LogiCORE IP Ethernet AVB Endpoint v2.4

DS677 July 23, 2010 **Product Specification**

Introduction

The LogiCORE™ IP Ethernet Audio Video Bridging (AVB) Endpoint core delivers a flexible solution to enhance standard Ethernet MAC functionality, providing prioritized channels through an existing MAC designed to supply reliable, low latency, Quality of Service for *live* audio or video data. The core is designed to emerging *IEEE802.1* standards from the Audio/Video Bridging (AVB) Task Group.

Features

- Designed to the following IEEE emerging specifications:
  - *P802.1AS* (based on IEEE1588)

    Supports clock master functionality, clock slave functionality, and the Best Master Clock Algorithm (BMCA)
  - P802.1Qav

    Supports arbitration between different priority traffic and implements bandwidth policing
- Supports AVB Endpoint talker and/or listener functionality
- Supports Ethernet speeds of 100 Mbps and 1 Gbps
- Includes software drivers for the core (which implement P802.1AS). These drivers can be run on an embedded processor; the core includes an IBM PLB v4.6 bus, providing easy integration with the Xilinx Embedded Development Kit (EDK).
- Can optionally be delivered with the directory structure and supporting files required for immediate import into an EDK project ('pcore' format).

LogiCORE IP Facts				
Core Specifics				
Supported Device Family[1] [2]	Virtex®-6 (LX, LXT, SXT, HXT) Virtex-5 Spartan®-6,Spartan-3/3E Spartan-3A/3AN/3A DSP			
Resources Used[3]				
FPGA Family	I/O	LUTs	FFs	Block RAMs
Virtex-5/Virtex-6 Spartan-6	n/a	1490-2060	1420-1910	2
Spartan-3A/3AN/3A DSP Spartan-3/3E	n/a	1260-1940	1440-1860	3
Special Features	Seamless connection to LogiCORE IP Tri-Mode Ethernet MAC core			
Provided with Core				
Documentation	Product Specification User Guide Release Notes			
Design File Formats	VHDL and/or Verilog			
Constraints File	.ucf (user constraints file)			
Verification	VHDL and/or Verilog demonstration test bench			
Instantiation Template	VHDL and/or Verilog Wrapper			
Additional Items	Software drivers provided to the P802.1AS specification			
Design Tool Requirements				
Xilinx Tools	ISE® v12.2			
Simulation	Mentor Graphics ModelSim v6.5c and above Cadence Incisive Enterprise Simulator (IES) v9.2 and above Synopsys VCS and VCS MX 2009.12 and above			
Synthesis	XST 12.2			

1. For the complete list of supported devices, see the release notes for this core.
2. Slowest speed grades of listed device families are supported.
3. Resource utilization depends on user configuration; see Table 29.

© Copyright 2009-2010 Xilinx, Inc. XILINX, the Xilinx logo, Virtex, Spartan, ISE and other designated brands included herein are trademarks of Xilinx in the United States and other countries. The PowerPC name and logo are registered trademarks of IBM Corp. and used under license. All other trademarks are the property of their respective owners.

Overview of Audio Video Bridging

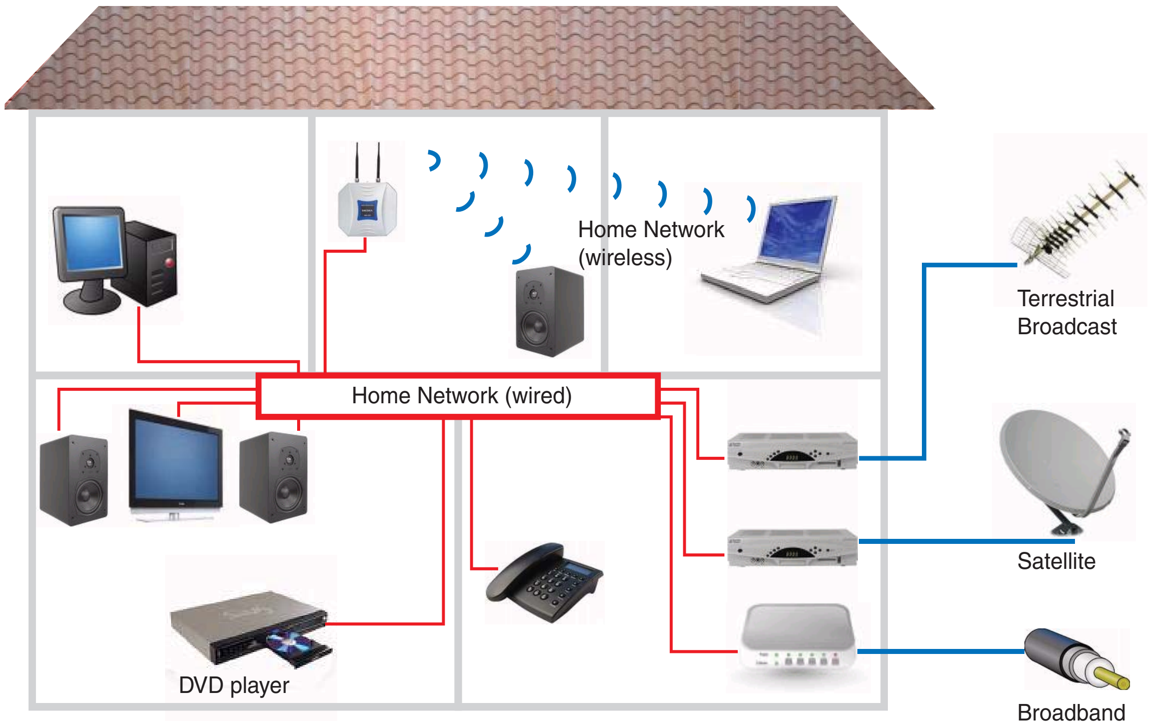

Figure 1: **An Example AVB Home Network**

Figure 1 illustrates a potential home network, consisting of wired (Ethernet) and wireless components that utilize the technology being defined by the *IEEE802.1* Audio Video Bridging (AVB) Task Group. This illustrates potential audio/video *talkers* (for example, a Cable or Satellite Content Provider, or home MP3 player) and a number of potential *listeners* (for example, TV sets in several rooms). In addition, users of the various household PCs may be surfing the internet. It is important to note that all of this data is transferred across the single home network backbone. To understand the requirements of this network, it is important to differentiate between data types:

- **Audio and Video streaming data,** referred to in this document as *AV traffic*. Requires a good quality of service to avoid, for example, TV picture breakup, and must be transferred reliably and with guaranteed low latency.
- **Other data,** