later, as well as Cadence NCSim, version 11.10.007 and Synopsis VCS version 2011.12.

Testbench with Adapters

Figure 2–3 illustrates the top-level modules of the 40GbE and 100GbE example testbenches that use adapters.

Figure 2–3. Testbench with Adapters (1)



Note to Figure 2–3:

(1) In the file names shown, * denotes either 40 for 40GbE or 100 for 100GbE

Table 2–4 lists the key files that implement the **SIV_alt_e40_avalon_tb**, **SV_alt_e40_avalon_tb**, **SIV_alt_e100_avalon_tb**, and **SV_alt_e100_avalon_tb** example testbenches.

Table 2–4. Testbench with Adapters File Descriptions (Part 1 of 2)

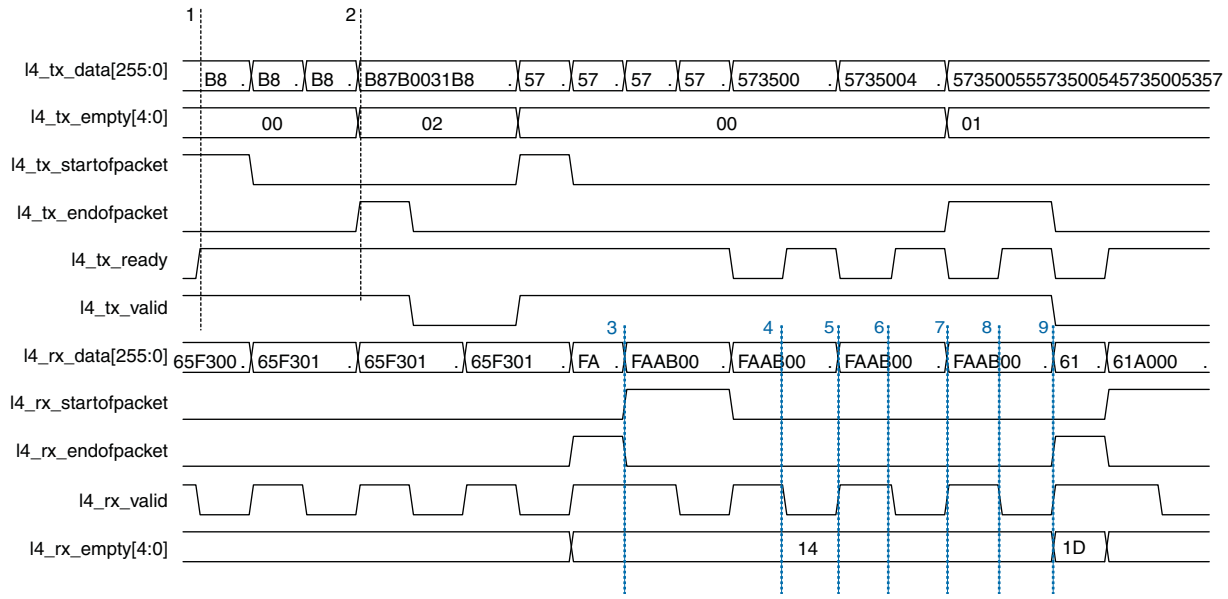
File Name	Description
Testbench and Simulation Files	
alt_e40_avalon_tb.sv **alt_e100_avalon_tb.sv**	The testbench as shown in Figure 2–3.
alt_e40_avalon_tb_packet_gen_.v **alt_e100_avalon_tb_packet_gen_.v**	The packet generator.
alt_e40_avalon_tb_packet_gen_sanity_check.v **alt_e100_avalon_tb_packet_gen_sanity_check.v**	The packet checker.

Table 2–4. Testbench with Adapters File Descriptions (Part 2 of 2)

File Name	Description
alt_e40_avalon_tb_sample_tx_rom.hex **alt_e100_avalon_tb_sample_tx_rom.hex**	The sample TX ROM.
alt_e40_avalon_tb_sample_tx_rom.v **alt_e100_avalon_tb_sample_tx_rom.v**	Lists the contents of the sample TX ROM (**alt_e40_avalon_tb_sample_tx_rom.hex** and **alt_e100_avalon_tb_sample_tx_rom.hex**).
Testbench Scripts	
run_vsim.do	The ModelSim script to run the testbench.
run_ncsim.sh	The Synopsis VCS script to run the testbench.
run_vcs.sh	The Cadence NCSim script to run the testbench.

Figure 2–4 shows typical traffic from the simulation testbench created using the **alt_e40_avalon_tb_run_vsim.do** script in ModelSim. Figure 2–4 shows the client-side Avalon-ST interface for the 40GbE IP core using the four- to two-word adapters.

Figure 2–4. Typical 40GbE Traffic on the Avalon-ST Interface Using the Four- to Two-Word adapters



The markers in Figure 2–4 show the following sequence of events:

1. At marker 1, `l4_tx_startofpacket` is asserted, indicating the beginning of a TX packet.
2. At marker 2, `l4_tx_endofpacket` is asserted, indicating the end of the TX packet, and `l4_tx_empty[4:0]` indicates that the 2 least significant bytes of the last data cycle are empty.
3. At marker 3, `l4_rx_startofpacket` is asserted, indicating the beginning of an RX packet. A second transfer has already started on the TX bus.

4. At marker 4, l4_rx_valid is deasserted, indicating that the 40GbE IP core does not have new valid data to send to the client on l4_rx_data[255:0]. l4_rx_data[255:0] remains valid and unchanged for a second cycle.
5. A marker 5, l4_rx_valid is asserted, indicating that the 40GbE IP core has valid data to send to the client on l4_rx_data[255:0].
6. At marker 6, l4_rx_valid is deasserted, indicating that the 40GbE IP core does not have new valid data to send to the client on l4_rx_data[255:0]. l4_rx_data[255:0] remains unchanged for a second cycle.
7. A marker 7, l4_rx_valid is asserted, indicating that the 40GbE IP core has valid data to send to the client on l4_rx_data[255:0].
8. At marker 8, l4_rx_valid is deasserted, indicating that the 40GbE IP core does not have new valid data to send to the client on l4_rx_data[255:0]. l4_rx_data[255:0] remains unchanged for a second cycle.
9. At marker 9, l4_rx_endofpacket is asserted indicating the end of the RX packet. l4_rx_empty[4:0] has a value of 0x1D, indicating that 29 least significant bytes of the last cycle of the RX packet empty.

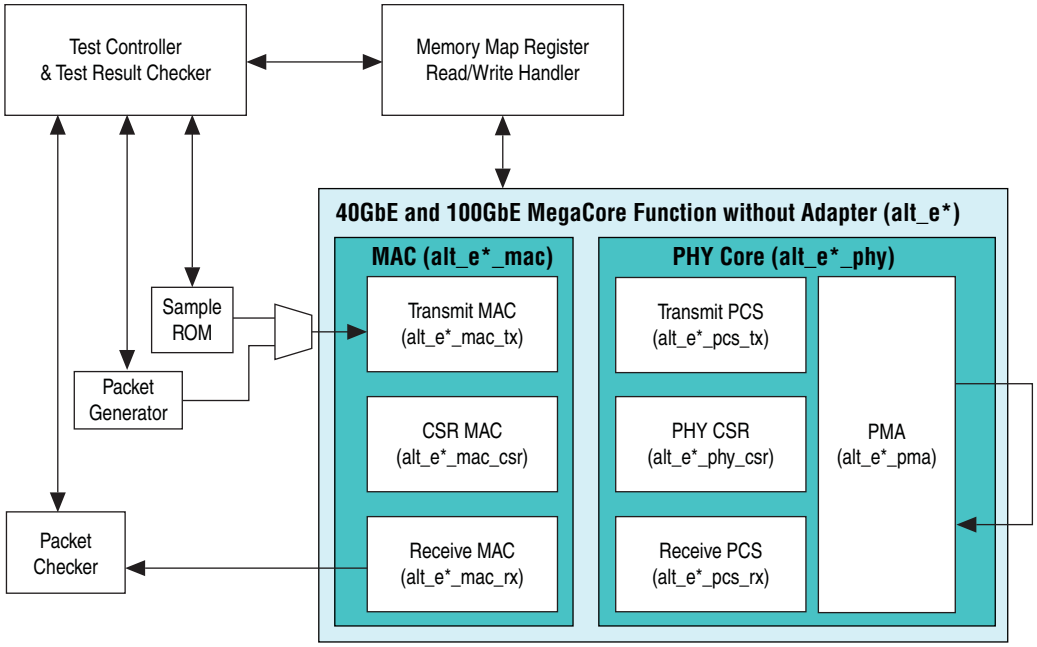
Early access versions of the 40-100GbE IP core with adapters had a read latency of 1. The read latency for all cores with adapters is now 0.

For more information about the Avalon-ST protocol, refer to the *Avalon Interface Specifications*.

Testbenches without Adapters

Figure 2–5 illustrates the top-level modules of the 40GbE and 100GbE example testbenches that do not use adapters.

Figure 2–5. Testbench without Adapters (1)



Note to Figure 2–5:

(1) In the file names shown, * denotes either 40 for 40GbE or 100 for 100GbE

Table 2–5 lists the key files that implement the **SIV_alt_e40_custom_tb**, **SV_alt_e40_custom_tb**, **SIV_alt_e100_custom_tb**, and **SV_alt_e100_custom_tb** example testbenches.

Table 2–5. Testbench without Adapters File Descriptions (Part 1 of 2)

File Name	Description
Testbench and Simulation Files	
alt_e40_custom_tb.sv **alt_e100_custom_tb.sv**	The testbench as shown in Figure 2–5
alt_e40_custom_tb_packet_gen_.v **alt_e100_custom_tb_packet_gen_.v**	The packet generator
alt_e40_custom_tb_packet_gen_sanity_check.v **alt_e100_custom_tb_packet_gen_sanity_check.v**	The packet sanity checker
alt_e40_custom_tb_sample_tx_rom.hex **alt_e100_custom_tb_sample_tx_rom.hex**	The sample TX ROM
alt_e40_custom_tb_sample_tx_rom.v **alt_e100_custom_tb_sample_tx_rom.v**	Lists the contents of the sample TX ROM (**alt_e40_custom_tb_sample_tx_rom.v** and **alt_e100_custom_tb_sample_tx_rom.v**)
Testbench Scripts	

Table 2–5. Testbench without Adapters File Descriptions (Part 2 of 2)

File Name	Description
run_vsim.do	The ModelSim script to run the testbench.
run_ncsim.sh	The Synopsis VCS script to run the testbench.
run_vcs.sh	The Cadence NCSim script to run the testbench.

Simulating the 40-100GbE IP Core

You can simulate the 40-100GbE IP core using ModelSim® SE, version 10.0d, Cadence NCSim, version 11.10.007, or Synopsis VCS version 2011.12.

There are four example testbenches for the 40-100GbE IP core (**alt_e100_avalon_tb.sv**, **alt_e40_avalon_tb.sv**, **alt_e100_custom_tb.sv** and **alt_e40_custom_tb.sv**). The example testbenches simulate packet traffic and error behavior at the digital level. The testbench does not require special SystemVerilog class libraries.

The top-level testbench file consists of a simple packet generator and checker and one core in a loopback configuration. The packet generator skews and reorders its transmitter digital output to emulate actual transceiver behavior and optical cabling lane permutations. The test procedure also injects some errors and confirms the appropriate response by the lanes and the higher level protocol.

The example testbenches contain the testbench and run scripts for the ModelSim, Cadence, and Synopsis simulators. The run scripts are written to use the file lists in the wrapper files and are found by default when simulations are launched from the original location. You can access design files from any location when the structure of the run scripts are followed.

The following examples provide directions for running tests with the ModelSim, Cadence, and Synopsis simulators

Example 1: Modelsim Simulation

To run the simulation in ModelSim,

1. From the command line, open the **vsim** directory.
2. In the command line type:
   `vsim -c -do run_vsim.do ↵`.

The example testbench will run and pass.

The waveform is not generated by default. To enable waveform generation, open the **run_vsim.do** file and search for the text "UNCOMMENT FOLLOWING." If you follow the directions and uncomment the text, the waveforms will be saved in vsim.wlf when you run the simulation.

When you pass the source file list in Modelsim, the directory path remains relative to the original working directory. If you are running your simulation from a location other than the default location, you must change directory to the wrapper location but compile back to working library in original working directory. You can then move back to original working directory to continue compilation and simulation. You can change directories by editing the beginning of the script. The example `run_vsim.do` script does this automatically; the path is set based on the location of the wrappers from the example testbench directory.

Example 2: NCSim Simulation

Complete the following steps to run the simulation in NCSim:

1. From the command line, open the **ncsim** directory.
2. In the command line type:
   `sh run_ncsim.sh` ↵.

The example testbench will run and pass.

The waveform is not generated by default. To enable waveform generation, open your testbench file and search for the text "UNCOMMENT FOLLOWING." If you follow the directions and uncomment the text, you will generate a `verilog.dump` file. To open the file in Simvision, type `run: simvision verilog.dump` in the command window. The **.vcd** dump file is converted to a **.trn** file and the waveforms can be viewed Simvision Waveform Viewer.

Example 3: VCS Simulation

Complete the following steps to run the simulation in VCS:

1. From the command line, open the **vcs** directory.
2. In the command line type:
   `sh run_vcs.sh` ↵.

The example testbench will run and pass.

The waveform is not generated by default. To enable waveform generation, open your testbench file and search for the text "UNCOMMENT FOLLOWING." If you follow the directions and uncomment the text, you will generate a `vcdplus.vpd` file that can be viewed in the Discovery Visual Environment (DVE).

Understanding the 40GbE and 100GbE IP Core Variants

The 40-100GbE IP cores implement virtual lanes as defined in the IEEE 802.3ba-2010 40G and 100G Ethernet Standard. The 40GbE IP cores are fixed at 4 virtual lanes and each lane is sent over a 10 Gbps physical lane. The 100GbE IP cores are fixed at 20 virtual lanes; the 20 virtual lanes are typically bit-interleaved over ten 10-Gbps physical lanes. When the lanes arrive at the receiver the lane streams are in an undefined order. Each lane carries a periodic PCS-VLANE alignment tag to restore the original ordering. The simulation establishes a random permutation of the physical lanes that is used for the remainder of the simulation.

Within each virtual lane stream, the data is 64b/66b encoded. Each word has two framing bits which are always either 01 or 10, never 00 or 11. The RX logic uses this pattern to lock onto the correct word boundaries in each serial stream. The process is probabilistic due to false locks on the pseudo-random scrambled stream. To reduce hardware costs, the receiver does not test alignments in parallel; consequently, the process can be somewhat tedious in simulation.

Both the word lock and the alignment marker lock implement hysteresis as defined in the *IEEE 802.3ba-2010 40G and 100G Ethernet Standard*. Multiple successes are required to acquire lock and multiple failures are required to lose lock. To speed up the alignment marker lock process, simulations typically override the `COUNTER_BITS` parameter to increase the frequency of alignment markers in the data stream. The override results in a small loss of bandwidth in return for reducing startup time by 99%.

The "fully locked" messages in the simulation log indicate the point at which a physical lane has successfully identified the word boundary and virtual lane assignment.

In the event of a catastrophic error, the RX PCS automatically attempts to reacquire alignment. The MAC properly identifies errors in the datastream.

Example Testbench: 100GbE with Adapters

Each of the eight example testbenches has five simulation stages. In this section, the 100GbE IP core with adapters testbench (**alt_e100_avalon_tb.sv**) is used to show successful simulation.

Stage one simulates direct transmission to the TX PCS, writing 32'h1 to 0x102 (MAC_CMD_config). Example 2–1 shows a successful stage one simulation.

Example 2–1. 100GbE with Adapters Testbench Stage 1

```
# Stage 1: Direct to PCS TX
# Wiping stats...
# Starting CGMII traffic...
#   +-------------------------------------------------------------
#   | Simulation Monitor at time             20655000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                    22
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time             25775000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                   260
#   +-------------------------------------------------------------
# Counters Refreshed:
#    fcs_error_cnt:          0
#    runt_cnt:          0
#    rx_err_block_cnt:          0
#    rx_acc_dblk_cnt:                5406
#    rx_acc_start_cnt:                 265
#    rx_dblk_cnt:                5247
#    rx_start_cnt:                 265
#    tx_start_cnt:                 265
# Sent                  265 packets
# Received                  265 packets
# Stage 1: OK
```

Stage two simulates a full datapath with pause insertion. Traffic is sent from a sample ROM. Example 2–2 shows a successful stage two simulation.

Example 2–2. 100GbE with Adapters Testbench Stage 2

```
# Stage 2: Full path with pause insertion
# Wiping stats...
# Starting testbench sample ROM traffic...
# Inserting pause
#   +-------------------------------------------------------------
#   | Simulation Monitor at time             29865000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                     6
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time             34985000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                    33
#   +-------------------------------------------------------------
```

Example 2-2. 100GbE with Adapters Testbench Stage 2 (*Continued*)

```
#   +-----------------------------------------------------------
#   | Simulation Monitor at time             40105000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                   61
#   +-----------------------------------------------------------
#   +-----------------------------------------------------------
#   | Simulation Monitor at time             45225000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                   91
#   +-----------------------------------------------------------
#   +-----------------------------------------------------------
#   | Simulation Monitor at time             50345000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  121
#   +-----------------------------------------------------------
#   +-----------------------------------------------------------
#   | Simulation Monitor at time             55465000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  148
#   +-----------------------------------------------------------
#   +-----------------------------------------------------------
#   | Simulation Monitor at time             60585000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  176
#   +-----------------------------------------------------------
#   +-----------------------------------------------------------
#   | Simulation Monitor at time             65705000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  206
#   +-----------------------------------------------------------
#   +-----------------------------------------------------------
#   | Simulation Monitor at time             70825000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  236
#   +-----------------------------------------------------------
#   +-----------------------------------------------------------
#   | Simulation Monitor at time             75945000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  266
#   +-----------------------------------------------------------
```

Example 2-2. 100GbE with Adapters Testbench Stage 2 (*Continued*)

```
# Counters Refreshed:
#    fcs_error_cnt:          0
#    runt_cnt:          0
#    rx_err_block_cnt:          0
#    rx_acc_dblk_cnt:               16491
#    rx_acc_start_cnt:                  266
#    rx_dblk_cnt:               16438
#    rx_start_cnt:                  266
#    tx_start_cnt:                  266
# Sent                  266 packets
# Received                  266 packets
# Stage 2: OK
```

Stage three simulates a full datapath with bit errors. Traffic is sent from a sample ROM and error bits are injected at serial loopback. Example 2–3 shows a successful stage three simulation.

Example 2–3. 100GbE with Adapters Testbench Stage 3

```
# Stage 3: Full path with bit errors
# Wiping stats...
# Enabling errors...
# Starting testbench sample ROM traffic...
#   +-------------------------------------------------------------
#   | Simulation Monitor at time             78465000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                    0
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time             83585000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:02020 bip_error: f3fff
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                   21
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time             88705000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00082 bip_error: f3fff
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                   50
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time             93825000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:22800 bip_error: fffcf
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                   80
#   +-------------------------------------------------------------
```

Example 2-3. 100GbE with Adapters Testbench Stage 3 (*Continued*)

```
#   +-------------------------------------------------------------
#   | Simulation Monitor at time             98945000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:80020 bip_error: fffff
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  110
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            104065000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:a2000 bip_error: fffff
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  140
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            109185000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:82020 bip_error: cefef
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  168
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            114305000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:02008 bip_error: fffff
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  195
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            119425000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:000a0 bip_error: ffcff
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  225
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            124545000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:02002 bip_error: fffff
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  255
#   +-------------------------------------------------------------
# Disabling errors...
# Counters Refreshed:
#    fcs_error_cnt:        117
#    runt_cnt:          0
#    rx_err_block_cnt:       1180
#    rx_acc_dblk_cnt:                 8391
#    rx_acc_start_cnt:                  153
#    rx_dblk_cnt:               16749
#    rx_start_cnt:                 261
#    tx_start_cnt:                 270
# Sent                  270 packets
# Received                  261 packets
# Received       1180 err blocks
# Received        117 FCS errors
# Lanes with BIP errors  : fffff
# Lanes with framing errors  : a28aa
#                  153 packets were accepted by the RX
# Stage 3: OK
```

Stage four simulates the return to normal traffic. Traffic is sent from the packet generator and received packets are checked by the packet checker. Example 2–4 shows a successful stage four simulation.

Example 2–4. 100GbE with Adapters Testbench Stage 4

```
# Stage 4: Return to normal traffic - switch to packet generator
# OK.  Wiping stats...
# Starting packet generator traffic...
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            129175000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                    0
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            134295000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                   77
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            139415000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  195
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            144535000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  316
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            149655000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  442
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            154775000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  546
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            159895000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  667
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
```

Example 2-4. 100GbE with Adapters Testbench Stage 4 (*Continued*)

```
#   | Simulation Monitor at time            165015000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  786
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            170135000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  900
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            175255000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                 1026
#   +-------------------------------------------------------------
# Counters Refreshed:
#    fcs_error_cnt:          0
#    runt_cnt:          0
#    rx_err_block_cnt:          0
#    rx_acc_dblk_cnt:               28847
#    rx_acc_start_cnt:                1068
#    rx_dblk_cnt:               28549
#    rx_start_cnt:                1068
#    tx_start_cnt:                1068
# Sent                 1068 packets
# Received                 1068 packets
# Stage 4: OK
```

Stage five simulates full datapath with asynchronous reset. Traffic is sent from a sample ROM and asynchronous resets are asserted during traffic flow. In order to check whether the TX and RX counter values match, traffic is paused after asserting the resets and is restarted after the IP core recovers and the counters are cleared. This prevents on-flight traffic from existing in the IP core during resets which can cause TX and RX counter mismatch. Example 2–5 shows a successful stage five simulation.

Example 2–5. 100GbE with Adapters Testbench Stage 5

```
# Stage 5: Full path with MAC, PCS and PMA arst
# Wiping stats...
# Starting testbench sample ROM traffic...
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            179395000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                    0
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
```

Example 2-5. 100GbE with Adapters Testbench Stage 5 (*Continued*)

```
#   | Simulation Monitor at time            184515000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  25
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            189635000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  50
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            194755000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  80
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            199875000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                 110
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            204995000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                 140
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            210115000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                 167
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            215235000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                 195
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            220355000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                 225
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
```

Example 2-5. 100GbE with Adapters Testbench Stage 5 (*Continued*)

```
#   | Simulation Monitor at time            225475000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                 255
#   +-----------------------------------------------------------
# Reseting MAC, PCS and PMA through AVALON_MM interface...
#   +-----------------------------------------------
#   |  Expected loss of lock
#   +-----------------------------------------------
# TX interface is ready at time            226680000
#   +-----------------------------------------------
#   |  Waitig for all lanes locked and deskewed...
#   +-----------------------------------------------
# RX interface is ready at time            226940000
# Lane          17 fully locked at time            229830000
#   +-----------------------------------------------------------
#   | Simulation Monitor at time            230595000 :
#   |    am_locked: 20000 all_locked: 0 lanes_deskewed: 0
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                 0
#   +-----------------------------------------------------------
# Lane          14 fully locked at time            230610000
# Lane           0 fully locked at time            230610000
# Lane          12 fully locked at time            231390000
# Lane          11 fully locked at time            231390000
# Lane           2 fully locked at time            231390000
# Lane           1 fully locked at time            231390000
# Lane           9 fully locked at time            233210000
# Lane          10 fully locked at time            235550000
# Lane           7 fully locked at time            235550000
#   +-----------------------------------------------------------
#   | Simulation Monitor at time            235715000 :
#   |    am_locked: 25e87 all_locked: 0 lanes_deskewed: 0
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                 0
#   +-----------------------------------------------------------
# Lane          13 fully locked at time            236330000
# Lane           6 fully locked at time            236330000
# Lane           4 fully locked at time            237110000
# Lane           3 fully locked at time            237110000
# Lane          19 fully locked at time            238150000
# Lane          18 fully locked at time            238150000
# Lane          15 fully locked at time            238150000
#   +-----------------------------------------------------------
#   | Simulation Monitor at time            240835000 :
#   |    am_locked: efedf all_locked: 0 lanes_deskewed: 0
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                 0
#   +-----------------------------------------------------------
# Lane          16 fully locked at time            241270000
# Lane           8 fully locked at time            242050000
#   +-----------------------------------------------------------
```

Example 2-5. 100GbE with Adapters Testbench Stage 5 (*Continued*)

```
#   | Simulation Monitor at time            245955000 :
#   |    am_locked: fffdf all_locked: 0 lanes_deskewed: 0
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                    0
#   +-------------------------------------------------------------
# All lanes locked.  Starting deskew at time            246889110
# Lane           5 fully locked at time            246990000
# All lanes locked at time            247020000
# Deskew complete at time            248585060
# Lanes deskewed at time            248840000
# Wiping stats...
# Restarting Traffic...
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            249775000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                    5
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            254895000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                   30
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            260015000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                   60
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            265135000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                   90
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            270255000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  120
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            275375000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  148
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
```

Example 2-5. 100GbE with Adapters Testbench Stage 5 (*Continued*)

```
#   | Simulation Monitor at time            280495000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  175
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            285615000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  205
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            290735000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  235
#   +-------------------------------------------------------------
#   +-------------------------------------------------------------
#   | Simulation Monitor at time            295855000 :
#   |    am_locked: fffff all_locked: 1 lanes_deskewed: 1
#   |    framing_error:00000 bip_error: 00000
#   |    deskew_failure: 0 pma_rx_ready:1  pma_tx_ready:1
#   |    Number of packets sent:                  265
#   +-------------------------------------------------------------
# Counters Refreshed:
#    fcs_error_cnt:          0
#    runt_cnt:          0
#    rx_err_block_cnt:          0
#    rx_acc_dblk_cnt:               16794
#    rx_acc_start_cnt:                  270
#    rx_dblk_cnt:                16740
#    rx_start_cnt:                  270
#    tx_start_cnt:                  270
# Sent                   270 packets
# Received                   270 packets
# Stage 5: OK
```

3. Functional Description

This chapter provides a detailed description of the 40-100GbE IP cores. It begins with a high-level overview of typical Ethernet systems and then provides detailed descriptions of the MAC, transmit (TX) and receive (RX) datapaths, register descriptions, and an Ethernet glossary. It includes the following sections:

- High Level System Overview
- MAC and PHY Functional Description
- Software Register Interface
- Ethernet Glossary

High Level System Overview

Figure 3–1 illustrates the 40GbE and 100GbE MAC and PHY MegaCore functions.

Figure 3–1. 40GbE and 100GbE MAC and PHY MegaCore Functions

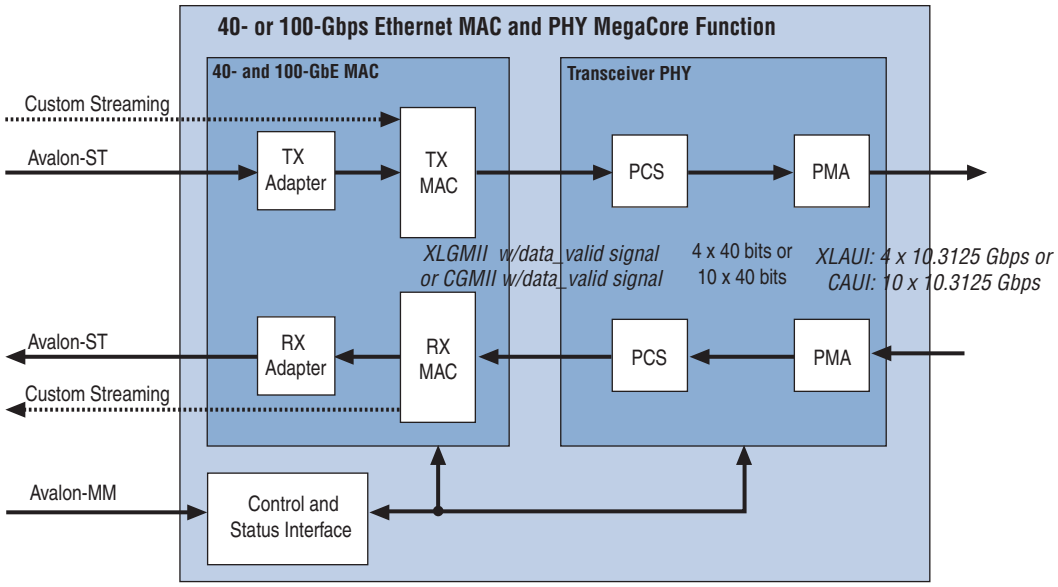
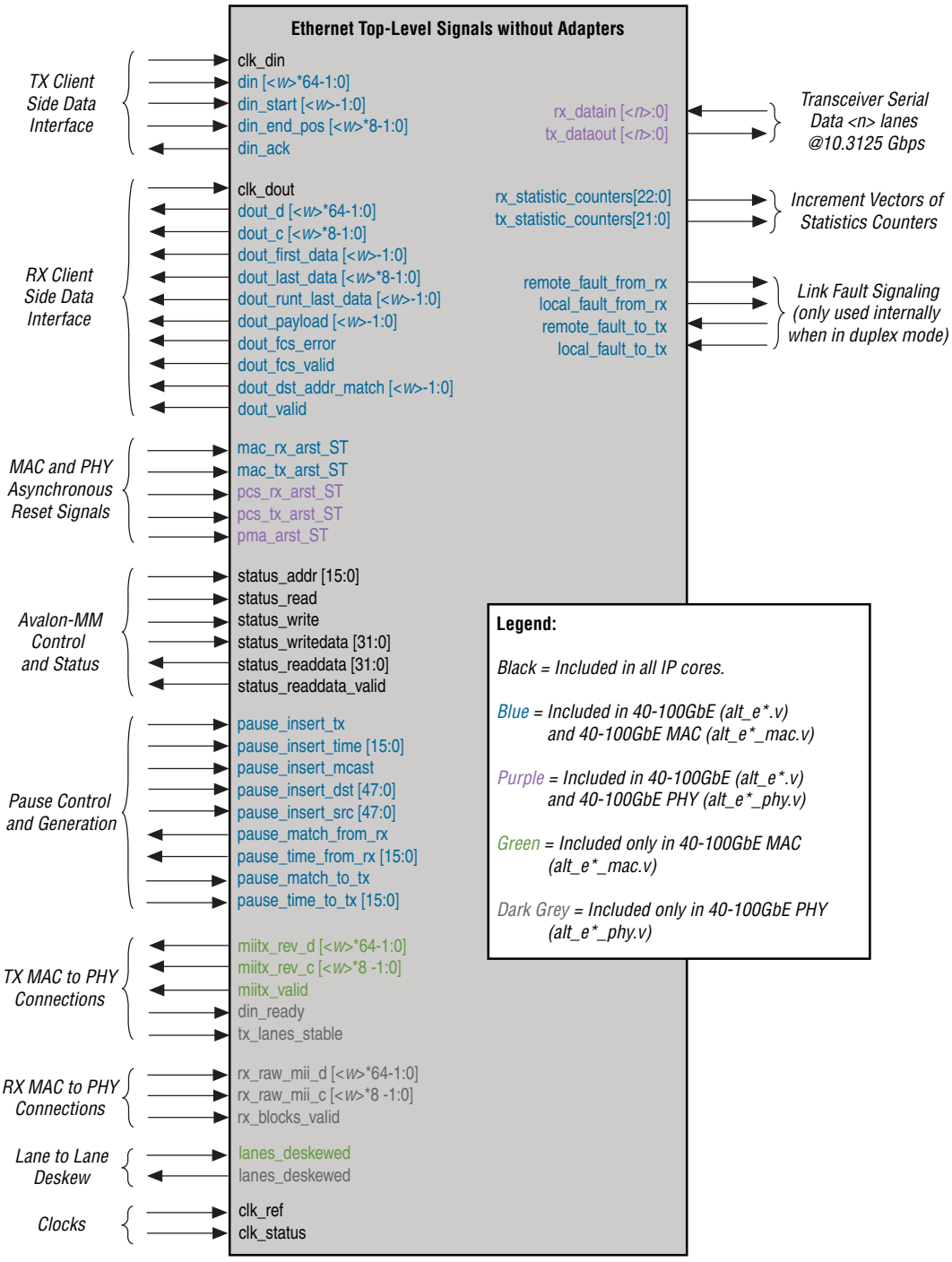


Figure 3–2 shows the top-level signals of the 40-100GbE IP core without adapters.

Figure 3–2. Top-Level Signal of the 40-100GbE IP Core without Adapters

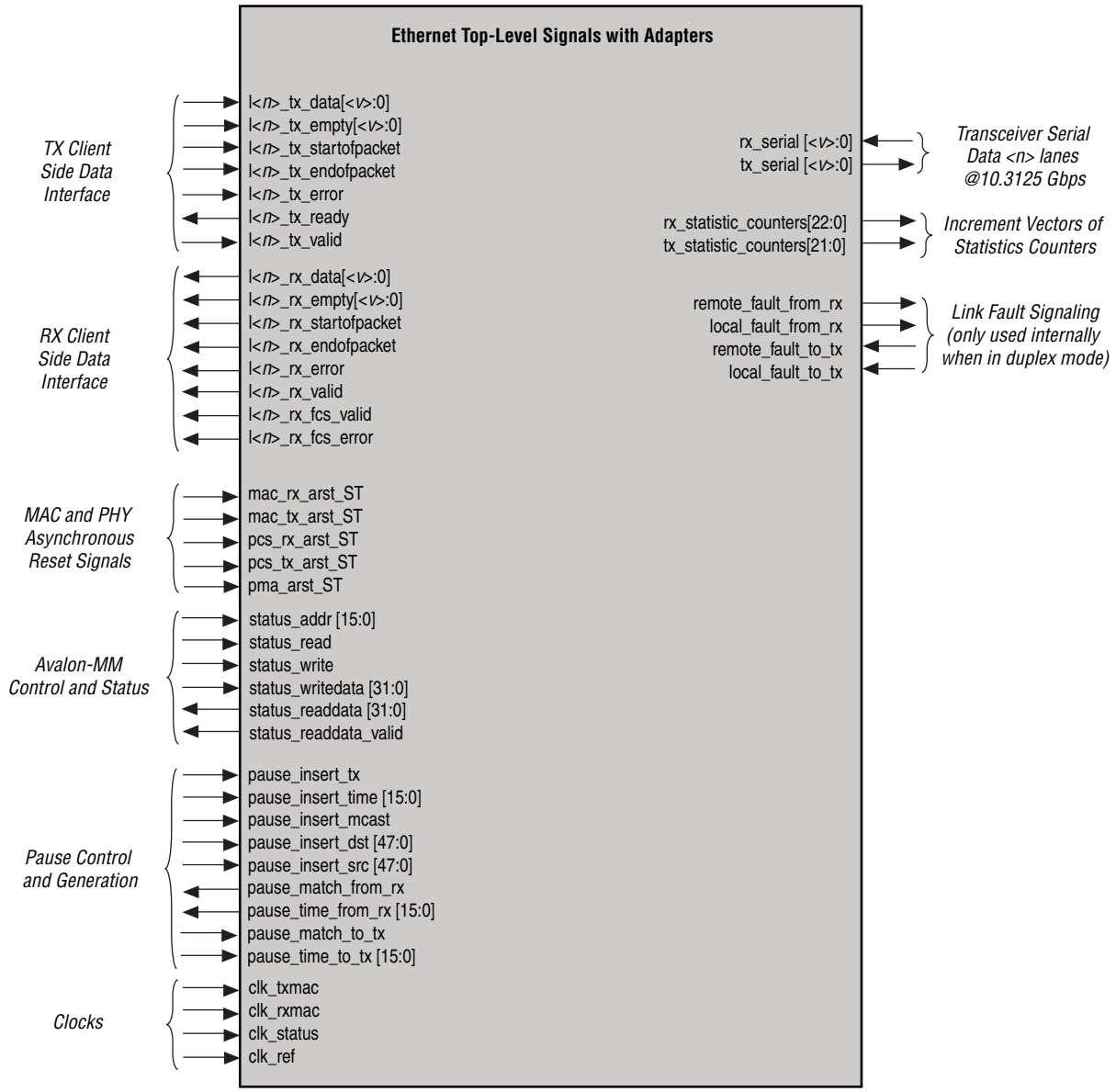


Note to Figure 3–2:

(1) For more information on the increment vectors of statistics counters, refer to "Statistics Registers" on page 3–50.

Figure 3–3 shows the top-level signals of the 40-100GbE IP core with adapters.

Figure 3–3. Top-Level Signal of the 40-100GbE IP Core with Adapters



Note to Figure 3–3:

(1) For more information on the increment vectors of statistics counters, refer to "Statistics Registers" on page 3–50.

As these figures illustrate, the Ethernet flow control pause signals are not available in the top-level pinout for the 100GbE IP core with adapters; however, you can implement the pause functionality using the pause registers. For more information, refer to "Pause Registers" on page 3–47.

MAC and PHY Functional Description

The Altera 40-100GbE IP core implements the 40-100GbE Ethernet MAC in accordance with the *IEEE 803.3ba 2010 Higher Speed Ethernet Standard*. This module handles the frame encapsulation and flow of data between a client logic and Ethernet network via a 40-100GbE Ethernet PCS and PMA (PHY). In the TX direction, the MAC accepts client frames, inserts inter-packet gap (IPG), preamble, start of frame delimiter (SFD), header, padding, and checksum before passing them to the PHY. The PHY encodes the MAC frame as required for reliable transmission over the media to the remote end. Similarly, in the receive direction, the MAC accepts frames from the PHY, performs checks, updates statistics counters, strips out the CRC, preamble, and SFD, and passes the rest of the frame to the client.

The MAC includes the following interfaces:

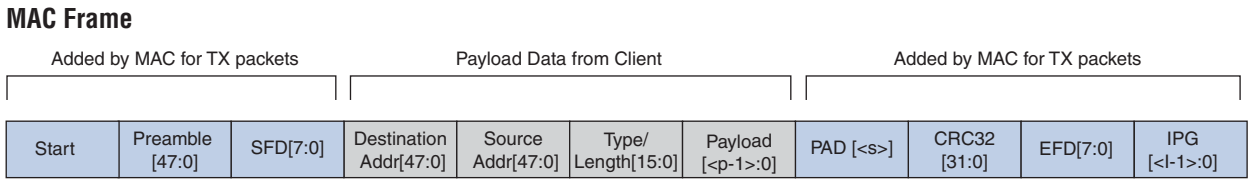
- Datapath interface, client side. The following options are available:
  - 40GbE with adapters—Avalon-ST, 256 bits
  - 40GbE—Streaming, 128 bits
  - 100GbE with adapters—Avalon-ST, 512 bits
  - 100GbE—Streaming, 320 bits
- Datapath PHY side–The following options are available:
  - 40GbE—XLAUI
  - 100GbE—CAUI
- Management interface
- Avalon-MM host slave interface for MAC management

TX Datapath

The TX MAC module receives the client payload data with the destination and source addresses and then adds, appends, or updates various header fields in accordance with the configuration specified. The MAC does not modify the destination address or the payload received from client. However, the TX MAC module adds a preamble, pads the payload to satisfy the minimum Ethernet frame payload of 46 bytes, and calculates the CRC over the entire MAC frame. (If padding is added, it is also included in CRC calculation.) The TX MAC module can also modify the source address, and always inserts IDLE bytes to maintain an average inter-packet gap (IPG).

Figure 3–4 illustrates the changes that the TX MAC makes to the client frame.

Figure 3–4. Typical Client Frame at the Transmit Interface



Notes to Figure 3–4:

(1) <p> = payload size = 0–1500 bytes, or 9600 bytes for jumbo frames.

(2) <s> = padding bytes = 0–46 bytes

(3) <I> = number of IPG bytes

The following sections describe the functions that the TX module performs.

Preamble, Start, and SFD Insertion

In the TX datapath the MAC appends a one-byte START, 6-byte preamble, and 1-byte SFD to the client frame. (This MAC module also incorporates the functions of the reconciliation sublayer.)

Address Insertion

The client provides the destination MAC address and the source address of the local MAC. However, if enabled by a `MADDR_CTRL` register (0xc2, bit 31), the source MAC address can be replaced by the source address contained in two, 32-bit MAC registers: `SRC_AD_LO` and `SRC_AD_HI`.

Length/Type Field Processing

This two-byte header represents either the length of the payload or the type of MAC frame. When the value of this field is equal to or greater than 1536 (0x600) it indicates a type field. Otherwise, this field provides the length of the payload data that ranges from 0–1500 bytes. The TX MAC does not modify this field and forwards it to the network.

Frame Padding

When the length of client frame is less than 64 bytes (meaning the payload is less than 46 bytes), the TX MAC module inserts pad bytes (0x00) after the payload to create a frame length equal to the minimum size of 64 bytes.

Frame Check Sequence (CRC-32) Insertion

The TX MAC computes and inserts CRC32 checksum in the transmitted MAC frame. The frame check sequence (FCS) field contains a 32-bit CRC value. The MAC computes the CRC32 over the frame bytes that include the source address, destination address, length, data, and pad. The CRC checksum computation excludes the preamble, SFD, and FCS. The encoding is defined by the following generating polynomial:

$$FCS(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X^{1} + 1$$

CRC bits are transmitted with MSB (X^{32}) first. (Refer to "Byte Order on the Avalon-ST Interface Lanes" on page 3–12 and "Octet Transmission on the Avalon-ST SIgnals" on page 3–13 for illustrations of byte ordering.

Independent user configuration register bits control FCS CRC insertion at runtime. The `tx_crc_config` register enables and disables CRC insertion; by default, the CRC insertion register is enabled.

Inter-packet Gap Generation and Insertion

The TX MAC maintains the minimum inter-packet gap (IPG) between transmitted frames required by the IEEE 802.3 Ethernet standard. The average IPG is maintained at 96 bit times (or 12 byte times). The deficit idle counter maintains an average IPG of 12 bytes.

TX Data Bus Interfaces

40-100GbE IP core provides two different user interfaces: the Avalon-ST interface and a custom interface. The Avalon-ST interface requires adapters and the custom interface does not require adapters.

This section describes the TX data bus at the user interface and includes the following topics:

- TX Data Bus with Adapters (Avalon-ST Interface)
- TX Data Bus without Adapters (Custom Streaming Interface)

TX Data Bus with Adapters (Avalon-ST Interface)

The TX datapath employs the Avalon-ST protocol. The Avalon-ST protocol is a synchronous point-to-point, unidirectional interface that connects the producer of a data stream (source) to a consumer of data (sink). The key properties of this interface include:

- Frame transfers marked by start of packet and end of packet signals.
- Signals from the source to sink are qualified by a valid signal.
- Errors marking a current packet, are aligned with the end of packet cycle.
- Use of a ready signal by the sink backpressures the source. The source typically responds to the deassertion of the ready signal from sink by driving the same data until the sink can accept it. The `readyLatency` defines the relationship between assertion and deassertion of the ready signal, and cycles which are considered to be `ready` for data transfer.The `readyLatency` on both the TX and RX interfaces is one cycle.

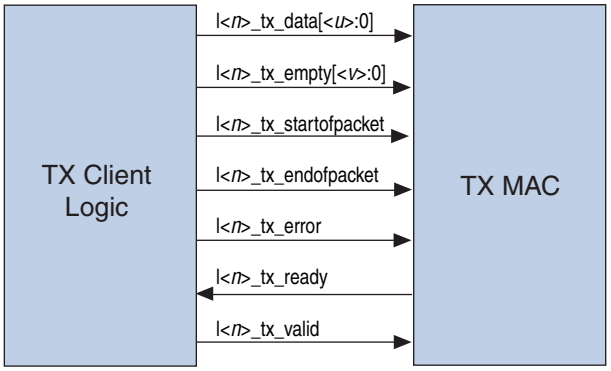
Altera provides an Avalon-ST interface with adapters for both the 40GbE and 100GbE IP cores. The Avalon-ST interface requires that the start of packet (SOP) is always in the MSB, simplifying the interpretation and processing of incoming data. The TX adapter for the 100GbE IP core increases the client interface Avalon-ST bus width from 5 words (320 bits) to 8 words (512 bits). The TX adapter for the 40GbE IP core increases the client interface Avalon-ST bus width from 2 words (128 bits) to 4 words (256 bits). In both cases the client interfaces operate at a frequency above 312.5 MHz, nominally at 315 MHz.

The client acts as a source and the TX MAC acts as a sink in the transmit direction.

For more information about the Avalon-ST interface refer to the *Avalon Interface Specifications.*

Figure 3–5 illustrates the TX Avalon-ST interface with adapters.

Figure 3–5. TX Data Bus with Adapters (1)



Note to Figure 3–5:

(1) <n> = 4 for the 40GbE IP core and <n> = 8 for the 100GbE IP core. Lane variables are denoted by <u> and <v>.

Table 3–1 defines the signals in the TX interface with adapters.

Table 3–1. TX Data Interface with Adapters (1)

Signal Name	Direction	Description
l<n>_tx_data[<u>:0]	Input	TX data.
l<n>_tx_empty[<v>:0]	Input	Indicates the number of empty bytes in a packet if l<n>_tx_endofpacket is asserted.
l<n>_tx_startofpacket	Input	When asserted, indicates the start of a packet. The packet starts on the MSB.
l<n>_tx_endofpacket	Input	When asserted, indicates the end of packet.
l<n>_tx_error	Input	This signal is not used in the current implementation.
l<n>_tx_ready	Output	When asserted, the MAC is ready to receive data. The l<n>_tx_ready signal acts as an acknowledge. The source drives l<n>_tx_valid and l<n>_tx_data[<n>64-1:0] then waits for the sink to assert l<n>_tx_ready. The readyLatency is one cycle so that valid data is accepted one cycle after l<n>_tx_ready is asserted.
l<n>_tx_valid	Input	When asserted l<n>_tx_data is valid. Only valid between the l<n>_tx_startofpacket and l<n>_tx_endofpacket signals.

Note to Table 3–1:

(1) <n> = 4 for the 4-to-2 adapter. and <n> = 8 for the 8-to-5 adapter. Lane variables are denoted by <u> and <v>.

Figure 3–6 shows typical traffic for the TX and RX Avalon-ST interface 40GbE IP core. This example was taken from a ModelSim simulation of the parallel testbench described in "Testbench with Adapters" on page 2–8.

Figure 3–6. Traffic on the TX and RX Client Interface for 40GbE IP Core Using the Four- to Two-Word Adapters

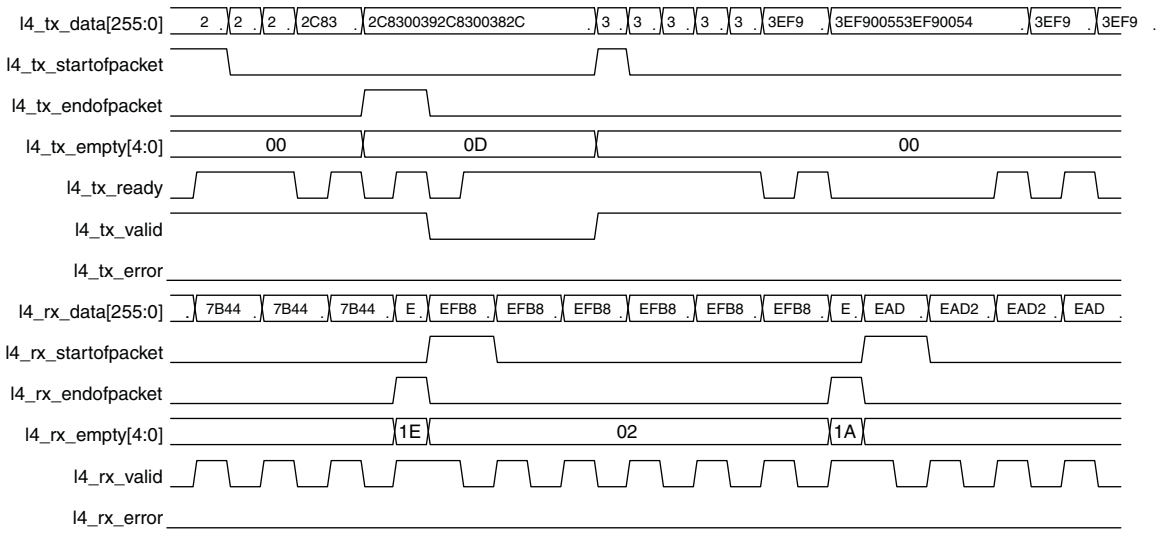
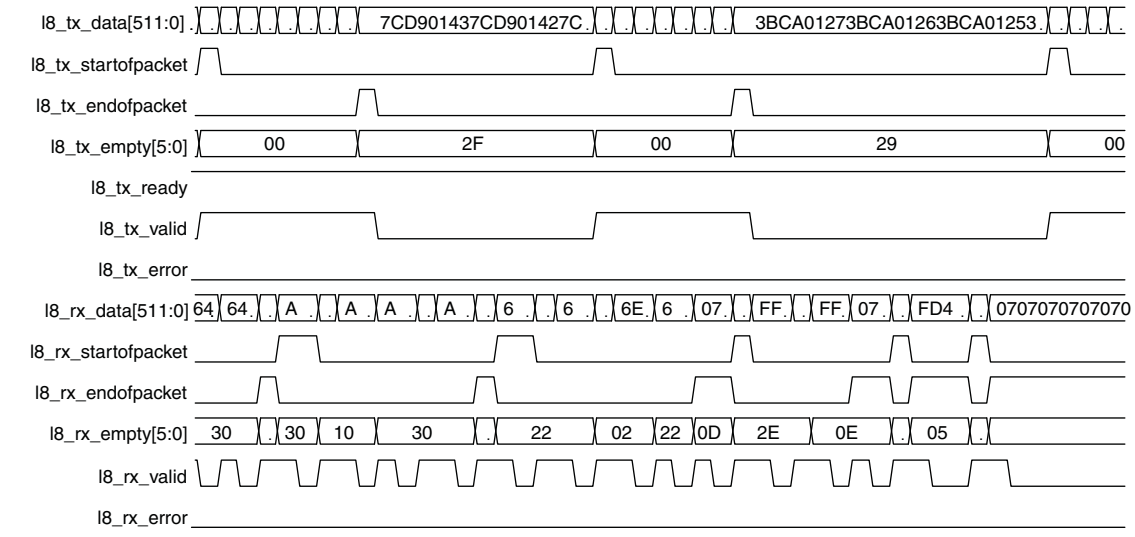


Figure 3–7 shows typical traffic for the TX and RX Avalon-ST interface of the 100GbE IP core. This example was taken from a ModelSim simulation of the parallel testbench described in "Testbench with Adapters" on page 2–8.

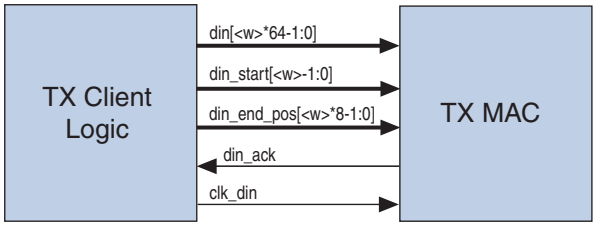
Figure 3–7. Traffic on the TX and RX Client Interface for 100GbE IP Core Using the Eight- to Five-Word Adapters



TX Data Bus without Adapters (Custom Streaming Interface)

When no adapters are used, the 40GbE custom interface bus width is 2 words (128 bits) and the 100GbE custom interface bus width is 5 words (320 bits). In both cases the client interfaces operate at a frequency above 312.5 MHz, nominally at 315 MHz. Figure 3–8 illustrates this configuration.

Figure 3–8. TX Data Bus (1)



Note to Figure 3–8:

(1) <*w*> = 2 for the 40GbE IP core and <*w*> = 5 for the 100GbE IP core.

Table 3–2 defines the signals in the TX interface without adapters.

Table 3–2. TX Data Bus (1)

Signal Name	Direction	Description
`din[<w>*64-1:0]`	Input	Data bytes to send in big endian mode.
`din_start[<w>-1:0]`	Input	Start of packet (SOP) location in the TX data bus. Only the most significant byte of each 64-bit word may be a start of packet. Bit 63 or 127 are possible for the 40GbE and bits 319, 255, 191, 127, or 63 are possible for 100 GbE.
`din_end_pos[<w>*8-1:0]`	Input	End of packet. Any byte may be the last byte in a packet.
`din_ack`	Output	Indicates that input data was accepted by the IP core.
`clk_din`	Input	TX MAC clock. This clock must exceed 312.5 MHz for the circuit to function correctly. Altera recommends 315 MHz. The `clk_din` and `clk_dout` which clocks the RX datapath are not related and their rates do not have to match.

Note to Table 3–2:

(1) <*w*> = 2 for the 40GbE IP core and <*w*> = 5 for the 100GbE IP core.

The bytes are read in the standard left to right order. A packet may start in the most significant byte of any word. A packet may end on any byte.They are reversed for transmission per Ethernet requirements. The transmitter handles the insertion of the inter-packet gap, frame delimiters, and padding with zeros as necessary. The transmitter also handles FCS computation and insertion.

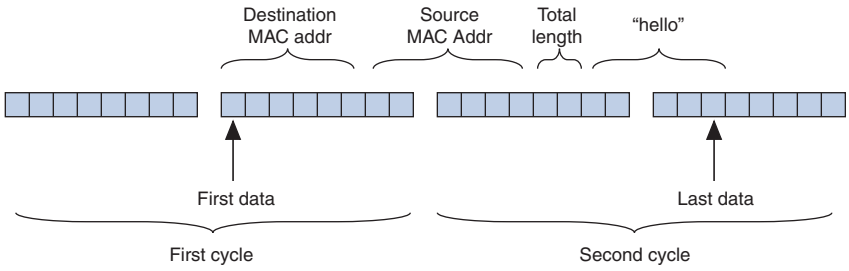
The Ethernet MAC and PHY transmit complete packets. Once transmission begins, it must complete with no idle insertions. Between the end of one packet and the beginning of the next packet, the data input is not considered and the transmitter sends idle characters. There is no limit on the number of idle characters that can be sent between packets.

The following sections provide examples for the 40GbE and 100GbE IP cores.

40GbE IP Core without Adapters

Figure 3–9 illustrates the transmission of a short packet.

Figure 3–9. Short Packet Example



Example 3–1 shows the Verilog HDL code that represents the simple packet illustrated in Figure 3–9. Note that bit `din_end[5]`, in the second cycle, corresponding to the "Last data" in Figure 3–9 is asserted.

Example 3–1. Bus Representation of a Short TX Packet

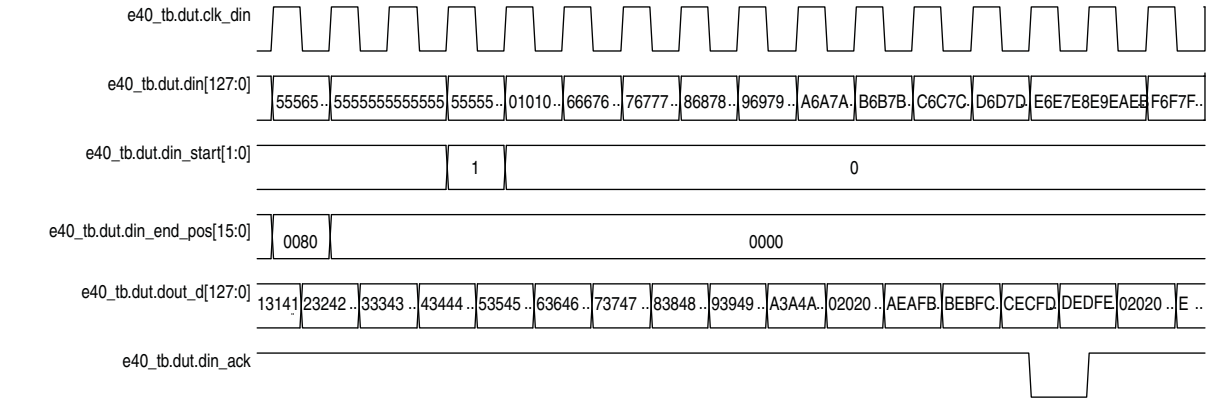
```
wire [6*8-1:0] dst_addr = 48'hffff ffff ffff;
wire [6*8-1:0] src_addr = 48'h0007 edff 1234;
wire [2*8-1:0] len = 16'd64; =

First cycle:
din = {64'h0, dst_addr, src_addr[47:32]};
din_start = 2'b01;
din_end = 16'b00000000_00000000;

Second cycle:
din = {src_addr[31:0], len, "hello", 40'h0};
din_start = 2'b00;
din_end = 16'b00000000_00100000;
```

Figure 3–10 shows illustrates the deassertion of the `din_ack` signal. The data beginning with 0xe6e7 is not immediately accepted. The `din` bus must be held until `din_ack` returns to one. At this point normal data flow resumes.

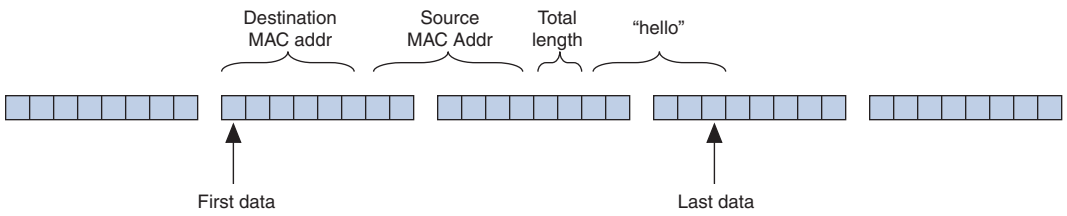
Figure 3–10. Sample TX Bus Activity



100GbE IP Core without Adapters

Figure 3–11 illustrates the transmission of a short packet for the 100GbE IP core.

Figure 3–11. Short Packet Example



Example 3–2 shows the Verilog HDL code that represents the simple packet illustrated in Figure 3–11. Note that bit `din_end[13]`, corresponding to the "Last data" in Figure 3–11 is asserted.

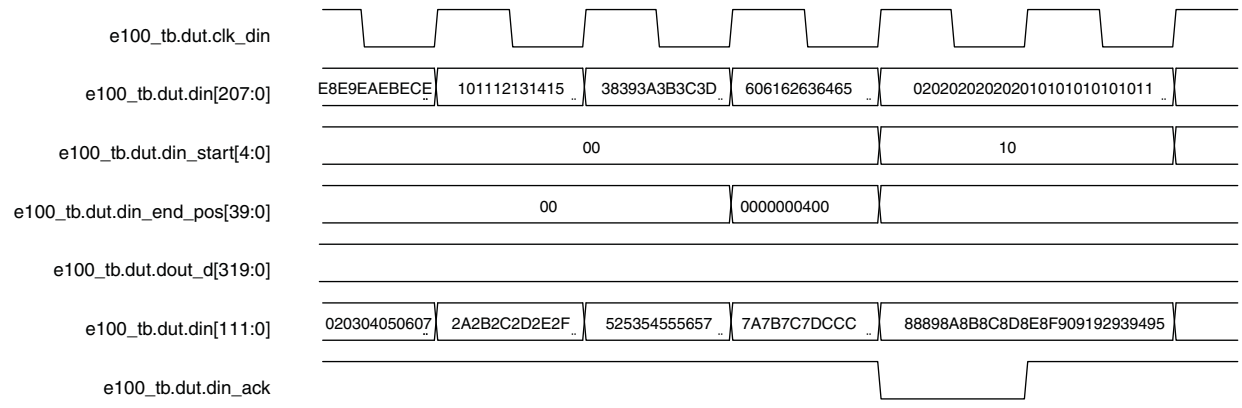
Example 3–2. Bus Representation of a Short TX Packet

```
wire [6*8-1:0] dst_addr = 48'hffff ffff ffff;
wire [6*8-1:0] src_addr = 48'h0007 edff 1234;
wire [2*8-1:0] len = 16'd64; =

din = {64'h0, dst_addr, src_addr, len, "hello", 40'h0, 64'h0};
din_start = 5'b01000;
din_end = 40'b00000000_00000000_00000000_00100000_00000000;
```

Figure 3–12 illustrates the deassertion of the `din_ack` signal. The data beginning with 0x0202 is not immediately accepted. The `din` bus must be held until `din_ack` returns to one. At this point normal data flow resumes.

Figure 3–12. Sample TX Bus Activity

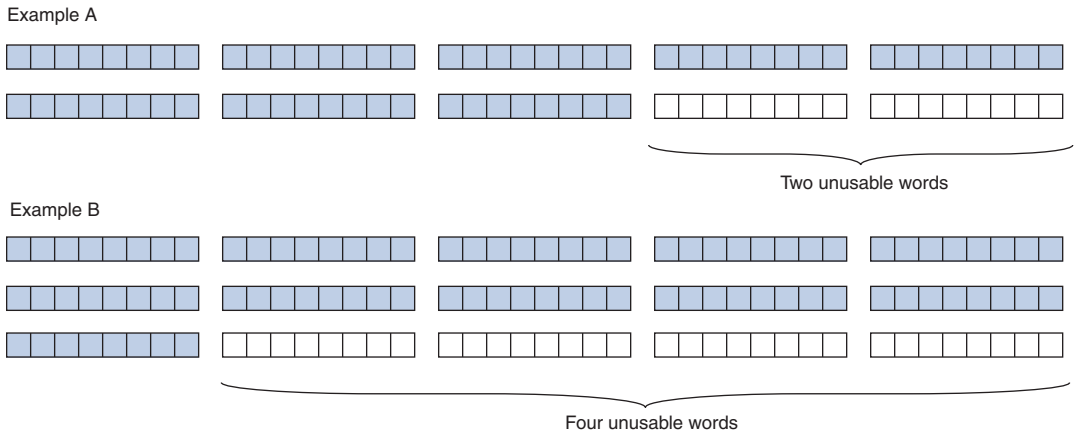


The TX logic supports packets of less than the usual length. For example, `din_start` might be set to 5'b11111, indicating the start of a new packet in 5, successive words. In this case, `din_end_pos` equals 40'h0101010101, indicating 5 packets of 8 bytes. Each 8-byte packet is padded with 0's to create 64-byte packet.

Bus Quantization Effects

The TX custom streaming interface uses two or five words (40 bytes). The TX bus allows a packet to start at any of two or five positions to maximize utilization of the link bandwidth. If the start of packet (SOP) must be restricted to the most significant position in the client logic data bus, bus bandwidth is reduced. Figure 3–13 illustrates the reduction of bandwidth caused by left-aligning the SOP for the 100GbE IP core.

Figure 3–13. Reduced Bandwidth due to left aligned SOP



In Figure 3–13, Example A shows the minimum-sized packet of eight words. Example B shows an 11-word packet which is the worst-case for bandwidth utilization. Assuming another packet is waiting for transmission, the effective ingress bandwidth is reduced by 20% and 26%, respectively. Running the MAC portion of the logic slightly faster than is required can mitigate this loss of bandwidth. Additional increases in the MAC frequency can provide further mitigation, although doing so makes timing closure more difficult.

Order of Transmission

Bytes are transmitted starting with the preamble and ending with the FCS in accordance with the IEEE 802.3 standard. Transmit frames received from the client on the Avalon-ST interface are big endian. Frames transmitted on the XGMII/CGMII are little endian; the MAC TX transmits frames on this interface beginning with the lease significant byte. Table 3–3 describes the byte order on the Avalon-ST interface.

Table 3–3. Byte Order on the Avalon-ST Interface Lanes

	Destination Address (DA) (1)						Source Address (SA)						Type/ Length		Data (D)		
Octet	**5**	**4**	**3**	**2**	**1**	**0**	**5**	**4**	**3**	**2**	**1**	**0**	**1**	**0**	**00**	**...**	**NN**
Bit	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]	[15:8]	[7:0]	MSB[7:0]	...	LSB[7:0]

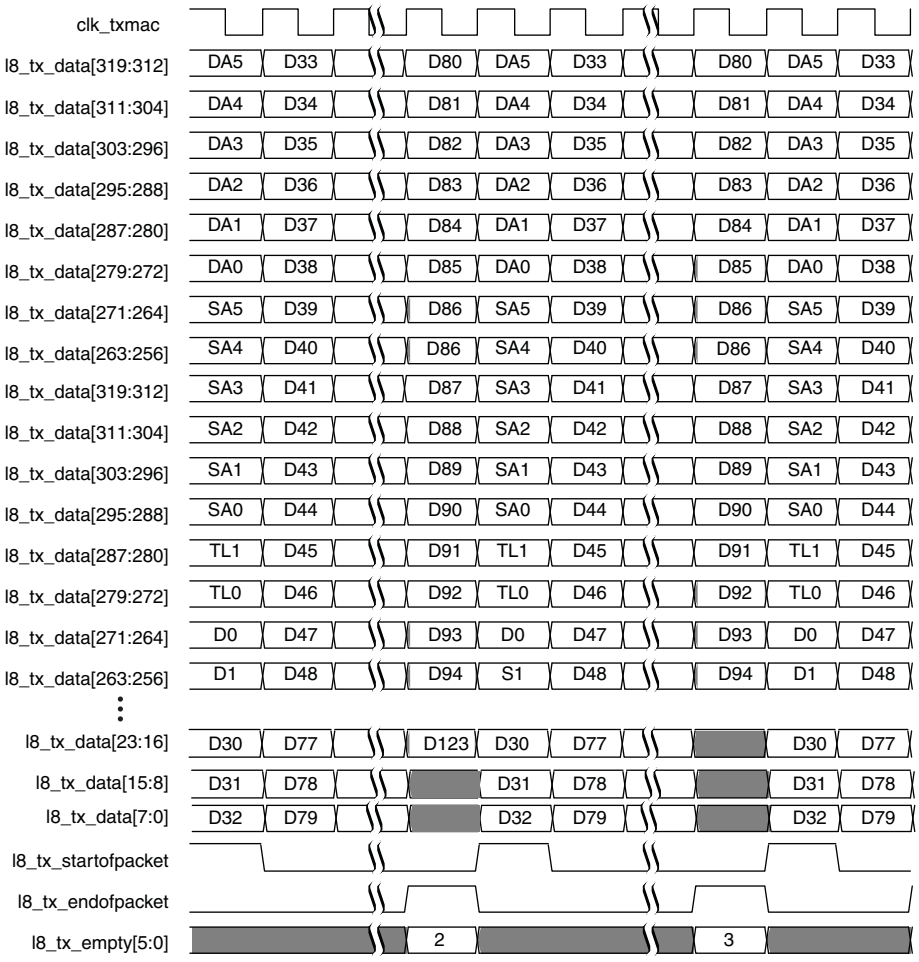
Note to Table 3–3:

(1) Destination Address[40] = type bit, broadcast/multicast bit.

(2) Destination Address[41] = Locally administrated address bit

For example, the destination MAC address includes the following six octets AC-DE-48-00-00-80, the first octet transmitted (octet 0 of the MAC address described in 802.3) is AC and the last octet transmitted (octet 7 of the MAC address) is 80. The first bit transmitted is the low-order bit of AC, a zero. The last bit transmitted is the high order bit of 80, a one. Figure 3–14 further illustrates how the octets of the client frame are transferred over the TX datapath. As Table 3–3 and Figure 3–14 illustrate, 0xAC is driven on `DA5[47:40]` and 0x80 is driven on `DA0[7:0]`.

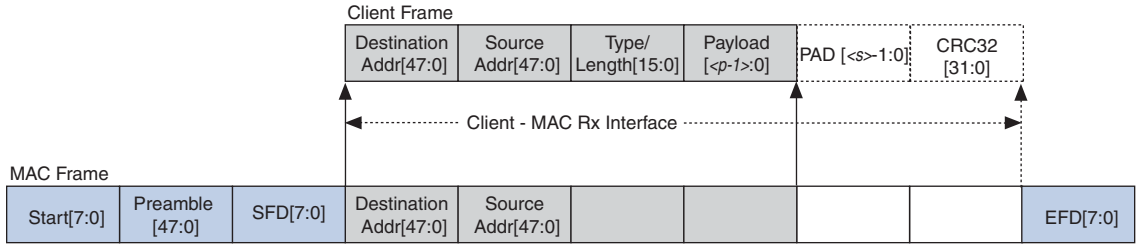
Figure 3–14. Octet Transmission on the Avalon-ST SIgnals



RX Datapath

The RX MAC receives Ethernet frames from the PHY and forwards the payload with relevant header bytes to the client after performing some MAC functions on header bytes. Figure 3–15 illustrates the typical flow of frame through the MAC RX.

Figure 3–15. Flow of Frame through the MAC RX



Notes to Figure 3–15:

(1) <p> = payload size = 0–1500 bytes.

(2) <s> = pad bytes = 0–46 bytes.

Oversized-frame handling can be enabled in store and forward mode. When the maximum frame size is set to 9600 bytes, the core passes some of the frames between 9601-9644 bytes in size. Frames of 9645 bytes or more are dropped. For the 100GbE IP core, if the frame size is within 44 bytes over the specified maximum frame size, it may or may not be dropped, but oversized frames of over 44 bytes will always be dropped. For the 40GbE IP core, if the frame size is within 20 bytes over the specified maximum frame size, it may or may not be dropped, but oversized frames of over 20 bytes will always be dropped.

The following sections describe the functions performed by the RX MAC.

Preamble Processing

The preamble sequence is Start, six preamble bytes, and SFD. If this sequence is incorrect the frame is ignored. The start word must be on receive lane 0 (most significant byte). The IP core uses the SFD byte (0xD5) to identify the last byte of the preamble. The MAC RX looks for the Start, six preamble bytes and SFD. The MAC RX removes all Start, SFD and IPG bytes from accepted frames.

FCS (CRC-32) Removal

Independent user configuration register bits control FCS CRC removal at runtime. CRC removal supports both narrow and wide bus options. The `rx_crc_config` register enables and disables CRC removal; by default, the CRC removal register is enabled.

In the user interface, the `eop` signal indicates the end of CRC data if CRC is not removed. When CRC is removed, the `eop` signal indicates the last byte of payload. The `fcs_error` signal and the `eop` signal are asserted on the same clock cycle when there is an FCS error in the current frame.

CRC Checking

The 32-bit CRC field is received in the order: X^{32}, X^{30}, . . . X^{1}, and X^{0} , where X^{32} is the MSB of FCS field and occupies the LSB position on first FCS byte field. If a CRC32 error is detected, the MAC RX marks the frame invalid by asserting the `dout_fcs_error` and `dout_fcs_valid`. The CRC is aligned with the end of packet signal.

When operating in cut-through mode, runt filtering is configurable and the FCS result is preserved. Runt filtering is enabled by the `RX_FILTER_CTRL[3]` (0x103) register.

When operating in the cut-through or filtering modes with Avalon–ST or a custom interface, the FCS result is always preserved.

RX CRC Forwarding

The CRC-32 field is forwarded to the client interface after the last byte of data, if the CRC removal option is not enabled.

Address Checking

The RX MAC supports all three types of addresses:

- Unicast—Specifies a destination address is a unicast (individual) address. Bit 0 is 0.
- Multicast—Specifies a destination address is a multicast or group address. Bit 0 is 1.
- Broadcast—Specifies a broadcast address when all 48 bits in the destination address are all 1s, 48'hFFFF_FFFF_FFFF.

If destination address matching is disabled, all frames are accepted. If destination address matching is enabled, address checking compares the address to the addresses programmed in the destination address register. The destination address table can store 1 address. You must enable filtering to discard mismatched destination addresses. For more information, refer to the "MAC Configuration and Filter Registers" on page 3–46 and "MAC Address Registers" on page 3–50. If the promiscuous mode of operation is enabled, address checking is omitted and all received client data frames are accepted.

Pause Frame Termination

Pause frames are forwarded in the same manner as any other frame. For more information about forwarding frames, refer to "Address Checking" on page 3–15.

Inter Packet Gap

IPG octets received are removed by the MAC RX and are not forwarded to the client.

Pause Ignore

When the pause frame receive enable bits are not set, the pause frames are not processed, so that the MAC TX traffic is not affected by the valid pause frames. You can enable unicast or multicast pause receive by setting the appropriate bits of the pause registers. For more information, refer to "Pause Registers" on page 3–47.

RX Data Bus Interfaces

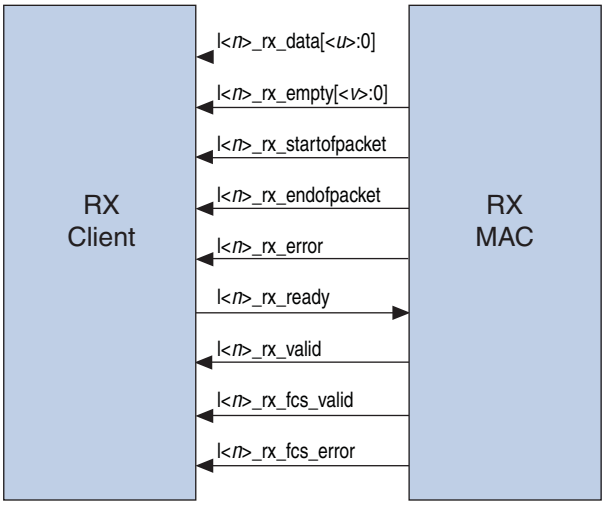
This section describes the RX data bus at the user interface, with and without adapters. It includes the following topics:

- RX Data Bus with Adapters (Avalon-ST Interface)
- RX Data Bus without Adapters (Custom Streaming Interface)

RX Data Bus with Adapters (Avalon-ST Interface)

The adapter for the RX interface of the 100GbE IP core increases the bus width from 5 words (320 bits) to 8 words (512 bits). The adapter for the RX interface of the 40GbE IP core increases the bus width from 2 word (128 bits) to 4 words (256 bits). The Avalon-ST interface always locates the SOP at the MSB, simplifying the interpretation of incoming data. Figure 3–16 illustrates this interface.

Figure 3–16. RX Client to MAC Interface with Adapter



Note to Figure 3–16:

(1) <n> = 4 for the 40GbE IP core and <n> = 8 for the 100GbE IP core. Lane variables are denoted by <u> and <v>.

Table 3–4 describes the signals in RX interface.

G40

Table 3–4. RX Interface with Adapters (Part 1 of 2)

Name	Direction	Description
l<n>_rx_data[<u>:0]	Output	RX data.
l<n>_rx_empty[<v>:0]	Output	Indicates the number of empty bytes in a packet if l<n>_rx_endofpacket is asserted, starting from the least significant byte (LSB).
l<n>_rx_startofpacket	Output	When asserted, indicates the start of a packet. The packet starts on the MSB.
l<n>_rx_endofpacket	Output	When asserted, indicates the end of packet.
l<n>_rx_error	Output	When asserted indicates an error condition.
l<n>_rx_ready	Input	Unused.

Table 3–4. RX Interface with Adapters (Part 2 of 2)

Name	Direction	Description
l<*n*>_rx_valid	Output	When asserted, indicates that RX data is valid. Only valid between the l<*n*>_rx_startofpacket and l<*n*>_rx_endofpacket signals.
l<*n*>_rx_fcs valid	Output	When asserted, indicates that FCS is valid.
l<*n*>_rx_fcs_error	Output	When asserted, indicates an FCS error condition.

Note to Table 3–4:

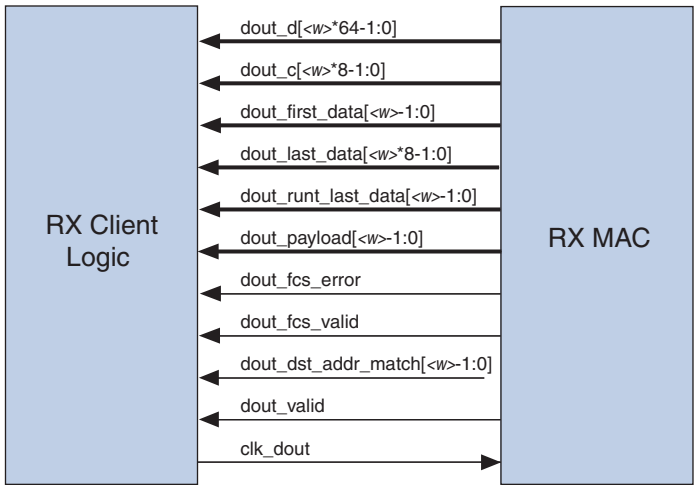
(1) <*n*> = 8 for the 8-to-5 adapter and <*n*> = 4 for the 4-to-2 adapter. Lane variables are denoted by <*u*> and <*v*>.

Refer to "Traffic on the TX and RX Client Interface for 100GbE IP Core Using the Eight- to Five-Word Adapters" on page 3–8 and "Traffic on the TX and RX Client Interface for 40GbE IP Core Using the Four- to Two-Word Adapters" on page 3–8 for timing diagrams illustrating this interface.

RX Data Bus without Adapters (Custom Streaming Interface)

The RX bus without adapters consists of five, 8-byte words or 320 bits operating at a frequency above 312.5 MHz for the 100GbE IP core or two, 8-byte words or 128 bits for the 40GbE IP core, nominally at 315 MHz. This bus drives data from the RX MAC to the RX client as Figure 3–17 illustrates.

Figure 3–17. RX PCS to RX MAC Interface



Note to Figure 3–17:

(1) <*w*> = 2 for the 40GbE IP core and <*w*> = 5 for the 100GbE IP core.

Table 3–5 describes the signals in the RX data bus.

Table 3–5. RX Data Bus without Adapters (Part 1 of 2)

Signal Name	Direction	Description
dout_d[<*w*>*64-1:0]	Output	Received data bytes.
dout_c[<*w*>*8-1:0]	Output	Control bits.
dout_first_data[<*w*>-1:0]	Output	The first data byte of a frame, after the preamble.

Table 3–5. RX Data Bus without Adapters (Part 2 of 2)

Signal Name	Direction	Description
dout_last_data[<w>8:0]	Output	The last data byte of a frame, before the FCS.
dout_runt_last_data[<w>-1:0]	Output	Last data corresponds to a runt frame (< 64 bytes).
dout_payload[<w>-1:0]	Output	Word contains packet data as opposed to control information or idling.
dout_fcs_error	Output	Last data corresponds to a frame with an incorrect FCS (CRC-32) value.
dout_fcs_valid	Output	The FCS error bit is valid.
dout_dst_addr_match[<w>-1:0]	Output	The first data word is a frame which matches one of the specified destination addresses.
dout_valid	Output	The dout bus contents are valid. This signal will occasionally be deasserted due to clock crossing.
clk_dout	Output	RX MAC clock. This clock must exceed 312.5 MHz. Altera recommends 315 MHz. The clk_dout and clk_din which clocks the TX datapath are not related and their rates do not have to match.

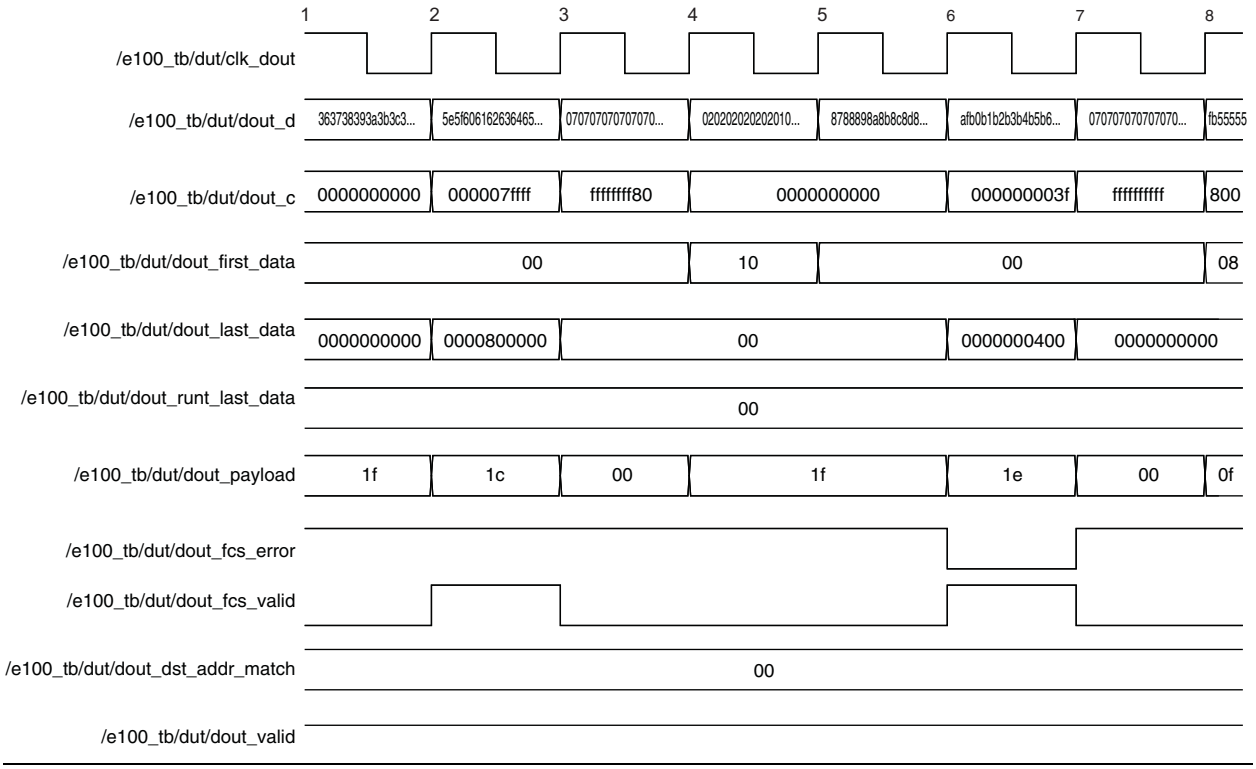
Note to Table 3–5:

(1) <*w*> = 2 for the 40GbE IP core and <*w*> = 5 for the 100GbE IP core.

The data bytes use 100 Gigabit Media Independent Interface (CGMII-like) encoding. For packet payload bytes, the dout_c bit, is set to 0 and the dout_d byte is the packet data. You can use this information to transmit out-of-spec data such as customized preambles when implementing non-standard variants of the *IEEE 802.3ba-2010 100G Ethernet Standard*. If the additional customized data is not required, simply ignore all words which are marked with dout_payload = 0 and discard the dout_c bus.

Figure 3–18 illustrates typical RX bus activity.

Figure 3–18. RX Bus Activity without Adapters



In Figure 3–18, `dout_last_data` is asserted in the second cycle, indicating the end of a packet. The `dout_payload` signal returns to 0 (5'h1c = 5'b11100). The `dout_c` and `dout_d` busses are set to 1b'1 and 8'h07 respectively to indicate idling. In the fourth cycle, `dout_first_data` is asserted and a short packet begins. This packet terminates successfully in the final cycle. Note that the first packet has a CRC error (`dout_fcs_error` = 1 and `fcs_valid` = 1).

The `dout_first_data` signal marks the first byte of data after the preamble. This byte is always the most significant byte of the word. There are 5 legal starting positions for the 100GbE IP core and 2 legal starting positions for the 40GbE IP core. The `dout_last_data` signal marks the last data byte before the FCS (CRC). It can occur at any byte position.

The `dout_runt_last_data` signal indicates that the packet ending in this word is less than the minimum legal length of 64 bytes from first data to the last FCS byte inclusive. Runts of eight or fewer bytes cannot be legally represented in CGMII and trigger a decoding error rather than this flag.

The `dout_fcs_error` and `dout_fcs_valid` signals indicate the result of the CRC checking logic. `dout_fcs_valid` = 1 and `dout_fcs_error` = 1 indicate a corrupted frame. Note that the CRC checking network works correctly on runts of 40–63 bytes. Runts of less than 40 bytes may be incorrectly determined to have a proper CRC.

The `dout_payload` signal marks words that contain frame data payload as opposed to idle or sequence control information, or preamble/frame delimiters.

The dout_valid signal exists for clock rate compensation. This signal is almost always asserted. When dout_valid is deasserted all other dout signals should be ignored for one clock cycle.

RX Selective Buffering

The RX interface can use an optional adapter which filters incoming packets based upon various annotations from the RX MAC interface, removing packets which do not meet the criteria specified. You can control the filtering criteria via the software register interface. Removing packets from the stream necessitates a deep RX-side FIFO memory and operation in "store and forward" mode.

Please contact your local Altera sales representative (www.altera.com/corporate/contact/con-index.html) for current availability.

Error Conditions on RX Datapath

The RX MAC indicates error conditions by asserting l<*n*>_rx_error. The following error conditions are defined:

- Received frame terminated early or with an error
- Received frame has a CRC error
- Error characters received from PHY
- Receive frame is too short (less than 64 bytes) or too long (longer than the maximum specified length)

Congestion and Flow Control Using Pause Frames

The 40-100GbE IP cores provide flow control to control congestion at the local or remote link partner. When either link partner devices experience congestion, the respective transmit control sends pause frames. The pause frame instructs the remote transmitter to stop sending data for the duration specified by the congested receiver.

When a device receives the XOFF pause control frame, it stops transmitting frames to the link partner for a period equal to the pause quanta of the incoming pause frame. The pause quanta can be configured in the pause quanta register of the device sending XOFF frames. If the pause frame is received in the middle of a frame transmission, the transmitter finishes sending the current frame and then suspends transmission for a period specified by the pause quanta. Data transmission resumes when a pause frame with quanta of zero is received or when the timer has expired. The pause quanta received overrides any counter currently stored. When more than one pause quanta is sent, the value of the pause is set to the last quanta received.

XOFF pause frames stop the remote transmitter. XON pause frames let remote transmitter resume data transmission. Figure 3–19 illustrates these frames.

Figure 3–19. The XOFF and XON Pause Frames (1) (Part 1 of 2)

XOFF Frame
START[7:0]
PREAMBLE[47:0]
SFD[7:0]

XON Frame
START[7:0]
PREAMBLE[47:0]
SFD[7:0]

Figure 3–19. The XOFF and XON Pause Frames (1) (Part 2 of 2)

XOFF Frame
DESTINATION ADDRESS[47:0] = 0x010000C28001 (2)
SOURCE ADDR[47:0]
TYPE[15:0] 0x8808
OPCODE[15:0] - 0X0001
PAUSE QUANTA[15:0] = 0xP1, 0xP2 (3)
PAD[335:0]
CRC[31:0]

XON Frame
DESTINATION ADDRESS[47:0] = 0x010000C28001
SOURCE ADDR[47:0]
TYPE[15:0] 0x8808
OPCODE[15:0] - 0X0001
PAUSE QUANTA[15:0] = 0x00000000
PAD[335:0]
CRC[31:0]

Note to Figure 3–19:

(1) One pause quanta fraction is equivalent to 512 bit times, which equates to 512/64 (the width of the MAC data bus), or 8 cycles for the system clock.

(2) This is a multicast destination address. You can use a unicast address if enabled in the pause register.

(3) The bytes `P1` and `P2` are filled with the value configured in the `pause_quant` register.

Conditions Triggering XOFF Frame Transmission

The TX MAC transmits XOFF frames when one of the following conditions occurs:

- Client requests XOFF transmission—A client can explicitly request that XOFF frames be sent using the pause control interface signals. When `pause_insert_tx` is asserted and `pause_insert_time` is not zero, an XOFF frame is sent to the Ethernet network when the current frame transmission completes. This option is not available for the 40-100GbE IP core with adapters.
- Host (software) requests XOFF transmission—Setting the pause control register triggers a request that an XOFF frame be sent. This option is available for the 40-100GbE IP core with adapters.

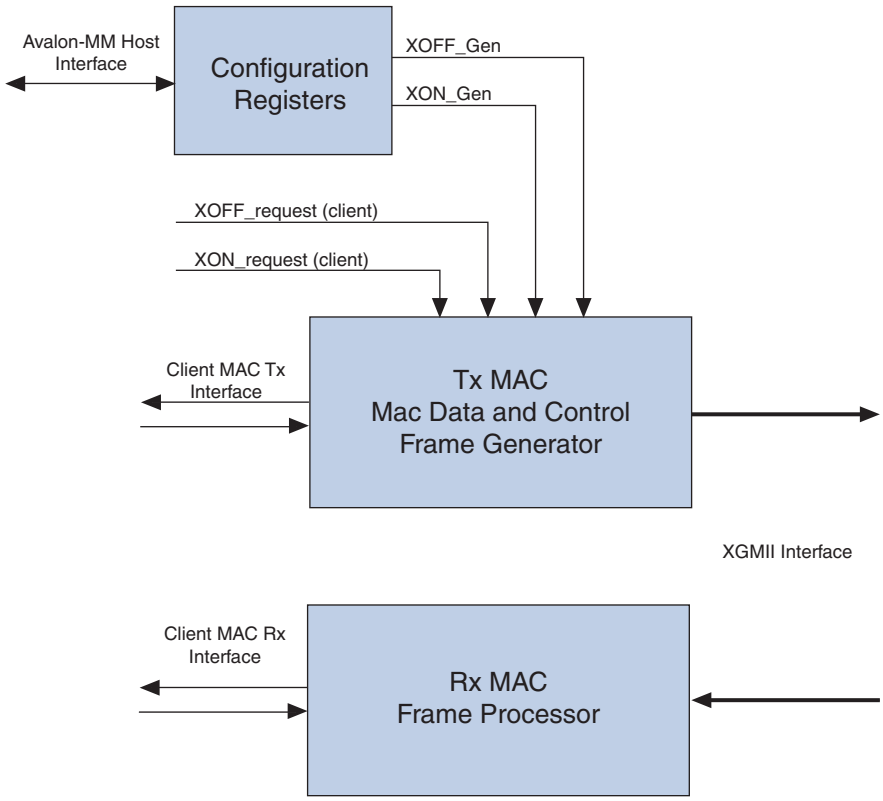
Conditions Triggering XON frame transmission

The TX MAC transmits XON frames when one of the following conditions occurs:

- Client requests XON transmission—A client can explicitly request that XON frames be sent using the pause control interface signals. `pause_insert_tx` is asserted and `pause_insert_time` is zero, an XON frame is sent to the Ethernet network when the current frame transmission completes. This option is not available for the 40-100GbE IP core with adapters.
- Host (software) requests XON transmission—Setting the pause control register triggers a request that an XON frame be sent. This option is available for the 40-100GbE IP core with adapters.

Figure 3–20 illustrates the pause transmission logic.

Figure 3–20. Block Diagram of Pause Transmission Logic



Pause Control and Generation Interface

The pause control interface implements flow control as specified by the *IEEE 802.3ba-2010 100G Ethernet Standard*. The pause logic, upon receiving a pause packet, temporarily stops packet transmission, and can insert an outgoing pause packet in the transmit direction. Table 3–6 describes the signals that implement the pause control functionality. These signals are only available for the 40-100GbE IP core without adapters; however, you can also access the pause functionality via the pause registers for any variant of the Ethernet IP core. For more information, refer to "Pause Registers" on page 3–47.

Table 3–6. Avalon-MM Control and Status Bus (Part 1 of 2)

Signal Name	Direction	Description
pause_insert_tx	Input	Edge triggered signal which inserts a pause packet.
pause_insert_time[15:0]	Input	Specifies the duration of the pause in pause quanta. The pause control settings, described in Table 3–34 on page 3–47 determine the duration of the pause quanta (pause quanta is equal to 512-bit time).
pause_insert_mcast	Input	When asserted, specifies that a pause packet with the well-known multicast address of 01-80-C2-00-00-01. If deasserted, the pause is generated with the specified MAC address (pause-insert-dst), which is typically a unicast address.

Table 3–6. Avalon-MM Control and Status Bus (Part 2 of 2)

Signal Name	Direction	Description
pause_insert_dst[47:0]	Input	Specifies the MAC address for a unicast pause.
pause_insert_src[47:0]	Input	Specifies source address of the pause packet.
pause_match_from_rx	Output	Asserted when there is an RX pause signal match. Used only when RX configurations are instantiated.
pause_time_from_rx[15:0]	Output	Asserted for RX pause time. Used only when RX configurations are instantiated.
pause_match_to_tx	Input	Asserted when there is an TX pause signal match. Used only when TX configurations are instantiated.
pause_time_to_tx[15:0]	Input	Asserted for TX pause time. Used only when TX configurations are instantiated.

Modes of Operations

You can configure the 40-100GbE IP core datapath for different modes of operation. The following sections describe these options.

Regular Mode

This is the normal mode of operation when the MAC transmits and receives data to and from a remote link partner MAC. It is also called the filtered or non-promiscuous mode of operation. In this mode, MAC TX performs all enabled functions on client data and transmits on the physical media. The RX MAC performs address filtering, various header checking, and control frame termination as per the IEEE 802.3 standard. You must enable filtering to discard mismatched destination addresses. For more information, refer to the "MAC Configuration and Filter Registers" on page 3–46 and "MAC Address Registers" on page 3–50.

The following functions are available in the regular mode:

- TX Operations
  - Start, preamble, and SFD insertion
  - Payload padding
  - Source address overwrite by MAC (optional)
  - CRC insertion
  - Inter-packet gap (IPG) insertion
- RX Operations
  - Preamble, SFD, and IPG striping
  - CRC checking
  - CRC stripping
  - Address checking
  - Runt, oversized frame or FCS error filtering
  - No payload padding removal

Promiscuous Mode

To enable promiscuous mode, set `RX_FILTER_CTRL[0]` to one. In promiscuous mode, most of the MAC functions, especially on the RX datapath are ignored; the data is passed on to the client after stripping the preamble and SFD. The following functions are available in promiscuous mode:

- TX Operations
  - Add preamble and SFD
  - Payload padding
  - CRC insertion
- RX Operations
  - Preamble, SFD, and IPG stripping
  - CRC check plus forwarding to the client.
  - No Payload padding removal

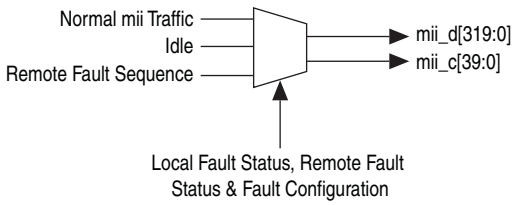
Link Fault Signaling Interface

The 40-100GbE IP core provides link fault signaling as defined in the *IEEE 802.3ba-2010 100G Ethernet Standard*. The 40GbE and 100GbE MAC include a Reconciliation Sublayer (RS) located between the MAC and the XLGMII or CGMII to manage local and remote faults. Link fault signaling is disabled by default but can be enabled by the `Enable Link Fault Sequence` register. When enabled, local RS TX can transmit remote fault sequences in case of a local fault and can transmit idle control words in case of a remote fault. An additional configuration register (`MAC/RS link fault sequence configuration`) is provided to select the type of information to be transmitted in case of a local or remote fault. Using the configuration bits, you can send remote fault sequence ordered sets, idle control words, or regular traffic in the case of a local or remote fault. Table 3–27 and Table 3–28 describe the link fault signaling registers.

The RS RX sets `remote_fault_status` or `local_fault_status` to 1 when RS RX receives remote fault or local fault sequence ordered sets. When valid data is received in more than 127 columns, the RS RX resets the relevant fault status (`remote_fault_status` or `local_fault_status`) to 0.

Figure 3–21 shows an example of link fault signaling.

Figure 3–21. Link Fault Signaling Example



IEEE specifies RS monitoring of RXC<7:0> and RXD<63:0> for Sequence `ordered_sets`. For more information, refer to *Figure 81–9—Link Fault Signaling state diagram* and *Table 81-5—Sequence* `ordered_sets` from the *IEEE 802.3ba-2010 100G Ethernet Standard* available from the IEEE website (www.ieee.org). The variable `link_fault` is set to indicate the value of as RX Sequence `ordered_set` when four `fault_sequences` containing the same fault value are received with fault sequences separated by less than 128 columns and there are no intervening `fault_sequences` of different fault values. The variable `link_fault` is set to OK following any interval of 128 columns not containing a remote fault or local fault Sequence `ordered_set`.

Table 3–7 describes the link fault signaling interface signals.

Table 3–7. Link Fault Signaling Interface

Signal Name	Direction	Description
`remote_fault_from_rx`	Output	Asserted when remote fault is detected in RX MAC. Used only when RX configurations are instantiated.
`local_fault_from_rx`	Output	Asserted when local fault is detected in RX MAC. Used only when RX configurations are instantiated.
`remote_fault_to_tx`	Input	Asserted when remote fault is detected. Used only when TX configurations are instantiated.
`local_fault_to_tx`	Input	Asserted when local fault is detected. Used only when TX configurations are instantiated.

For more information on the Link Fault, refer to "Link Fault Signaling Registers" on page 3–44.

Statistics Counters Interface

For information on the increment vectors of statistics counters, refer to "Statistics Counters Increment Vectors" on page 3–55.

MAC – PHY XLGMII or CGMII Interface

The PHY side of MAC interfaces implements the XLGMII or CGMII protocol as defined by IEEE 802.3ba standard. The standard XLGMII or CGMII implementation consists of 32 bit wide data bus. However, the Altera implementation uses a wider bus interface in connecting a MAC to the internal PHY. The width of this interface is 320 bits for the 100GbE IP core and 128 bits for the 40GbE IP core.

Table 3–8 lists XL/CGMII permissible encodings. Memorizing a few of the XL/CGMII encodings greatly facilitates understanding of Ethernet waveforms. The XL/CGMII encodings are backwards compatible with older Ethernet and have convenient mnemonics. The `DATAPATH_OPTION` parameter instantiates TX and RX backwards compatibility and is set by default.

Table 3–8. XL/CGMII Permissible Encodings (Part 1 of 2)

Control	Data	Description
0	xx	Packet data, including preamble and FCS bytes.
1	07	Idle.
1	fb	Start of Frame (fb = frame begin).

Table 3–8. XL/CGMII Permissible Encodings (Part 2 of 2)

Control	Data	Description
1	fd	End of Frame (fd = frame done).
1	fe	XL/CGMII Error. Typically a bit error which switched a 66-bit block between data and control, or corrupt control information (fe = frame error).

MAC to PHY Connection Interface

The MAC to PHY connection interface is included in the 40-100GbE MAC and PHY IP Cores without adapters. Table 3–9 describes the TX and RX connections.

Table 3–9. MAC to PHY TX and RX Connections

Signal Name	Direction	Description
TX MAC to PHY Connections (from MAC)		
miitx_rev_d[<w>*64-1:0]	Output (MAC only)	Media independent interface (MII) data TX connection, starting with the least significant bit (LSB)
miitx_rev_c[<w>*8-1:0]		MII control TX connection
miitx_valid		Asserted upon valid TX to MII connection
din_ready	Input (PHY only)	Asserted when TX data bus is ready for MII connection
tx_lanes_stable		Asserted when TX lanes are stable and deskewed
RX MAC to PHY Connections (from MAC)		
rx_raw_mii_d[<w>*64-1:0]	Input (PHY only)	MII data RX connection, starting with the LSB and least significant word, outputting a 5-word data stream
rx_raw_mii_c[<w>*8-1:0]		MII control RX connection
rx_blocks_valid		Asserted when RX blocks are valid

Lane to Lane Deskew Interface

The lane to lane deskew signal is included in the 40-100GbE IP Cores with and without adapters. When both MAC and PHY options are selected, the lane to lane deskew signal acts as an internal signal. Table 3–10 describes the lane to lane deskew interface.

Table 3–10. Lane to Lane Deskew

Signal Name	Direction	Description
lanes_deskewed	Input	Indicates lane to lane skew is corrected. Available as an input to the 40-100GbE MAC IP cores only.
lanes_deskewed	Output	Indicates lane to lane skew is corrected. Available as an output from the 40-100GbE PHY IP cores only.

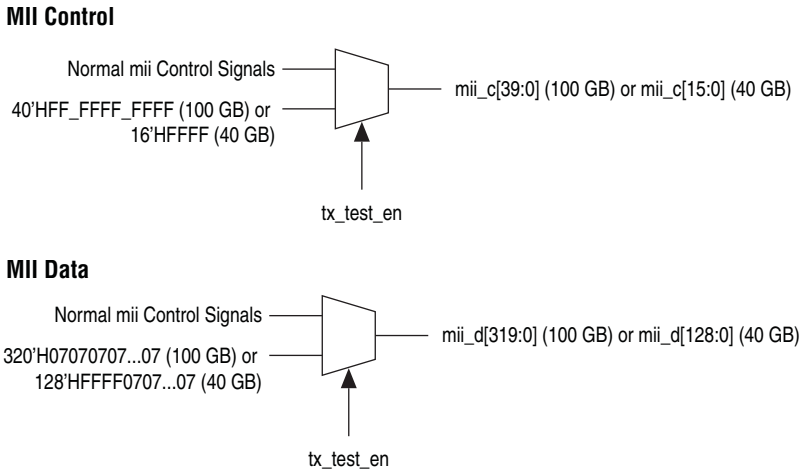
PCS Test Pattern Generation and Test Pattern Check

The PCS can generate a test pattern and detect a scrambled idle test pattern. PCS test-pattern mode is suitable for receiver tests and for certain transmitter tests.

When a scrambled idle pattern is enabled, a test pattern is generated by the scrambler. No seeding of the scrambler is required during test-pattern operation. The input to the scrambler is a control block (blocktype=0x1E). Synchronous headers and alignment markers are added to the data stream enabling the RX PCS to align and deskew the PCS lanes. For information on the definition of idle test patterns, refer to *Figure 82–5—64B/66B block formats* illustrated in the *IEEE 802.3ba-2010 100G Ethernet Standard* available from the IEEE website (www.ieee.org).

The generation of the PCS test pattern is shown in Figure 3–22.

Figure 3–22. PCS Test Pattern Generation



The scrambled idle test-pattern checker utilizes the block lock state diagram, the alignment marker state diagram, the PCS deskew state diagram, and the descrambler; these blocks operate as if in normal data reception. The BER monitor state diagram is disabled during RX test-pattern mode. When `align_status` is true and the scrambled idle RX test-pattern mode is active, the scrambled idle test-pattern checker observes the synchronous header and the output from the descrambler. When the synchronous header and the output of the descrambler is an idle pattern, a match is detected. When operating in scrambled idle test pattern, the test-pattern error counter counts blocks with a mismatch. Any mismatch indicates an error and shall increment the test-pattern error counter.

The test-pattern check includes the following steps:

1. The single register bit `rx_test_en` enables the RX test-pattern mode.
2. A 32-bit register implemented as a test-pattern error counter counts the number of mismatched blocks when in test-pattern mode.
3. The single register bit `Clr_counter` enables software to clear the test-pattern error counter.

Transceiver PHY Serial Data Interface

The core uses a 40-bit ×<*n*> lane digital interface to send data to the TX high-speed serial I/O pins operating at 10.3125 Gbps. The `rx_datain` and `tx_dataout` ports connect to the 10.3125 Gbps pins. The protocol includes automatic reordering of serial lanes so that any ordering is acceptable. Virtual lanes 0 and 1 transmit data on `tx_dataout[0]`.

PCS BER Monitor

The PCS implements bit error rate (BER) monitoring as specified by the *IEEE 802.3ba-2010 100G Ethernet Standard*. When the PCS deskews the data and aligns the lanes, the BER monitor checks the signal quality and asserts `hi_ber` if it detects excessive errors. When `align_status` is asserted and `hi_ber` is deasserted, the RX PCS continuously accepts blocks and generates RXD <63:0> and RXC <7:0> on the XLGMII or CGMII interface.

High BER occurs when 97 invalid 66-bit synchronous headers are detected for 100GbE within 500 µs or detected for 40GbE within 1.25 ms. When less than 97 invalid 66-bit synchronous headers occur in the same window the high BER state is exited.

For more information, refer to *Figure 82–13—BER monitor state diagram* illustrated in the *IEEE 802.3ba-2010 100G Ethernet Standard* available from the IEEE website (www.ieee.org).

Control and Status Interface

The control and status interface provides an Avalon-MM interface to the control and status registers. The Avalon-MM interface implements a standard memory-mapped protocol. You can connect an embedded processor or JTAG Avalon master to this bus to access the control and status registers. Table 3–11 describes the signals that comprise this interface.

Table 3–11. Avalon-MM Control and Status Interface

Signal Name	Direction	Description
`status_addr[15:0]`	Input	Address for reads and writes
`status_read`	Input	Read command
`status_write`	Input	Write command
`status_writedata[31:0]`	Input	Data to be written
`status_readdata[31:0]`	Output	Read data
`status_readdata_valid`	Output	Read data is ready for use

The status interface is designed to operate at a low frequencies, typically 50 MHz for Stratix IV devices and 100 MHz for Stratix V devices, so that control and status logic does not compete for resources with the surrounding high speed datapath. If you require immediate access to status information, you can connect to the status wires directly one level down in the design hierarchy. For information on the 40-100GbE IP Core registers, refer to "Software Register Interface" on page 3–36.

For more information about the Avalon-MM protocol, including timing diagrams, refer to the *Avalon Memory-Mapped Interfaces* chapter in the *Avalon Interface Specifications*.

Clocks

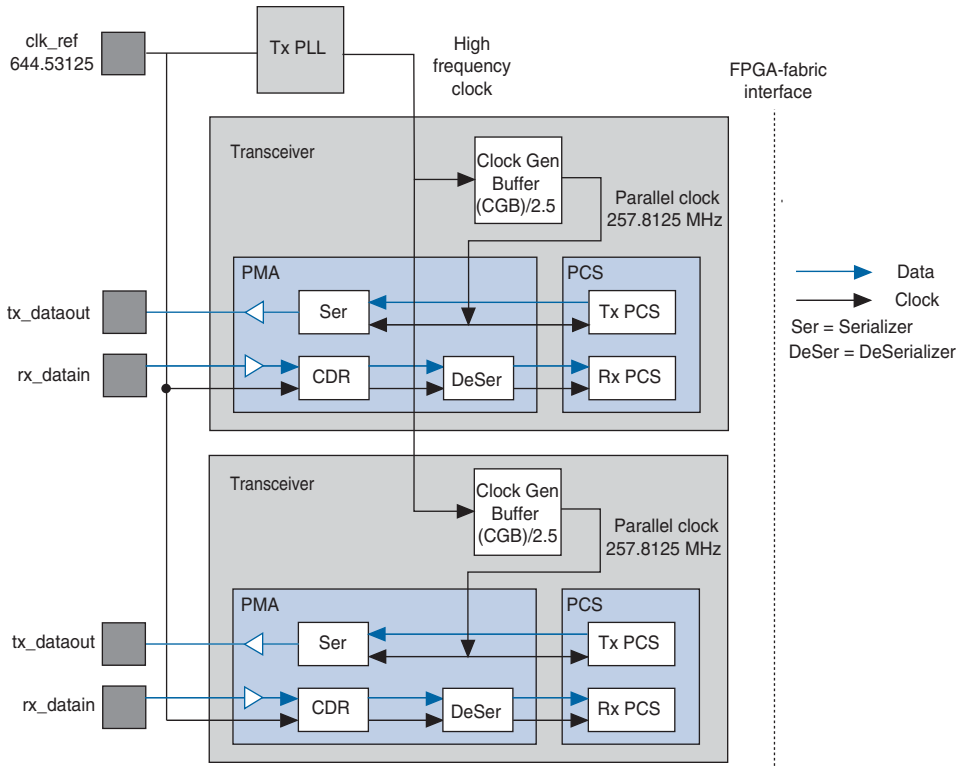
Table 3–12 describes the input clocks that you must provide.

Table 3–12. Clock Inputs

Signal Name	Direction	Description
clk_status	Input	A clock for reconfiguration, offset cancellation, and housekeeping functions. This clock is also used for clocking the control and status interface to the same source. The clock quality and pin chosen are not critical. clk_status is a 50 MHz clock for Stratix IV and a 100 MHz clock for Stratix V devices.
clk_ref	Input	Clk_ref is the reference clock for the transceivers that has a frequency of 644.53125 MHz with a ±100 ppm accuracy per the *IEEE 802.3ba-2010 100G Ethernet Standard.* In addition, clk_ref must meet the jitter specification of the *IEEE 802.3ba-2010 100G Ethernet Standard.* The PLL and clock generation logic use this reference clock to derive the transceiver and PCS clocks. It should be a high quality signal on the appropriate dedicated clock pin. The PCS clock frequency is 257.8125 MHz. Refer to Figure 3–23 for a high-level view of the clock generation circuitry and distribution for the transceiver.
clk_din/clk_txmac	Input	The input TX clock for the IP cores without adapters is clk_din and with adapters is clk_txmac. The recommended clock frequency for clk_din and clk_txmac is 315 MHz and the minimum clock frequency is 312.5 MHz.
clk_dout/clk_rxmac	Input	The input RX clock for the IP cores without adapters is clk_dout and with adapters is clk_rxmac. The minimum clock frequency for clk_dout and clk_rxmac is 312.5 MHz.

Figure 3–23 provides a high-level view of the clock generation circuitry and distribution for the transceiver.

Figure 3–23. Clock Generation Circuitry



Resets

The 40-100GbE IP core provides five reset bits to allow independent reset control for all configurations. The reset bits are provided in the Avalon-ST and Avalon-MM interfaces.

The MAC and PHY asynchronous reset signals are included in the 40-100GbE IP Core with adapters and without adapters. Table 3–13 describes the reset signals.

Table 3–13. Asynchronous Reset Signals

Signal Name	Direction	Description
mac_rx_arst_ST	Input	MAC RX asynchronous reset signal
mac_tx_arst_ST	Input	MAC TX asynchronous reset signal
pcs_rx_arst_ST	Input	PHY PCS RX asynchronous reset signal
pcs_tx_arst_ST	Input	PHY PCS TX asynchronous reset signal
pma_arst_ST	Input	PHY PMA asynchronous reset signal

The pma_arst_ST TX reset must be released simultaneously with, or immediately after, the mac_tx_arst_ST reset. The mac_tx_arst_ST reset cannot be reset without resetting the pma_arst_ST TX reset. Altera recommends that all parts of the 40-100GbE IP core are reset simultaneously. The reset signal must be held for at least 1 status clock cycle.

Port Listing by Transmission Mode Parameter Configuration

Table 3–14 shows the parameter configurations for the TX and RX configuration ports in the 40-100GbE IP Cores without adapters.

Table 3–14. IP Cores Without Adapters: Configuration Ports by Transmission Mode (Part 1 of 4)

Signal Name	Direction	Description
TX Transmission Mode Configuration Ports		
mac_tx_arst_ST	Input	Refer to "Resets" on page 3–30
pcs_tx_arst_ST	Input	
pma_arst_ST	Input	
clk_ref	Input	Refer to "Clocks" on page 3–29
tx_dataout[9:0]	Output	Refer to "Transceiver PHY Serial Data Interface" on page 3–28
clk_din	Input	Refer to "Clocks" on page 3–29 and "TX Data Bus without Adapters (Custom Streaming Interface)" on page 3–9
din[<w>*64-1:0]	Input	Refer to "TX Data Bus without Adapters (Custom Streaming Interface)" on page 3–9
din_start[<w>-1:0]	Input	
din_end_pos[<w>*8-1:0]	Input	
din_ack	Output	
clk_status	Input	Refer to "Clocks" on page 3–29
status_addr[15:0]	Input	Refer to "Control and Status Interface" on page 3–28
status_read	Input	
status_write	Input	
status_writedata[31:0]	Input	
status_readdata[31:0]	Output	
status_readdata_valid	Output	
pause_insert_tx	Input	Refer to "Pause Control and Generation Interface" on page 3–22
pause_insert_time[15:0]	Input	
pause_insert_mcast	Input	
pause_insert_dst[47:0]	Input	
pause_insert_src[47:0]	Input	
pause_match_to_tx	Input	
pause_time_to_tx[15:0]	Input	
remote_fault_to_tx	Input	Refer to "Link Fault Signaling Interface" on page 3–24
local_fault_to_tx	Input	

Table 3–14. IP Cores Without Adapters: Configuration Ports by Transmission Mode (Part 2 of 4)

Signal Name	Direction	Description
tx_inc_64	Output	Refer to "Statistics Registers" on page 3–50 and "Statistics Counters Increment Vectors" on page 3–55
tx_inc_127	Output	
tx_inc_255	Output	
tx_inc_511	Output	
tx_inc_1023	Output	
tx_inc_1518	Output	
tx_inc_max	Output	
tx_inc_over	Output	
tx_inc_mcast_data_err	Output	
tx_inc_mcast_data_ok	Output	
tx_inc_bcast_data_err	Output	
tx_inc_bcast_data_ok	Output	
tx_inc_ucast_data_err	Output	
tx_inc_ucast_data_ok	Output	
tx_inc_mcast_ctrl	Output	
tx_inc_bcast_ctrl	Output	
tx_inc_ucast_ctrl	Output	
tx_inc_pause	Output	
tx_inc_fcs_err	Output	
tx_inc_fragment	Output	
tx_inc_jabber	Output	
tx_inc_sizeok_fcserr	Output	
RX Transmission Mode Configuration Ports		
mac_rx_arst_ST	Input	Refer to "Resets" on page 3–30
pcs_rx_arst_ST	Input	
pma_arst_ST	Input	
clk_ref	Input	Refer to "Clocks" on page 3–29
rx_datain[9:0]	Input	Refer to "Transceiver PHY Serial Data Interface" on page 3–28
clk_dout	Input	Refer to "Clocks" on page 3–29 and "RX Data Bus without Adapters (Custom Streaming Interface)" on page 3–17

Table 3–14. IP Cores Without Adapters: Configuration Ports by Transmission Mode (Part 3 of 4)

Signal Name	Direction	Description
dout_d[<w>*64-1:0]	Output	Refer to "RX Data Bus without Adapters (Custom Streaming Interface)" on page 3–17
dout_c[<w>*8-1:0]	Output	
dout_first_data[<w>-1:0]	Output	
dout_last_data[<w>*8-1:0]	Output	
dout_runt_last_data[<w>-1:0]	Output	
dout_payload[<w>-1:0]	Output	
dout_fcs_error	Output	
dout_fcs_valid	Output	
dout_dst_addr_match[<w>-1:0]	Output	
dout_valid	Output	
clk_status	Input	Refer to "Clocks" on page 3–29
status_addr[15:0]	Input	Refer to "Control and Status Interface" on page 3–28
status_read	Input	
status_write	Input	
status_writedata[31:0]	Input	
status_readdata[31:0]	Output	
status_readdata_valid	Output	

Table 3–14. IP Cores Without Adapters: Configuration Ports by Transmission Mode (Part 4 of 4)

Signal Name	Direction	Description
rx_inc_runt	Output	Refer to "Statistics Registers" on page 3–50 and "Statistics Counters Increment Vectors" on page 3–55
rx_inc_64	Output	
rx_inc_127	Output	
rx_inc_255	Output	
rx_inc_511	Output	
rx_inc_1023	Output	
rx_inc_1518	Output	
rx_inc_max	Output	
rx_inc_over	Output	
rx_inc_mcast_data_err	Output	
rx_inc_mcast_data_ok	Output	
rx_inc_bcast_data_err	Output	
rx_inc_bcast_data_ok	Output	
rx_inc_ucast_data_err	Output	
rx_inc_ucast_data_ok	Output	
rx_inc_mcast_ctrl	Output	
rx_inc_bcast_ctrl	Output	
rx_inc_ucast_ctrl	Output	
rx_inc_pause	Output	
rx_inc_fcs_err	Output	
rx_inc_fragment	Output	
rx_inc_jabber	Output	
rx_inc_sizeok_fcserr	Output	
pause_match_from_rx	Output	Refer to "Pause Control and Generation Interface" on page 3–22
pause_time_from_rx[15:0]	Output	
remote_fault_from_rx	Output	Refer to "Link Fault Signaling Interface" on page 3–24
local_fault_from_rx	Output	

Many port configurations for the TX and RX ports in the 40-100GbE IP Cores without adapters are identical to the IP cores with adapters. Table 3–15 identifies the port configurations that differ for the 40-100GbE IP Cores with adapters.

Table 3–15. IP Cores With Adapters: Configuration Ports by Transmission Mode (Part 1 of 2) (1)

Signal Name	Direction	Description
TX Transmission Mode Configuration Ports		
clk_txmac	Input	Refer to "Clocks" on page 3–29

Table 3–15. IP Cores With Adapters: Configuration Ports by Transmission Mode (Part 2 of 2) (1)

Signal Name	Direction	Description
l<*n*>_tx_data[<*u*>:0]	Input	Refer to "TX Data Bus with Adapters (Avalon-ST Interface)" on page 3–6
l<*n*>_tx_empty[<*v*>:0]	Input	
l<*n*>_tx_startofpacket	Input	
l<*n*>_tx_endofpacket	Input	
l<*n*>_tx_ready	Output	
l<*n*>_tx_valid	Input	
tx_serial[9:0]	Output	This signal replaces the tx_dataout signal from the IP cores without adapters.
RX Transmission Mode Configuration Ports		
clk_rxmac	Input	Refer to "Clocks" on page 3–29
l<*n*>_rx_data[<*u*>:0]	Output	Refer to "RX Data Bus with Adapters (Avalon-ST Interface)" on page 3–16
l<*n*>_rx_empty[<*v*>:0]	Output	
l<*n*>_rx_startofpacket	Output	
l<*n*>_rx_endofpacket	Output	
l<*n*>_rx_error	Output	
l<*n*>_rx_valid	Output	
l<*n*>_rx_fcs_valid	Output	
l<*n*>_rx_fcs_error	Output	
rx_serial[9:0]	Input	This signal replaces the rx_datain signal from the IP cores without adapters.

Note to Table 3–15

(1) <*n*> = 4 for the 4-to-2 adapter. and <*n*> = 8 for the 8-to-5 adapter. Lane variables are denoted by <*u*> and <*v*>.

Table 3–16 identifies the parameter configurations that differ for the 40-100GbE MAC IP Cores.

Table 3–16. MAC IP Cores: Configuration Ports by Transmission Mode (Part 1 of 2)

Signal Name	Direction	Description
TX Transmission Mode Configuration Ports		
miitx_rev_d[<*w*>*64-1:0]	Output	Refer to "MAC to PHY Connection Interface" on page 3–26
miitx_rev_c[<*w*>*8-1:0]	Output	
miitx_valid	Output	
din_ready	Input	
tx_lanes_stable	Input	
RX Transmission Mode Configuration Ports		

Table 3–16. MAC IP Cores: Configuration Ports by Transmission Mode (Part 2 of 2)

Signal Name	Direction	Description
rx_raw_mii_d[<w>*64-1:0]	Input	Refer to "MAC to PHY Connection Interface" on page 3–26
rx_raw_mii_c[<w>*8-1:0]	Input	
rx_blocks_valid	Input	
lanes_deskewed	Input	Refer to "Lane to Lane Deskew Interface" on page 3–26

When simulating the full design the lanes_deskewed input comes from the output of the RX PCS, indicating a fully locked status. To avoid confusion, when simulating the alt_e40_mac or alt_e100_mac wrapper as the top level, drive the lanes_deskewed input and the din_ready input to '1'.

Table 3–17 identifies the parameter configurations that differ for the 40-100GbE PHY IP Cores.

Table 3–17. PHY IP Cores: Configuration Ports by Transmission Mode

Signal Name	Direction	Description
TX Transmission Mode Configuration Ports		
clk_din	Input	Refer to "Clocks" on page 3–29
miitx_rev_d[<w>*64-1:0]	Input	Refer to "MAC to PHY Connection Interface" on page 3–26
miitx_rev_c[<w>*8 -1:0]	Input	
miitx_valid	Input	
din_ready	Output	
tx_lanes_stable	Output	
RX Transmission Mode Configuration Ports		
clk_dout	Input	Refer to "Clocks" on page 3–29
rx_raw_mii_d[<w>*64-1:0]	Output	Refer to "MAC to PHY Connection Interface" on page 3–26
rx_raw_mii_c[<w>*8 -1:0]	Output	
rx_blocks_valid	Output	
lanes_deskewed	Output	Refer to "Lane to Lane Deskew Interface" on page 3–26

Software Register Interface

This section provides information on the memory-mapped registers. You access these registers using the control and status interface, described in the "Control and Status Interface" on page 3–28. The registers use 32-bit addresses; they are not byte addressable. Table 3–18 lists the memory mapped registers for the 40-100GbE IP core and example design.

Writes to a register marked as read-only are discarded without ill effect. Registers marked reserved return an unspecified constant on read. Writes to reserved registers are discarded.

The default register settings put the IP core in a typical operational state with the exception of the PMD_CMD_CONFIG register which is set to 0 (disabled). Enabling the PMD causes the IP core to seek RX lock. The IP core automatically begins locking to the RX data if it RX lock is lost.

For improved organization and clarity, the address map has been revised. In addition, a few register names have changed. If you are upgrading to Quartus II version 12.0 from an earlier release, refer to Appendix C, Address Map Changes for 12.0 for a complete list of changes.

Table 3–18. 100GbE and Example Design Address Map (Part 1 of 2)

Word Offset	Register Description
40-100GbE IP Core Registers	
0x000–0x009	Refer to "Scratch and Clock Registers for Stratix IV and Stratix V Devices" on page 3–39
0x00a–0x00f	Reserved
0x010–0x014	Refer to "Lock Status Registers" on page 3–41
0x015–0x016	Refer to "Bit Error Flag Registers" on page 3–42
0x017	Refer to "PCS Hardware Error Registers" on page 3–42
0x018	Refer to "BER Monitor Registers" on page 3–43
0x019	Refer to "Test Mode Registers" on page 3–43
0x01a	Refer to "Test Pattern Counter Registers" on page 3–44
0x01b	Refer to "Link Fault Signaling Registers" on page 3–44
0x01c	Reserved
0x01d	PHY reset registers. Refer to "MAC and PHY Reset Registers" on page 3–44
0x01e–0x01f	Reserved
0x020–0x023	Refer to "PCS-VLANE Registers" on page 3–45
0x024–0x02F	Reserved
0x030–0x032	Refer to "PRBS Registers" on page 3–46
0x033–0x03F	Reserved
0x040–0x07F (Stratix IV only)	Reserved
0x040–0x07F (Stratix V only)	Maps to word addresses 0x040-0x07F in the Low Latency PHY IP core register map. For more information, including loopback configuration, refer to the *Low Latency PHY IP Core* chapter in the *Altera Transceiver PHY IP Core User Guide.*
0x080–0x0FF (Stratix V only)	Maps to word addresses 0x000-0x07F in the Transceiver Reconfiguration Controller register map. Refer to the *Transceiver Reconfiguration Controller* chapter in the *Altera Transceiver PHY IP Core User Guide.*
0x100–0x103	Refer to "MAC Configuration and Filter Registers" on page 3–46
0x104–0x10F	Reserved
0x110–0x117	Refer to "Pause Registers" on page 3–47
0x118–0x13F	Reserved

Table 3–18. 100GbE and Example Design Address Map (Part 2 of 2)

Word Offset	Register Description
0x120	Refer to "MAC Hardware Error Registers" on page 3–49
0x121	MAC reset registers. Refer to "MAC and PHY Reset Registers" on page 3–44
0x122	Refer to "Link Fault Signaling Registers" on page 3–44
0x123	Refer to "CRC Configuration Registers" on page 3–49
0x140–0x17F	Refer to "MAC Address Registers" on page 3–49
0x180–0x1FF	Reserved
0x200–0x229	Refer to "TX Statistics Counters" on page 3–50
0x22A–0x22D	Refer to "TX Packet Statistics" on page 3–52
0x22E–0x27F	Reserved
0x280–0x2A9	Refer to "RX Statistics Counters" on page 3–52
0x2AA–0x2B8	Refer to "RX Packet Statistics" on page 3–54
0x2B9–0x3FF	Reserved
100GbE Ethernet Example Design Registers	
0x400–0x403	Refer to "PMD Registers" on page 3–56
0x404–0xf0f	Reserved
0x410–0x413	Refer to "MDIO Registers" on page 3–57
0x414–0x41f	Reserved
0x420–0x423	Refer to "2-Wire Serial Registers" on page 3–58

Register Initialization

To initialize the 40-100GbE IP core, complete the following steps:

1. Drive the clock ports as specified.
2. Reset the IP core (for more information, refer to "Resets" on page 3–30).
3. Clear the statistics counters by writing '1' to bit 3 of the general control `MAC_CMD_config` register.
4. Wait until the IP core is fully locked before driving the stimulus. Read the fully locked/deskew status from bit 1 of `RX_AGGREGATE` register.

40-100GbE IP Core Registers

The following sections describe the registers included in the 40-100GbE IP core.

Transceiver PHY Control and Status Registers

The serial clocks for 10.3125 Gbps data on a 40-bit interface should operate at 257812.5 KHz. The TX serial rate is based on the reference clock and should be precise and stable. The RX serial rate is recovered from the remote system. It typically shows some instability during lock acquisition. The core clocks should exceed 312500 KHz for proper operation.

The registers in the transceiver PHY provide dynamic access to the analog configuration capability on a per channel (pin) basis. You can also use these registers to place the transceivers in loopback mode for diagnostic or error injection testing. In loopback mode, the TX output connects to the corresponding RX channel.

Table 3–19 describes the scratch and clock registers for addresses for the Stratix IV and Stratix V devices.

Table 3–19. Scratch and Clock Registers for Stratix IV and Stratix V Devices (Part 1 of 2)

Address	Name	Applicable Device(s)	Bit	Description	HW Reset Value	Access
0x000	PHY_VERSION	Stratix IV and Stratix V	[31:0]	MegaCore PHY function revision.	0x00E01200 (40GbE) 0x 00DE1200 (100GbE)	R
0x001	SCRATCH_PHY	Stratix IV and Stratix V	[31:0]	Scratch register available for testing.	0x00000000	RW
0x002	CLK_TXS	Stratix IV and Stratix V	[19:0]	TX serial clock rate monitor in KHz.	0x00000	R
0x003	CLK_RXS	Stratix IV and Stratix V	[19:0]	RX serial clock rate monitor in KHz.	0x00000	R
0x004	CLK_TXC	Stratix IV and Stratix V	[19:0]	TX core clock rate monitor in KHz.	0x00000	R
0x005	CLK_RXC	Stratix IV and Stratix V	[19:0]	RX core clock rate monitor in KHz.	0x00000	R
0x006	–	Stratix IV and Stratix V	–	Reserved.	–	–
0x007	GX_CTRL1	Stratix IV only (1)	[31]	When set, places the transceiver in Internal serial loopback, from TX to RX.	1'b0	RW
			[30]	When asserted, the transceiver channel's analog settings are read on the rising edge of clk_status.	1'b0	RW
			[29]	When asserted, the transceiver channel's analog settings are written on rising edge of clk_status.	1'b0	RW
			[28:4]	Specifies the analog settings to write. Refer to Table 3–20 for the fields of these registers which are read on GX_REPLY[24:0]. Bits[28:4] of this register correspond to bits[24:0] of GX_REPLY.	0x000000	RW
			[3:0]	Specifies the logical channel to select [0–9].	0x0	RW
0x008	GX_CTRL2	Stratix IV and Stratix V	[1]	Specifies bit error inject position 1 on rising edge.	1'b0	RW
			[0]	Specifies bit error inject position 0 on rising edge.	1'b0	RW

Table 3–19. Scratch and Clock Registers for Stratix IV and Stratix V Devices (Part 2 of 2)

0x009	GX_REPLY	Stratix IV only (1)	[26]	When asserted, indicates that the transceiver is busy.	1'b0	R
			[25]	When asserted, indicates that read data is valid.	1'b0	R
			[24:0]	Contains the analog settings from last read.	0x020080	R

Note to Table 3–19

(1) Reserved for Stratix V devices.

Table 3–20 describes the fields of the analog settings register, GX_CTRL1. The default settings are a good starting point if you are connecting to a PMD module running at 10.3125 Gbps.

Table 3–20. Stratix IV Transceiver Analog Settings Registers

Bits	Description	HW Reset Value	Access
[24:21]	RX equalization control. The equalizer uses a pass band filter. Specifying a low value passes low frequencies. Specifying a high value passes high frequencies.	4'b0011	RW
[20:18]	RX DC gain. Sets the equalization DC gain using one of the following settings: ■ 0–0 dB ■ 1–3 dB ■ 2–6 dB ■ 3–9 dB ■ 4–12 dB ■ 5–15 dB ■ 6–18 dB ■ 7–21 dB	3'b000	RW
[17:13]	Sets the pre-emphasis for TX tap 0.	5'b10000	RW
[12:8]	Sets the pre-emphasis for TX pre-emphasis tap 1.	5'b00000	
[7:3]	Sets the pre-emphasis for TX pre-emphasis tap 2.	5'b10000	
[2:0]	TX V_{OD} (amplitude).	3'b010	

If you encounter signal integrity issues using the default settings in Table 3–20, the following procedure may be helpful:

1. Adjust V_{OD}. A value which is too high tends to cause interference with other lanes. You want to select the lowest value that functions correctly.
2. Raise pre-emphasis tap 1 slightly. Raising tap 1 tends to help when there is a long trace length or multiple connectors causing signal loss.
3. Adjust the equalization control. This control is roughly analogous to a stereo equalizer. It emphasizes and de-emphasizes portions of the signal by frequency.
4. Iterate, making minor adjustments to all of the controls while monitoring the error rate. Note that the controls do interact.

For information about Stratix V transceiver PHY IP control and status registers, refer to the *Altera Transceiver PHY IP Core User Guide*

Lock Status Registers

The following registers show the lock status of the high speed I/O and RX PCS. `RX_AGGREGATE[0]` aggregates the status of the individual RX PCS channels. When set to 1, the RX PCS is operating normally. When set to 0, the other information indicates the cause. Table 3–21 describes the lock status registers.

Table 3–21. Lock Status Registers (Part 1 of 2)

Address	Name	Bit	Description	HW Reset Value	Access
0x010	IO_LOCKS	[31:25] (100 GbE)	Reserved.	0x7f	R
		[24:22] (100 GbE)	When asserted, indicates that the corresponding TX PLL is locked.	0x0	R
		[21:12] (100 GbE)	Reserved.	0x3ff	R
		[11:2] (100 GbE)	When asserted, indicates that the corresponding RX CDR locked. The lowest bit corresponds to lane 0 and so forth.	0x0	R
		[31:7] (40 GbE)	Reserved.	0x1ffffff	R
		[6] (40 GbE)	When asserted, indicates that the corresponding TX PLL is locked.	1'b0	R
		[5:2] (40 GbE)	When asserted, indicates that the corresponding RX CDR locked. The lowest bit corresponds to lane 0 and so forth.	0x0	R
		[1]	When asserted, indicates that the TX interface is ready.	1'b0	R
		[0]	When asserted, indicates that the RX interface is ready.	1'b0	R
0x011	LOCKED_TIME	[31:0]	Counts the RX continuous up time in seconds. The counter will roll over in approximately 126 years.	0x00000000	R
0x012	WORD_LOCKS	[19:0]	When asserted, indicates that the physical channel has identified 66 bit block boundaries in the serial data stream.	0x00000000	R
0x013	AM_LOCKS	[19:0]	When asserted, indicates that the physical channel has identified virtual lane alignment markers in the data stream.	0x00000000	R

Table 3–21. Lock Status Registers (Part 2 of 2)

0x014	RX_AGGREGATE	[4]	When asserted, indicates a change in PCS-VLANE permutation. This status bit clears on read.	1'b0	R
		[3]	When asserted, indicates a change in lanes deskewed status. This status bit clears on read.	1'b0	R
		[2]	When asserted, indicates a change in PCS-VLANE tag drop position. This status bit clears on read.	1'b0	R
		[1]	When asserted, indicates that all lanes are locked and lane-to-lane deskew is complete so that the 40-100GbE IP core is in operating normally.	1'b0	R
		[0]	When asserted, indicates that all lanes are word and alignment marker locked.	1b'0	R

Bit Error Flag Registers

Bit errors occur naturally from time to time on high speed serial links. The higher level Ethernet protocol includes mechanisms to respond to errors and tally them appropriately. These lower level flags are useful for tracking errors in a physical link and computing error rates. Table 3–22 describes the Bit Error Flag registers.

Table 3–22. Bit Error Flag Registers

Address	Name	Bit	Description	HW Reset Value	Access
0x015	FRAMING_ERR	[31]	When asserted, indicates that a framing error has occurred on any lane. This status bit clears on read.	1'b0	R
		[19:0]	When asserted, indicates that a framing error has occurred on the corresponding physical lane. This status bit clears on read.	0x00000	R
0x016	BIP_ERR	[19:0]	When asserted, indicates that a BIP (lane parity) error has occurred on the corresponding physical lane. This status bit clears on read.	0x00000	R

PCS Hardware Error Registers

Table 3–23 describes the PCS hardware error register.

Table 3–23. PCS Hardware Error Register

Address	Name	Bit	Description	HW Reset Value	Access
0x017	PCS_HW_ERR	[8]	When asserted, indicates a RX phase compensation error.	1'b0	R
		[7]	When asserted, indicates a TX phase compensation error.	1'b0	R
		[6]	When asserted, indicates a deskew failure suggesting a problem with the remote transmitter.	1'b0	R
		[5]	When asserted, indicates a parity error in the RX lanes (RXL) section.	1'b0	R
		[4]	When asserted, indicates a RX deskew FIFO buffer underflow error.	1'b0	R
		[3]	When asserted, indicates a RX deskew FIFO buffer overflow error.	1'b0	R
		[2]	When asserted, indicates a parity error in the TX lanes (TXL) section.	1'b0	R
		[1]	When asserted, indicates a TX deskew FIFO buffer underflow error.	1'b0	R
		[0]	When asserted, indicates a TX deskew FIFO buffer overflow error.	1'b0	R

BER Monitor Registers

Table 3–24 describes the BER monitor registers.

Table 3–24. BER Monitor Registers

Address	Name	Bit	Description	HW Reset Value	Access
0x018	BER_MONITOR	[0]	When asserted by the BER monitor block, this register indicates the PCS is recording a high BER.	1'b0	R
		[1]	This register enables the BER monitor.	1'b0	RW

Test Mode Registers

Table 3–25 describes the test mode registers.

Table 3–25. Test Mode Registers

Address	Name	Bit	Description	HW Reset Value	Access
0x019	TEST_MODE	[0]	This register enables TX test mode.	1'b0	RW
		[1]	This register enables RX test mode.	1'b0	RW
		[2]	This register enables the Clr_test_pattern_counter.	1'b0	RW

Test Pattern Counter Registers

Table 3–26 describes the test pattern counter register. Unlike other statistics counters, the RX TEST_PATTERN_COUNTER is 32 bits and saturates.

Table 3–26. Test Pattern Counter Register

Address	Name	Bit	Description	HW Reset Value	Access
0x1a	TEST_PATTERN_COUNTER	[31:0]	This register enables the test pattern error counter. The counter saturates at 0xffffffff.	32'b0	R

Link Fault Signaling Registers

Table 3–27 describes the Link Fault Sequence enable register.

Table 3–27. Link Fault Sequence Enable Register

Address	Name	Bit	Description	HW Reset Value	Access
0x1b	Enable Link Fault Sequence	[0]	When asserted, the PCS generates remote fault sequence if conditions are met.	1'b0	RW

Table 3–28 describes the Link Fault Signaling configuration register.

Table 3–28. Link Fault Signaling Configuration Register

Address	Name	Bit	Description	HW Reset Value	Access
0x122	MAC/RS link fault sequence configuration	[1:0]	The local fault configuration register. Possible configurations include: ■ 2'b01: sends idle frames when local fault is received ■ 2'b11: sends remote fault sequence when local fault is received ■ 2'bx0: sends normal traffic when local fault is received	2'b00	RW
		[3:2]	The remote fault configuration register. Possible configurations include: ■ 2'b01: sends idle frames when remote fault is received ■ 2'b11: sends remote fault sequence when remote fault is received ■ 2'bx0: sends normal traffic when remote fault is received	2'b00	RW
		[4]	The local fault status register.	1'b0	R
		[5]	The remote fault status register.	1'b0	R

MAC and PHY Reset Registers

The following registers control the 40-100GbE MAC and PHY resets. Writing a 1'b1 to any of the reset registers initiates the reset sequence.

Table 3–29 describes the MAC reset register.

Table 3–29. MAC Reset Register

Address	Name	Bit	Description	HW Reset Value	Access
0x121	MAC Reset	[0]	The MAC RX reset register.	1'b0	RW
		[1]	The MAC TX reset register.	1'b0	RW

Table 3–30 describes the MAC reset register

Table 3–30. PHY Reset Registers

Address	Name	Bit	Description	HW Reset Value	Access
0x01d	PHY reset	[0]	The PCS RX reset register.	1'b0	RW
		[1]	The PCS TX reset register.	1'b0	RW
		[2]	The PMA reset register.	1'b0	RW

PCS-VLANE Registers

When the RX PCS is properly locked, the PCS-VLANE registers contain a permutation of the numbers [0–19] stored in 5-bit values. You may need this information to isolate a problem with the remote transmitter. Any problem with the permutation, for example two lanes with the same number, stops the reception of valid data. Table 3–31 describes the PCS-VLANE registers.

Table 3–31. PCS-VLANE Registers (Part 1 of 2)

Address	Name	Bit	Description	HW Reset Value	Access
0x020	PCS-VLANE0	[4:0]	Virtual index for physical lane 0	5h'00	R
		[9:5]	Virtual index for physical lane 1	5h'00	R
		[14:10]	Virtual index for physical lane 2	5h'00	R
		[19:15]	Virtual index for physical lane 3	5h'00	R
		[24:20]	Virtual index for physical lane 4	5h'00	R
		[29:25]	Virtual index for physical lane 5	5h'00	R
0x021	PCS-VLANE1	[4:0]	Virtual index for physical lane 6	5h'00	R
		[9:5]	Virtual index for physical lane 7	5h'00	R
		[14:10]	Virtual index for physical lane 8	5h'00	R
		[19:15]	Virtual index for physical lane 9	5h'00	R
		[24:20]	Virtual index for physical lane 10	5h'00	R
		[29:25]	Virtual index for physical lane 11	5h'00	R

Table 3–31. PCS-VLANE Registers (Part 2 of 2)

0x022	PCS-VLANE2	[4:0]	Virtual index for physical lane 12	5h'00	R
		[9:5]	Virtual index for physical lane 13	5h'00	R
		[14:10]	Virtual index for physical lane 14	5h'00	R
		[19:15]	Virtual index for physical lane 15	5h'00	R
		[24:20]	Virtual index for physical lane 16	5h'00	R
		[29:25]	Virtual index for physical lane 17	5h'00	R
0x023	PCS-VLANE3	[4:0]	Virtual index for physical lane 18	5h'00	R
		[9:5]	Virtual index for physical lane 19	5h'00	R

PRBS Registers

The PRBS operates on a per virtual lane basis. The PRBS streams are bit interleaved to form 10 Gbps lanes. PRBS transmissions are unframed. There is some deviation among manufacturers regarding the exact PRBS implementation on high rate Ethernet equipment. If the PRBS is working properly in loopback but consistently wrong with a remote device, Altera recommends that you confirm that the remote device is operating as described here. Table 3–32 describes the PRBS registers.

Please contact your local Altera sales representative (www.altera.com/corporate/contact/con-index.html) if you require different PRBS functionality or polynomials.

Table 3–32. PRBS Registers

Address	Name	Bit	Description	HW Reset Value	Access
0x030	PRBS_CTRL	[4]	When set to 1, selects PRBS-9 for the RX lane. When set to 0, selects PRBS-31 for the RX lane.	1'b0	RW
		[3]	When set to 1, selects PRBS-9 for the TX lane. When set to 0, selects PRBS-31 for the TX lane.	1'b0	RW
		[2]	When set to 1, enables the RX PRBS checker.	1'b0	RW
		[1]	When set to 1, enables the TX PRBS transmitter.	1'b0	RW
0x031	PRBS_ERR_INJ	[19:0]	When set to 1, Injects an error in the corresponding lane. This register is rising edge triggered. Write 0 to clear.	0x00000	RW
0x032	PRBS_EFLAGS	[19:0]	When set to 1, indicates a PRBS error in the corresponding PCS-VLANE. Clears on read.	0x00000	R

MAC Configuration and Filter Registers

Table 3–33 describes the MAC configuration and filter registers.

Table 3–33. General Control Register (Part 1 of 2)

Address	Name	Bit	Description	HW Reset Value	Access
0x100	MAC_VERSION	[31:0]	MegaCore function revision.	0x00DF1200 (40GbE) 0x00DD1200 (100GbE)	R
0x101	SCRATCH_MAC	[31:0]	Scratch register available for testing.	0x00000000	RW

Table 3–33. General Control Register (Part 2 of 2)

0x102	MAC_CMD_config	[4]	When set to 1, the transmit CRC FIFO is purged. Used for hardware diagnostics. Not required for normal operation. To clear, write a 0.	1'b0	RW
		[3]	When set to 1, the statistics counters are reset. This register is self-clearing.	1'b0	RW
		[2]	When set to 1, statistics collection is paused. The underlying counters continue to operate, but the readable values reflect a snapshot at the time the pause flag was activated. Write a 0 to release.	1'b0	RW
		[1:0]	Allows you to override normal transmission for hardware diagnostic purposes: The following patterns are defined: ■ 2'b00 / 2'b10–normal operation (default) ■ 2'b01–Send a small repeating loop of random content frames ■ 2'b11–Send idles	2'b00	RW
0x103	RX_FILTER_CTRL	[17:8]	Length limit in 16-byte words, packets of more than this length are discarded and recorded in the too long counter. (40GbE IP core)	0x2800a	RW
		[15:8]	Length limit in 40-byte words, packets of more than this length are discarded and recorded in the too long counter. (100GbE IP core)	0xf00a	RW
		[3]	When set to 1, runt frames are removed regardless of the filtering enable bit 0x013[0].	1b'1	RW
		[2]	When set to 1, the filter discard packets which are not for a matching destinations address.	1b'0	RW
		[1]	When set to 1, the filter discards packets with FCS errors.	1b'1	RW
		[0]	When set to 0 enables filtering. When set to 1, accepts all traffic. However, when bit [3] is set, runts are removed regardless of the value of bit [0].	1b'0	RW

Pause Registers

The pause register provides programmatic access to the pause functionality via the Avalon-MM control and status bus. These registers implement the pause functionality as defined in the *IEEE 802.3ba-2010 100G Ethernet Standard*. You can use the pause signals to reduce traffic in a congested networks.

Table 3–34 describes the pause registers.

Table 3–34. Pause Registers (Part 1 of 2)

Addr	Name	Bit	Description	HW Reset Value	Access

Table 3–34. Pause Registers (Part 2 of 2)

0x110	RECEIVE_PAUSE_STATUS	[16]	When set to 1, indicates that a pause is in progress.	1'b0	R
		[15:0]	The time value for the pause. Reading this field locks the pause source address.	0x140 (40GbE) 0x01A0 (100GbE)	R
0x111	RECEIVE_SOURCE_ADDR_LSB	[31:0]	Received pause source address lsb.	0x00000000	R
0x112	RECEIVE_SOURCE_ADDR_MSB	[31:0]	Received pause source address msb.	0x00000000	R
0x113	RECEIVE_PAUSE_CONTROL	[9]	When set to 1, enables unicast pause receive.	1'b0	RW
		[8]	When set to 1, enables multicast pause receive.	1'b0	RW
		[7:0]	Pause quantum time configuration as follows: ■ 1 pause_quanta = 512 bit times ■ For 40Gbps, 1 pause_quanta is 12.8 ns ■ For 100Gbps, 1 pause_quanta is 5.12 ns ■ pause_quantum_delta = 256 x Tclk/Tpause_quanta For example: 40 Gbps, clk =312.5 MHz, Tclk = 3.2ns, pause_quantum_delta = round(256*(3.2/12.8)) = 64.	0x00000000	RW
0x114	INSERT_PAUSE_CONTROL	[16]	When set to 1, sends a multicast pause request. When set to 0, sends a unicast pause request.	1'b0	RW
		[15:0]	Specifies the pause time. A non-zero value specifies an XON. Zero specifies XOFF.	0x0000	RW
0x115	TX_PAUSE_DST_ADDR_LSB	[31:0]	Destination address lsb.	0x00000000	RW
0x116	TX_PAUSE_DST_ADDR_MSB	[31:0]	Destination address msb.	0x00000000	RW
0x117	INSERT_PAUSE	[31:0]	Any write to this address triggers a pause packet insertion into the TX data stream.	0x00000000	W

MAC Hardware Error Registers

Table 3–35 describes the MAC hardware error registers.

Table 3–35. MAC Hardware Error Registers

Address	Name	Bit	Description	HW Reset Value	Access
0x120	MAC_HW_ERR	[6]	When asserted, indicates a parity error in the DOE storage RAM section.	1'b0	R
		[5]	When asserted, indicates a parity error in the DOE command FIFO buffer section.	1'b0	R
		[4]	When asserted, indicates a DOE_COMMAND FIFO buffer overflow error.	1'b0	R
		[3]	When asserted, indicates a parity error in the TX CRC read-ram (TXC) section.	1'b0	R
		[2]	When asserted, indicates a parity error in the TX CRC write-ram (TXW) section.	1'b0	R
		[1]	When asserted, indicates a parity error in the RX inspector (RXI) section.	1'b0	R
		[0]	When asserted, indicates a TX CRC FIFO buffer overflow error.	1'b0	R

CRC Configuration Registers

Table 3–36 describes the CRC configuration registers.

Table 3–36. CRC Configuration Registers

Address	Name	Bit	Description	HW Reset Value	Access
0x123	CRC_CONFIG	[0]	The TX CRC configuration register. Possible configurations include: ■ 1'b0: enables TX CRC insertion ■ 1'b1: disables TX CRC insertion	1'b0	RW
		[1]	The RX CRC configuration register. Possible configurations include: ■ 1'b0: removes RX CRC ■ 1'b1: retains RX CRC	1'b0	RW

MAC Address Registers

All MAC addresses are stored in 2 words of 24 bits (3 bytes) in natural reading order. For example, an Altera Device ID of 00-07-ed-11-22-33 is stored with the more significant word set to 24'h0007ed and the least significant word set to 24'h112233.

Table 3–37 describes the MAC address registers.

Table 3–37. MAC Address Registers

Address	Name	Bit	Description	HW Reset Value	Access
0x140	MADDR_CTRL	[31]	When set to 1, the source address is Inserted in TX packets.	1'b0	RW
		[30]	When set to 1, the destination address in RX packets is checked.	1'b0	RW
		[29:1]	Reserved.	0x00000000	
		[0]	When set to 1, enables destination address 0x161-0x162 checking.	0x0000	
0x141	SRC_AD_LO	[23:0]	Source address (lower 24 bits).	0x00FF1234	RW
0x142	SRC_AD_HI	[23:0]	Source address (upper 24 bits).	0x000007ED	RW
0x160	DST_AD0_LO	[23:0]	Destination address 0 (lower 24 bits).	0x00000000	RW
0x161	DST_AD0_HI	[23:0]	Destination address 0 (upper 24 bits).	0x00000000	RW
0x162-0x17f	RESERVED	[31:0]	Reserved.	0x00000000	

Statistics Registers

The statistics registers count Ethernet traffic and errors. These registers should never roll over if the link is functioning properly. The statistics registers check the size of frames, which includes the following fields:

- Size of the destination address
- Size of the source address
- Size of the data
- Four bytes of CRC

If the 64-bit statistics counters fill, the counters will roll over to ensure timing closure on the FPGA. To read the value of the of a statistics register without clearing it, first write the MAC_CMD_config to 0x4. To read the value of a statistics register and reset the counters, first write the MAC_CMD_config to 0xC. For more information about the MAC_CMD_config register refer to "General Control Register" on page 3–46.

Table 3–38 describes the statistic registers. The initial value after reset is 0 for all statistics registers.

Table 3–38. Statistics Registers (Part 1 of 5)

Address	Name	Description	Access
TX Statistics Counters			
0x200	CNTR_TX_64B_LO	Number of 64-byte transmitted frames, including the CRC field but excluding the preamble and SFD bytes (lower 32 bits)	RC
0x201	CNTR_TX_64B_HI	Number of 64-byte transmitted frames (upper 32 bits), including the CRC field but excluding the preamble and SFD bytes (upper 32 bits)	RC

Table 3–38. Statistics Registers (Part 2 of 5)

0x202	CNTR_TX_65to127B_LO	Number of transmitted frames between 65–127 bytes (lower 32 bits)	RC
0x203	CNTR_TX_65to127B_HI	Number of transmitted frames between 65–127 bytes (upper 32 bits)	RC
0x204	CNTR_TX_128to255B_LO	Number of transmitted frames between 128–255 bytes (lower 32 bits)	RC
0x205	CNTR_TX_128to255B_HI	Number of transmitted frames between 128–255 bytes (upper 32 bits)	RC
0x206	CNTR_TX_256to511B_LO	Number of transmitted frames between 256–511 bytes (lower 32 bits)	RC
0x207	CNTR_TX_256to511B_HI	Number of transmitted frames between 256–511 bytes (upper 32 bits)	RC
0x208	CNTR_TX_512to1023B_LO	Number of transmitted frames between 512–1023 bytes (lower 32 bits)	RC
0x209	CNTR_TX_512to1023B_HI	Number of transmitted frames between 512–1023 bytes (upper 32 bits)	RC
0x20A	CNTR_TX_1024to1518B_LO	Number of transmitted frames between 1024–1518 bytes (lower 32 bits)	RC
0x20B	CNTR_TX_1024to1518B_HI	Number of transmitted frames between 1024–151 (upper 32 bits)	RC
0x20C	CNTR_TX_1519toMAXB_LO	Number of transmitted frames between 1519 and max size defined in 0x103 (lower 32 bits)	RC
0x20D	CNTR_TX_1519toMAXB_HI	Number of transmitted frames between 1519 and max size defined in 0x103 (upper 32 bits)	RC
0x20E	CNTR_TX_OVERSIZE_LO	Number of oversized frames transmitted (lower 32 bits)	RC
0x20F	CNTR_TX_OVERSIZE_HI	Number of oversized frames transmitted (upper 32 bits)	RC
0x210	CNTR_TX_MCAST_DATA_ERR_LO	Number of errored multicast frames transmitted, excluding control frames (lower 32 bits)	RC
0x211	CNTR_TX_MCAST_DATA_ERR_HI	Number of errored multicast frames transmitted, excluding control frames (upper 32 bits)	RC
0x212	CNTR_TX_MCAST_DATA_OK_LO	Number of valid multicast frames transmitted, excluding control frames (lower 32 bits)	RC
0x213	CNTR_TX_MCAST_DATA_OK_HI	Number of valid multicast frames transmitted, excluding control frames (upper 32 bits)	RC
0x214	CNTR_TX_BCAST_DATA_ERR_LO	Number of errored broadcast frames transmitted, excluding control frames (lower 32 bits)	RC
0x215	CNTR_TX_BCAST_DATA_ERR_HI	Number of errored broadcast frames transmitted, excluding control frames (upper 32 bits)	RC
0x216	CNTR_TX_BCAST_DATA_OK_LO	Number of valid broadcast frames transmitted, excluding control frames (lower 32 bits)	RC
0x217	CNTR_TX_BCAST_DATA_OK_HI	Number of valid broadcast frames transmitted, excluding control frames (upper 32 bits)	RC
0x218	CNTR_TX_UCAST_DATA_ERR_LO	Number of errored unicast frames transmitted, excluding control frames (lower 32 bits)	RC
0x219	CNTR_TX_UCAST_DATA_ERR_HI	Number of errored unicast frames transmitted, excluding control frames (upper 32 bits)	RC

Table 3–38. Statistics Registers (Part 3 of 5)

0x21A	CNTR_TX_UCAST_DATA_OK_LO	Number of valid unicast frames transmitted, excluding control frames (lower 32 bits)	RC
0x21B	CNTR_TX_UCAST_DATA_OK_HI	Number of valid unicast frames transmitted, excluding control frames (upper 32 bits)	RC
0x21C	CNTR_TX_MCAST_CTRL_LO	Number of valid multicast frames transmitted (lower 32 bits)	RC
0x21D	CNTR_TX_MCAST_CTRL_HI	Number of valid multicast frames transmitted (upper 32 bits)	RC
0x21E	CNTR_TX_BCAST_CTRL_LO	Number of valid broadcast frames transmitted (lower 32 bits)	RC
0x21F	CNTR_TX_BCAST_CTRL_HI	Number of valid broadcast frames transmitted (upper 32 bits)	RC
0x220	CNTR_TX_UCAST_CTRL_LO	Number of valid unicast frames transmitted (lower 32 bits)	RC
0x221	CNTR_TX_UCAST_CTRL_HI	Number of valid unicast frames transmitted (upper 32 bits)	RC
0x222	CNTR_TX_PAUSE_LO	Number of valid pause frames transmitted (lower 32 bits)	RC
0x223	CNTR_TX_PAUSE_HI	Number of valid pause frames transmitted (upper 32 bits)	RC
0x224	CNTR_TX_FRAGMENTS_LO	Number of transmitted frames of less than 64 bytes reporting a CRC error (lower 32 bits)	RC
0x225	CNTR_TX_FRAGMENTS_HI	Number of transmitted frames of less than 64 bytes reporting a CRC error (upper 32 bits)	RC
0x226	CNTR_TX_JABBERS_LO	Number of transmitted oversized frames reporting a CRC error (lower 32 bits)	RC
0x227	CNTR_TX_JABBERS_HI	Number of transmitted oversized frames reporting a CRC error (upper 32 bits)	RC
0x228	CNTR_TX_CRCERR_LO	Number of transmitted frames between 64 bytes and the value configured in 0x103 register reporting a CRC error (lower 32 bits)	RC
0x229	CNTR_TX_CRCERR_HI	Number of transmitted frames between 64 bytes and the value configured in 0x103 register reporting a CRC error (upper 32 bits)	RC
TX Packet Statistics			
0x22A	CNTR_TX_ST_LO	Number of transmitted frame starts (lower 32 bits)	RC
0x22B	CNTR_TX_ST_HI	Number of transmitted frame starts (upper 32 bits)	RC
0x22C	CNTR_TX_DB_LO	Number of transmitted data blocks (lower 32 bits)	RC
0x22D	CNTR_TX_DB_HI	Number of transmitted data blocks (upper 32 bits)	RC
RX Statistics Counters			
0x280	CNTR_RX_64B_LO	Number of 64-byte received frames (lower 32 bits), including the CRC field but excluding the preamble and SFD bytes	RC
0x281	CNTR_RX_64B_HI	Number of 64-byte received frames (upper 32 bits), including the CRC field but excluding the preamble and SFD bytes	RC
0x282	CNTR_RX_65to127B_LO	Number of received frames between 65–127 bytes (lower 32 bits)	RC
0x283	CNTR_RX_65to127B_HI	Number of received frames between 65–127 bytes (upper 32 bits)	RC
0x284	CNTR_RX_128to255B_LO	Number of received frames between 128 –255 bytes (lower 32 bits)	RC
0x285	CNTR_RX_128to255B_HI	Number of received frames between 128 –255 bytes (upper 32 bits)	RC

Table 3–38. Statistics Registers (Part 4 of 5)

0x286	CNTR_RX_256to511B_LO	Number of received frames between 256 –511 bytes (lower 32 bits)	RC
0x287	CNTR_RX_256to511B_HI	Number of received frames between 256 –511 bytes (upper 32 bits)	RC
0x288	CNTR_RX_512to1023B_LO	Number of received frames between 512–1023 bytes (lower 32 bits)	RC
0x289	CNTR_RX_512to1023B_HI	Number of received frames between 512 –1023 bytes (upper 32 bits)	RC
0x28A	CNTR_RX_1024to1518B_LO	Number of received frames between 1024–1518 bytes (lower 32 bits)	RC
0x28B	CNTR_RX_1024to1518B_HI	Number of received frames between 1024–51 (upper 32 bits)	RC
0x28C	CNTR_RX_1519toMAXB_LO	Number of received frames between 1519 and the maximum size defined in 0x103 (lower 32 bits)	RC
0x18D	CNTR_RX_1519toMAXB_HI	Number of received frames between 1519 and the maximum size defined in 0x103 (upper 32 bits)	RC
0x28E	CNTR_RX_OVERSIZE_LO	Number of oversized frames received (lower 32 bits)	RC
0x28F	CNTR_RX_OVERSIZE_HI	Number of oversized frames received (upper 32 bits)	RC
0x290	CNTR_RX_MCAST_DATA_ERR_LO	Number of errored multicast frames received, excluding control frames (lower 32 bits)	RC
0x291	CNTR_RX_MCAST_DATA_ERR_HI	Number of errored multicast frames received, excluding control frames (upper 32 bits)	RC
0x292	CNTR_RX_MCAST_DATA_OK_LO	Number of valid multicast frames received, excluding control frames (lower 32 bits)	RC
0x293	CNTR_RX_MCAST_DATA_OK_HI	Number of valid multicast frames received, excluding control frames (upper 32 bits)	RC
0x294	CNTR_RX_BCAST_DATA_ERR_LO	Number of errored broadcast frames received, excluding control frames (lower 32 bits)	RC
0x295	CNTR_RX_BCAST_DATA_ERR_HI	Number of errored broadcast frames received, excluding control frames (upper 32 bits)	RC
0x296	CNTR_RX_BCAST_DATA_OK_LO	Number of valid broadcast frames received, excluding control frames (lower 32 bits)	RC
0x297	CNTR_RX_BCAST_DATA_OK_HI	Number of valid broadcast frames received, excluding control frames (upper 32 bits)	RC
0x298	CNTR_RX_UCAST_DATA_ERR_LO	Number of errored unicast frames received, excluding control frames (lower 32 bits)	RC
0x299	CNTR_RX_UCAST_DATA_ERR_HI	Number of errored unicast frames received, excluding control frames (upper 32 bits)	RC
0x29A	CNTR_RX_UCAST_DATA_OK_LO	Number of valid unicast frames received, excluding control frames (lower 32 bits)	RC
0x29B	CNTR_RX_UCAST_DATA_OK_HI	Number of valid unicast frames received, excluding control frames (upper 32 bits)	RC
0x29C	CNTR_RX_MCAST_CTRL_LO	Number of valid multicast frames received (lower 32 bits)	RC
0x29D	CNTR_RX_MCAST_CTRL_HI	Number of valid multicast frames received (upper 32 bits)	RC
0x29E	CNTR_RX_BCAST_CTRL_LO	Number of valid broadcast frames received (lower 32 bits)	RC
0x29F	CNTR_RX_BCAST_CTRL_HI	Number of valid broadcast frames received (upper 32 bits)	RC

Table 3–38. Statistics Registers (Part 5 of 5)

0x2A0	CNTR_RX_UCAST_CTRL_LO	Number of valid unicast frames received (lower 32 bits)	RC
0x2A1	CNTR_RX_UCAST_CTRL_HI	Number of valid unicast frames received (upper 32 bits)	RC
0x2A2	CNTR_RX_PAUSE_LO	Number of valid pause frames received (lower 32 bits)	RC
0x2A3	CNTR_RX_PAUSE_HI	Number of valid pause frames received (upper 32 bits)	RC
0x2A4	CNTR_RX_FRAGMENTS_LO	Number of received frames less than 64 bytes and reporting a CRC error (lower 32 bits)	RC
0x2A5	CNTR_RX_FRAGMENTS_HI	Number of received frames less than 64 bytes and reporting a CRC error (upper 32 bits)	RC
0x2A6	CNTR_RX_JABBERS_LO	Number of received oversized frames reporting a CRC error (lower 32 bits)	RC
0x2A7	CNTR_RX_JABBERS_HI	Number of received oversized frames reporting a CRC error (upper 32 bits)	RC
0x2A8	CNTR_RX_CRCERR_LO	Number of received frames between the length of 64 and the value configured in 0x103 register with CRC error (lower 32 bits)	RC
0x2A9	CNTR_RX_CRCERR_HI	Number of received frames between the length of 64 and the value configured in 0x103 register with CRC error (upper 32 bits)	RC
RX Packet Statistics			
0x2AA	CNTR_RX_ST_LO	Number of received frame starts (lower 32 bits)	RC
0x2AB	CNTR_RX_ST_HI	Number of received frame starts (upper 32 bits)	RC
0x2AC	CNTR_RX_DB_LO	Number of received data blocks (lower 32 bits)	RC
0x2AD	CNTR_RX_DB_HI	Number of received data blocks (upper 32 bits)	RC
0x2AE	CNTR_RXF_ST_LO	Number of accepted frame starts (lower 32 bits)	RC
0x2AF	CNTR_RXF_ST_HI	Number of accepted frame starts (upper 32 bits)	RC
0x2B0	CNTR_RXF_DB_LO	Number of accepted data blocks (lower 32 bits)	RC
0x2B1	CNTR_RXF_DB_HI	Number of accepted data blocks (upper 32 bits)	RC
0x2B2	CNTR_RXF_LONG	Number of packets aborted due to excessive length.	
0x2B3	Reserved		
0x2B4	CNTR_RX_EBLK	Number of received blocks with XL/CGMII errors	RC
0x2B5	Reserved		
0x2B6	CNTR_RX_RUNT	Number of received runt packets	RC
0x2B7	Reserved		
0x2B8	CNTR_RX_FCS	Number of received packets with FCS errors	RC

The statistics counters module is a synthesis option. The statistics status bit output vectors are provided whether or not the statistics counters module option is selected. Table 3–39 describes the statistics counters increment vectors.

The increment vectors are brought to the top level as output ports and function as input ports to the control and status registers (CSR). The counters are implemented inside the CSR. The parameter ENABLE_STATISTICS_CNTR is passed to mac_csr and phy_csr to setup this option. When the ENABLE_STATISTICS_CNTR parameter is set to 1, the counters are implemented in the CSR. When the ENABLE_STATISTICS_CNTR parameter is set to 0, the counters are not implemented in the CSR, and read access to the counters returns read data equal to 0.

Table 3–39. Statistics Counters Increment Vectors (Part 1 of 2)

Name	I/O	Description
TX Statistics Counters Increment Vectors		
tx_inc_64	Output	Asserted for one cycle when a 64-byte TX frame is transmitted.
tx_inc_127	Output	Asserted for one cycle when a 127-byte TX frame is transmitted.
tx_inc_255	Output	Asserted for one cycle when a 255-byte TX frame is transmitted.
tx_inc_511	Output	Asserted for one cycle when a 511-byte TX frame is transmitted.
tx_inc_1023	Output	Asserted for one cycle when a 1023-byte TX frame is transmitted.
tx_inc_1518	Output	Asserted for one cycle when a 1518-byte TX frame is transmitted.
tx_inc_max	Output	Asserted for one cycle when a maximum-size TX frame is transmitted.
tx_inc_over	Output	Asserted for one cycle when an oversized TX frame is transmitted.
tx_inc_mcast_data_err	Output	Asserted for one cycle when an errored multicast TX frame, excluding control frames, is transmitted.
tx_inc_mcast_data_ok	Output	Asserted for one cycle when valid a multicast TX frame, excluding control frames, is transmitted.
tx_inc_bcast_data_err	Output	Asserted for one cycle when an errored broadcast TX frame, excluding control frames, is transmitted.
tx_inc_bcast_data_ok	Output	Asserted for one cycle when a valid broadcast TX frame, excluding control frames, is transmitted.
tx_inc_ucast_data_err	Output	Asserted for one cycle when an errored unicast TX frame, excluding control frames, is transmitted.
tx_inc_ucast_data_ok	Output	Asserted for one cycle when a valid unicast TX frame, excluding control frames, is transmitted.
tx_inc_mcast_ctrl	Output	Asserted for one cycle when a valid multicast TX frame is transmitted.
tx_inc_bcast_ctrl	Output	Asserted for one cycle when a valid broadcast TX frame is transmitted.
tx_inc_ucast_ctrl	Output	Asserted for one cycle when a valid unicast TX frame is transmitted.
tx_inc_pause	Output	Asserted for one cycle when a valid pause TX frame is transmitted.
tx_inc_fcs_err	Output	Asserted for one cycle when a TX packet with FCS errors is transmitted.
tx_inc_fragment	Output	Asserted for one cycle when a TX frame less than 64 bytes and reporting a CRC error is transmitted.
tx_inc_jabber	Output	Asserted for one cycle when an oversized TX frame reporting a CRC error is transmitted.
tx_inc_sizeok_fcserr	Output	Asserted for one cycle when a valid TX frame with FCS errors is transmitted.
RX Statistics Counters Increment Vectors		
rx_inc_runt	Output	Asserted for one cycle when an RX runt packet is received.
rx_inc_64	Output	Asserted for one cycle when a 64-byte RX frame is received.

Table 3–39. Statistics Counters Increment Vectors (Part 2 of 2)

Name	I/O	Description
rx_inc_127	Output	Asserted for one cycle when a 127-byte RX frame is received.
rx_inc_255	Output	Asserted for one cycle when a 255-byte RX frame is received.
rx_inc_511	Output	Asserted for one cycle when a 511-byte RX frame is received.
rx_inc_1023	Output	Asserted for one cycle when a 1023-byte RX frame is received.
rx_inc_1518	Output	Asserted for one cycle when a 1518-byte RX frame is received.
rx_inc_max	Output	Asserted for one cycle when a maximum-size RX frame is received.
rx_inc_over	Output	Asserted for one cycle when an oversized RX frame is received.
rx_inc_mcast_data_err	Output	Asserted for one cycle when an errored multicast RX frame, excluding control frames, is received.
rx_inc_mcast_data_ok	Output	Asserted for one cycle when valid a multicast RX frame, excluding control frames, is received.
rx_inc_bcast_data_err	Output	Asserted for one cycle when an errored broadcast RX frame, excluding control frames, is received.
rx_inc_bcast_data_ok	Output	Asserted for one cycle when a valid broadcast RX frame, excluding control frames, is received.
rx_inc_ucast_data_err	Output	Asserted for one cycle when an errored unicast RX frame, excluding control frames, is received.
rx_inc_ucast_data_ok	Output	Asserted for one cycle when a valid unicast RX frame, excluding control frames, is received.
rx_inc_mcast_ctrl	Output	Asserted for one cycle when a valid multicast RX frame is received.
rx_inc_bcast_ctrl	Output	Asserted for one cycle when a valid broadcast RX frame is received.
rx_inc_ucast_ctrl	Output	Asserted for one cycle when a valid unicast RX frame is received.
rx_inc_pause	Output	Asserted for one cycle when valid RX pause frames are received.
rx_inc_fcs_err	Output	Asserted for one cycle when a RX packet with FCS errors is received.
rx_inc_fragment	Output	Asserted for one cycle when a RX frame less than 64 bytes and reporting a CRC error is received.
rx_inc_jabber	Output	Asserted for one cycle when an oversized RX frame reporting a CRC error is received.
rx_inc_sizeok_fcserr	Output	Asserted for one cycle when a valid RX frame with FCS errors is received.

40-100GbE Example Design Registers

This section defines registers that are included in the 100GbE example design and are not a part of the 40-100GbE IP core.

PMD Registers

The PMD control and status registers allows you to turn on and determine the status of the PMD device. Table 3–40 describes the PMD registers.

Table 3–40. PMD Control and Status Registers

Address	Name	Bits	Description	HW Reset Value	Access
0x400	PMD_VERSION	[31:0]	PMD revision. The character string is “OPTs.”	0x4F505473	R

Table 3–40. PMD Control and Status Registers

0x401	SCRATCH_PMD	[31:0]	Scratch register available for testing.	0x00000000	RW
0x402	PMD_CMD_CONFIG	[0]	Writing a 1 enables the PMD.	0x00000000	RW
0x403	CMD_STATUS	[6]	PMD global alarm.	1b'0	R
		[5]	Programmable alarm 3. Defaults to module ready.	1b'0	R
		[4]	Programmable alarm 2. Defaults to high power mode on.	1b'0	R
		[3]	Programmable alarm 1. Defaults to RX CDR lock.	1b'0	R
		[2]	RX side signal detected.	1'b0	R
		[1]	When asserted, the PMD module is physically plugged in. When deasserted, it is offline.	1b'1	R
		[0]	When 1, the PMD is resetting. When 0, the PMD is available.	1'b1	R

For more information about the PMD devices, refer to the *Next Gen PMD CFP MSA Baseline Specifications* available on the CFP Multi-Source Agreement website (www.cfp-msa.org).

MDIO Registers

The MDIO is a serial bus interface for the Ethernet. As Figure A–1 on page A–1 illustrates, client logic drives the MDIO module when an MDIO serial interface controls the external PMD or CFP device. This serial interface includes two wires, an MDC clock driven by the MAC and a bidirectional data line which can be driven by up to 31 PHY slave devices. The minimum clock frequency is 2.5 MHz. Refer to the specification of your PMD or CFP device for additional information.

Table 3–41 describes the MDIO registers. These registers are typically used to communicate with an external optical module such as a PMD 100G optical module.

Table 3–41. MDIO Registers

Address	Name	Bits	Description	HW Reset Value	Access
0x410	MDIO_WDATA	[15:0]	Data to be written.	0x0000	RW
0x411	MDIO_RDATA	[31]	Link is busy.	1b'0	R
		[15:0]	Result of previous read.	0x0000	R
0x412	MDIO_ADDR	[14:10]	PHY address driven to external module.	5'h00	RW
		[9:5]	PHY address to access.	5'h00	RW
		[4:0]	DEV address.	5'h01	RW
ox413	MDIO_CMD	[3]	Address of the register to be written.	1b'0	RW
		[2]	Write signal.	1b'0	RW
		[1]	Address of the register to read. This address post increments.	1b'0	RW
		[0]	Read signal.	1b'0	RW

MDIO feature support is currently lagging datapath support in the commercially available PMD modules. The MDIO ports on some PMDs may not function properly.

2-Wire Serial Registers

Several optical modules use the 2-wire serial interface protocol whose advantage is that it requires limited resources. This interface uses two, bidirectional open-drain lines, a serial data line (SDA) and a serial clock (SCL). The master node drives the clock and addresses slaves. Data is sent most significant bit first. The slave node receives the clock and address. Unlike the MDIO protocol, the 2-wire serial interface can include multiple masters. In addition, the master and slave roles may be changed after a STOP message is sent. The interface has an eight-bit address space, so a maximum of 255 nodes can communicate. The interface runs at 400 KHz. Table 3–42 describes the 2-wire serial registers. Refer to the datasheet for your 2-wire device for more information.

This 2-wire serial does not currently support the specialized CRC-8 check or multibyte transaction modes.

Table 3–42. 2-Wire Serial Registers

Address	Name	Bits	Description	HW Reset Value	Access
0x420	2WS_WDATA	[7:0]	Data to be written.	0x00	RW
0x421	2WS_RDATA	[31]	When asserted, the link is busy.	1b'0	R
		[30]	When asserted, indicates that the slave failed to acknowledge during the previous operation.	1b'0	R
		[7:0]	Result of previous read.	0x00	R
0x422	2WS_ADDR	[15:8]	Slave address.	0xA0	RW
		[7:0]	Memory address.	0x00	RW
0x423	2WS_CMD	[1]	When asserted, indicates a write command.	0x0	RW
		[0]	When asserted, indicates a read.	0x0	RW

Ethernet Glossary

Table 3–43 provides definitions for some terms associated with the Ethernet protocol.

Table 3–43. Ethernet Glossary (Part 1 of 2)

Term	Definition
BIP	Bit Interleaved Parity. A diagonal parity field which is carried in the periodic alignment markers on each virtual lane, allowing isolation of a bit error to a physical channel.
CAUI	100 gigabit attachment unit interface. (C is the symbol in Roman Numerals for 100). This is an electrical interface that which is based on a 10-lane interface with a bandwidth of 10 Gbps per lane. (In this implementation, the PMA multiplexes the 20 PCS lanes into 10 physical lanes.
CGMII	100 gigabit media independent interface. (C is the symbol in Roman Numerals for 100). This is the byte-oriented interface protocol that connects the PCS and MAC.
DIC	Deficit Idle Counter. A rule for inserting and deleting idles as necessary to maintain the average IPG. The alternative is to always insert idles which could lead to reduced bandwidth.
FCS	Frame Check Sequence. A CRC-32 with bit reordering and inversion.
Frame	Ethernet formatted packet. A frame consists of a start delimiter byte, a 7 byte preamble, variable length data, 4-byte FCS, and an end delimiter byte.

Table 3–43. Ethernet Glossary (Part 2 of 2)

Term	Definition
IPG	Inter Packet Gap. Includes the end of frame delimiter and subsequent idle bytes up to, but not including the next start of frame delimiter. The protocol requires an average gap of 12 bytes.
MAC	Media Access Control. Formats a user packet stream into proper Ethernet frames for delivery to the PCS. The MAC generates the FCS and checks and maintains the IPG.
MII	Media Independent Interface. The byte-oriented protocol used by the PCS. Sometimes distinguished with roman numerals, XGMII (10), XLGMII (40), CGMII (100).
Octet	Byte. Note that Ethernet specifications primarily use least significant bit first ordering which is opposite from the default behavior of most contemporary CAD tools.
PCS	Physical Coding Sublayer. Presents the underlying hardware as a byte-oriented communication channel.
XLAUI	40 gigabit attachment unit interface. (XL is the symbol in Roman Numerals for 40). This is an electrical interface that which is based on a 4-lane interface with a bandwidth of 10 Gbps per lane.
XLGMII	40 gigabit media independent interface. (XL is the symbol in Roman Numerals for 40). This is the byte-oriented interface protocol that connects the PCS and MAC.

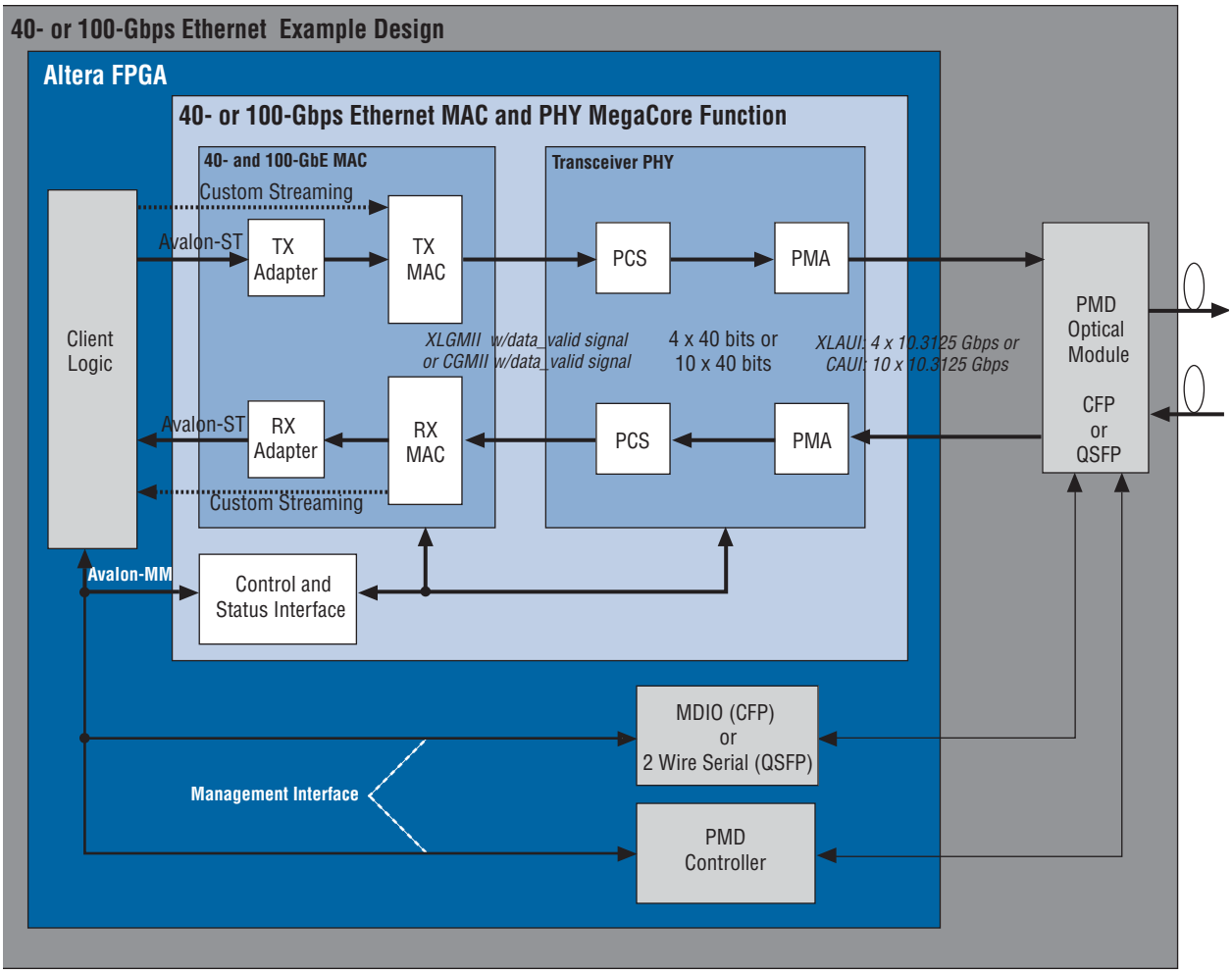
4. Debugging the 40GbE and 100GbE Link

If you are experiencing difficulties bringing up your 40-100GbE IP core link in hardware, Altera suggests that you begin debugging at the most basic level, with word lock. Then, consider higher level issues. The following steps should help you identify and resolve common problems that occur when bringing up a 40-100GbE IP core link:

1. Establish word lock—The RX lanes should be able to achieve word lock even in the presence of extreme bit error rates. If unable to word lock, check the transceiver clocking and data rate configuration. Check for cabling errors such as the reversal of the TX and RX lanes. Check the clock frequency monitors.
2. Establish the alignment marker lock—Virtual lane alignment marker lock requires a moderate quality transceiver connection. If the lock is completely absent, recheck the alignment marker period. If the lock is intermittent, recheck the transceiver physical connection and analog settings.
3. Establish lane integrity—When operating properly, the lanes should not experience bit errors at a rate greater than roughly one per hour per day. Bit errors within data packets are identified as FCS errors. Bit errors in control information including idles generally cause errors in XL/CGMII decoding. The bit interleaved parity (BIP) mechanism is a diagonal parity computation that enables tracing a protocol error back to a physical lane.
4. Verify packet traffic—The protocol includes automatic lane reordering so the higher levels should follow the PCS. If the PCS is locked, but higher level traffic is corrupted, there may be a problem with the remote transmitter virtual lane tags.
5. Tuning—You can adjust analog parameters to minimize any remaining bit error rate. Idle traffic is representative for analog purposes. For more information about the analog parameters for Stratix IV refer to the "Stratix IV Transceiver Analog Settings Registers" on page 3–40 or for Stratix V refer to the *Altera Transceiver PHY IP Core User Guide*.

Figure A–1 provides a high-level block diagram for the 40- or 100-Gbps Ethernet example design.

Figure A–1. High-Level System Overview Example



Notes to Figure A–1:

(1) CFP refers to 100 Gigabit Small Form Factor Pluggable.
(2) QSFP refers to Quad Small Form Factor Pluggable.
(3) MDIO refers to Management Data Input/Output.

Altera's example design includes either the 40GbE or 100GbE IP core. As Figure A–1 illustrates, client logic connects to the TX and RX Ethernet MAC. The TX and RX logic in the MAC includes an optional adapter which is available for both the 40GbE and 100GbE IP cores. When you use the optional adapter, the start of packet (SOP) is always in the most significant word (64 bits) of the bus, simplifying the interpretation

of incoming data. When the configuration includes adapters, the interface between the client logic and TX and RX FIFOs uses the Avalon-ST protocol. Without adapters, the interface between client logic and the TX and RX MAC is a custom streaming interface. The bandwidth for the two versions is the same; however, the version without adapters achieves this bandwidth with a narrower bus, because it does not restrict the SOP to be in the most significant word of the bus; instead, the SOP must simply be in the MSB of any word. The penalty for the less restrictive positioning of the SOP is a datastream which is more difficult to interpret.

As Figure A–1 illustrates, the interface between the MAC and PHY modules of the IP core is XLGMII for the 40GbE IP core and CGMII for the 100GbE IP core. The interface between the PHY and external physical medium dependent (PMD) optical module or other device is XLAUI for the 40GbE IP core and CAUI for the 100GbE IP core, providing a bandwidth of either 4 or 10 × 10.3125 Gbps.

An Avalon-MM control and status (management) interface provides access to the MAC and PHY registers in the IP core and also controls the MDIO, 2-wire serial, and the PMD controllers on the PCB.

For more information about the Avalon-MM and Avalon-ST protocols, including timing diagrams, refer to the *Avalon Interface Specifications*.

B. Updating the PMA

There are six PHY IP cores included for the 40-100GbE IP core: Stratix IV RX only; Stratix IV TX only; Stratix IV duplex (RX and TX); Stratix V RX only; Stratix V TX only; and Stratix V duplex (RX and TX). Each PHY IP core is accessible by navigating through the **alt_eth40g_12.0.zip** file or the **alt_eth100g_12.0.zip** file. Table B–1 shows the directory structures for locating the PHY IP cores.

Table B–1. Directory Structures of PHY IP Cores in 40-100GbE IP Core Release Packages

Subdirectory	Subdirectory	Subdirectory\PHY File
alt_eth40g_12.0.zip		
alt_eth_40g/quartus_synth\rtl_src\phy\	pma_siv	alt_e40_e4x10\alt_e40_e4x10.v
		alt_e40_rx_e4x10\alt_e40_rx_e4x10.v
		alt_e40_tx_e4x10\alt_e40_tx_e4x10.v
	pma_sv	alt_e40_e4x10\alt_e40_e4x10.v
		alt_e40_rx_e4x10\alt_e40_rx_e4x10.v
		alt_e40_tx_e4x10\alt_e40_tx_e4x10.v
alt_eth100g_12.0.zip		
alt_eth_100g/quartus_synth\rtl_src\phy\	pma_siv	alt_e100_e10x10\alt_e100_e10x10.v
		alt_e100_rx_e10x10\alt_e100_rx_e10x10.v
		alt_e100_tx_e10x10\alt_e100_tx_e10x10.v
	pma_sv	alt_e100_e10x10\alt_e100_e10x10.v
		alt_e100_rx_e10x10\alt_e100_rx_e10x10.v
		alt_e100_tx_e10x10\alt_e100_tx_e10x10.v

To change the PMA from the default PMA settings or to update the PMA for future design releases, complete the following steps:

1. Start the MegaWizard™ Plug-In Manager from Quartus II (on the Tools menu, click **MegaWizard Plug-In Manager**).
2. Choose **Edit an existing custom MegaFunction variation** and click **Next**.
3. Navigate to the main Verilog file for the appropriate PHY IP core (for example, `quartus_synth\rtl_src\phy\pma_sv\alt_e100_e10x10\alt_e100_e10x10.v`).
4. Click **Next**, make any desired modifications, and then click **Finish**.

The default PMA setting uses the clock multiplier unit (CMU) PLL, not the auxiliary transmit (ATX) PLL.

Integrated Ethernet solutions include PHY IP cores generated by the MegaWizard Plug-In Manager. Table B–1 shows the header comments from a PHY IP core file generated by the MegaWizard Plug-In Manager.

Example B–1. PHY IP Core File Generated from the MegaWizard Plug-In Manager

```
// megafunction wizard: %ALTGX%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: alt4gxb

// ============================================================
// File Name: alt_e100_e10x10.v
// Megafunction Name(s):
//                      alt4gxb
// Simulation Library File(s):
//                      stratixiv_hssi
// ============================================================
// ************************************************************
// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
//
// 12.0 Build 176 05/18/2012 SJ Full Version
// ************************************************************
```

In some cases, the header comments are appended by generic comment lines; if you regenerate the PHY IP cores, the MegaWizard Plug-In Manager will not recognize the files. Remove any generic comments from the PHY IP core so the file begins with comments shown in Table B–1. The MegaWizard Plug-In Manager will recognize the updated files and complete the regeneration process.

Simulation file lists might need to be updated when upgrading to a newer version of the Quartus II software. Refer to the *Altera Transceiver PHY IP Core User Guide* for more information on simulating the IP core.

C. Address Map Changes for 12.0

For improved organization and clarity, version 12.0 of the 40- and 100-Gbps MAC and PHY MegaCore Function has revised the address map and renamed a few registers. Table C–1 lists the current addresses and register names and the previous address and names if the name has changed. These address map changes apply to both the Stratix IV and Stratix V devices.

Table C–1. Address Map and Register Name Changes for 12.0

Current 40/100GbE Address	Current Name	Previous Address	Previous Name
40-100GbE IP Core Registers			
0x017	PCS_HW_ERR		HW_ERR
0x018	BER_MONITOR		Reserved
0x019	TEST_MODE		Reserved
0x01a	TEST_PATTERN_COUNTER		Reserved
0x01b	Enable Link Fault sequence		Reserved
0x01d	PHY reset		Reserved
0x120	MAC_HW_ERR		Reserved
0x121	MAC Reset		Reserved
0x122	MAC/RS link fault sequence configuration		Reserved
0x123	CRC_CONFIG		Reserved
0x162-0x17f	Reserved	0x160-0x017f	DST_ADR

D. Address Map Changes for 11.0 SP1

For improved organization and clarity, version 11.0 SP1 of the 40- and 100-Gbps MAC and PHY MegaCore Function has revised the address map and renamed a few registers. Table D–1 lists the current addresses and register names and the previous address and names if the name has changed. These address map changes apply to both the Stratix IV and Stratix V devices.

Table D–1. Address Map and Register Name Changes for 11.0 SP1

Current 40/100GbE Address	Current Name	Previous Address	Previous Name
40-100GbE IP Core Registers			
0x000	PHY Version	0x081	
0x001	SCRATCH_PHY	0x080	
0x002	CLK_TXS	0x082	
0x003	CLK_RXS	0x083	
0x004	CLK_TXC	0x084	
0x005	CLK_RXC	0x085	
0x006	TEMP_SENSE	0x086	
0x007	GX_CTRL1	0x087	GX_CTL1
0x008	GX_CTRL2	0x088	GX_CTL2
0x009	GX_REPLY	0x089	
0x00a-0x00f	Reserved		
0x010	IO_LOCKS	0x090	
0x011	LOCKED_TIME	0x091	
0x012	WORD_LOCKS	0x092	
0x013	AM_LOCKS	0x093	
0x014	RX_AGGREGATE	0x094	
0x015	FRAMING_ERR	0x095	
0x016	BIP_ERROR	0x096	
0x017	HW_ERR	0x097	
0x018-0x01f	Reserved		
0x020	PCS-VLANE0_5	0x0a8	
0x021	PCS-VLANE6_11	0x0a9	
0x022	PCS-VLANE12_17	0x0aa	
0x023	PCS-VLANE18_19	0x0ab	
0x024-0x02f	Reserved		
0x030	PRBS_CTRL	0x0ac	
0x031	PRBS_ERR_INJ	0x0ad	
0x032	PRBS_EFLAGS	0x0ae	
0x033-0x03f	Reserved		

Table D–1. Address Map and Register Name Changes for 11.0 SP1

Current 40/100GbE Address	Current Name	Previous Address	Previous Name
0x040-0x0ff	Reserved		
0x100	MAC_VERSION	0x0c1	
0x101	SCRATCH_MAC	0x0c0	
0x102	MAC_CMD_config	0x0a0	GEN_CTRL
0x103	RX_FILTER_CTRL	0x09f	RXF_CTRL
0x104-0x10f	Reserved		
0x110	RECEIVE_PAUSE_STATUS	0x0b4	
0x111	RECEIVE_SOURCE_ADDR_LSB	0x0b5-0x0b6	
0x112	RECEIVE_SOURCE_ADDR_MSB		
0x113	RECEIVE_PAUSE_CONTROL	0x0b7	
0x114	INSERT_PAUSE_CONTROL	0x0b8	
0x115	TX_PAUSE_DST_ADDR_LSB	0x0b9-0x0ba	RECEIVE_DST_ADDR
0x116	TX_PAUSE_DST_ADDR_MSB		
0x117	INSERT_PAUSE	0x0bb	
0x118-0x13f	Reserved		
0x140	MADDR_CTRL	0x0c2	
0x141	SRC_AD_LO	0x0d0	
0x142	SRC_AD_HI	0x0d1	
0x143–0x15f	Reserved		
0x160-0x017f	DST_ADR	0xe0-0x0ff	
0x180-x01ff	Reserved		
0x200-0x02	TX statistics registers	0x140-0x169, 0x0a1-0x0a4	
0x200-0x229	TX statistics counters	0x140-0x169, 0x0a1-0x0a4	
0x022a	CNTR_TX_ST_LO	0x0a1	
0x022b	CNTR_TX_ST_HI	0x0a2	
0x022c	CNTR_TX_DB_LO	0x0a3	
0x022d	CNTR_TX_DB_HI	0x0a4	
0x022e-0x027f	Reserved		
0x280-0x2ff	RX statistics registers	0x180-0x1a9, 0x98-0x9e, 0xaf-0xb3	
0x280-0x2a9	RX statistics counters	0x180-0x1a9	
0x2aa	CNTR_RX_ST_LO	0x09b	
0x2ab	CNTR_RX_ST_HI	0x09c	
0x2ac	CNTR_RX_DB_LO	0x09d	
0x2ad	CNTR_RX_DB_HI	0x09e	
0x2ae	CNTR_RXF_ST_LO	0x0b0	
0x2af	CNTR_RXF_ST_HI	0x0b1	
0x2b0	CNTR_RXF_DB_LO	0x0b2	
0x2b1	CNTR_RXF_DB_HI	0x0b3	
0x2b2	CNTR_RXF_LONG	0x0af	

Table D–1. Address Map and Register Name Changes for 11.0 SP1

Current 40/100GbE Address	Current Name	Previous Address	Previous Name
0x2b3	Reserved		
0x2b4	CNTR_RX_EBLK	0x098	
0x2b5	Reserved		
0x2b6	CNTR_RX_RUNT	0x099	
0x2b7	Reserved		
0x2b8	CNTR_RX_FCS	0x09a	
0x2b9			
0x2ba-0x2ff	Reserved		
0x300-0x3ff	Reserved		
100GbE Ethernet Example Design Registers			
0x400	PMD_VERSION	0x041	
0x401	SCRATCH_PMD	0x040	
0x402	PMD_CMD_CONFIG	0x043	CFP_ENA
0x403	PMD_STATUS	0x042	CFP_STATUS
0x404-0x40f	Reserved		
0x410	MDIO_WDATA	0x048	
0x411	MDIO_RDATA	0x049	
0x412	MDIO_ADDR	0x04a	
0x413	MDIO_CMD	0x04b	
0x414-0x41f	Reserved		
0x420	2WS_WDATA	0x050	I2C_WDATA
0x421	2WS_RDATA	0x051	I2C_RDATA
0x422	2WS_ADDR	0x052	I2C_ADDR
0x423	2WS_CMD	0x053	I2C_CMD

Revision History

The following table summarizes the new features and changes in the user guide for the 40- and 100-Gbps Ethernet MAC and PHY MegaCore functions.

Date	Version	Changes

June 2012	1.0	■ Updated for use with version 12.0 of the Quartus II software.
		■ Updated address map.
		■ Updated device family support.
		■ Updated interfaces for 40-100GbE IP cores with adapters and without adapters. Additional interfaces include:
		  ■ MAC and PHY asynchronous resets.
		  ■ MAC to PHY connections.
		  ■ Lane-to-lane deskew.
		  ■ Statistics counters increment vectors.
		  ■ Link fault signaling, including remote fault and local fault.
		■ Updated RTL hierarchy, directory structure, and wrapper reorganization.
		■ Clocking revisions:
		  ■ `clk50` has been removed and replaced by `clk_status`.
		  ■ `clk_din/clk_txmac` and `clk_dout/clk_rxmac` have been added as TX and RX input clocks.
		■ Feature additions:
		  ■ Controllable FCS (CRC) insertion and removal.
		  ■ Cut-through mode runt removal.
		  ■ PCS BER monitor.
		  ■ PCS test pattern generation and check.
		  ■ Reduced RX destination MAC address checking from 16 addresses to 1 address.
		  ■ Preserved FCS result.
		  ■ Statistics counters implemented as a synthesis option.
		■ Updated or added software registers:
		  ■ Test pattern counter.
		  ■ Link fault signaling.
		  ■ CRC configuration.
		  ■ MAC hardware error.
		  ■ MAC and PHY resets.
		  ■ PCS hardware error.
		  ■ BER monitor.
		  ■ Test mode.

Date	Version	Changes
June 2012 (continued)	1.0	■ Updated or added software registers (continued): ■ MAC address. ■ Statistics counters: roll-overs and increment vectors ■ Additional parameters: STATS_CNTRS_OPTION and FAST_SIMULATION. ■ Updated testbenches and simulation examples. ■ Updated reset signals and reset bits. ■ Additional information regarding oversized frames.
November 2011	Early Access	■ Corrected the following issues in the MegaCore function: ■ Corrected sequence ordered set encoding in PCS. ■ Corrected error control block encoding in PCS. ■ Corrected pause logic to accomodate multiple pause requests. ■ Timing performance improvements: ■ Optimized RTL for better timing performance. ■ Clocking revisions: ■ clk_status and clk_csr now support 100 MHz operation in Stratix V devices for PHY IP calibration. ■ Added Stratix V resource utilization information. ■ Updated definition l<*n*>_rx_ready to include fact that the RX MAC can only be backpressured for a limited number of cycles; consequently, the application should be able to accept a continuous data stream.
September 2011	Early Access	■ Feature additions: ■ Added optional adapters that guarantee the start of packet is always in lane 0. ■ Added full statistics counters module. ■ Modified and improved register map. ■ Provided separate product ID and ordering code for 40GbE and 100GbE MAC and PHY. ■ Corrected the following issues in the MegaCore function: ■ Data is no longer reversed in the adapters. ■ The multicast address is used for the multicast pause frames. ■ The pause state machine loads new pause times as required. ■ Short frames on the TX datapath are converted correctly from 8 words to 5 words. ■ Updated resource utilization numbers. ■ User guide enhancements: ■ Rewrote and added new sections and drawings. ■ Added default values of registers after reset. ■ Combined 40GbE and 100GbE in one user guide. ■ Corrected description of PMD_CMD_CONFIG bit. Writing a 1 enables the PMD.

Date	Version	Changes
September 2011 (continued)	Early Access	■ User guide enhancements (continued): ■ Corrected description of `PMD_CMD_CONFIG` bit. Writing a 1 enables the PMD. ■ Corrected description of `RX_AGGREGATE` register. Definitions for bit[0] and bit[1] were reversed. ■ Corrected description of `pause_quanta`. ■ Corrected descriptions of Figure 3–10 on page 3–11 through Figure 3–12 on page 3–12. ■ Removed `din_in_packet` from Figure 3–2 on page 3–3.
July 2010	Early Access	Initial early access release.

How to Contact Altera

To locate the most up-to-date information about Altera products, refer to the following table.

Contact (1)	Contact Method	Address
Technical support	Email	40-100gbe_support@altera.com
Nontechnical support (general)	Email	nacomp@altera.com
(software licensing)	Email	authorization@altera.com

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *Stratix IV Design Guidelines*.
italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections in a document and titles of Quartus II Help topics. For example, "Typographic Conventions."

Visual Cue	Meaning
Courier type	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. The suffix `n` denotes an active-low signal. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
↵	An angled arrow instructs you to press the Enter key.
1., 2., 3., and a., b., c., and so on	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■ ■	Bullets indicate a list of items when the sequence of the items is not important.
	The hand points to information that requires special attention.
	The question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	The multimedia icon directs you to a related multimedia presentation.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.
	The feedback icon allows you to submit feedback to Altera about the document. Methods for collecting feedback vary as appropriate for each document.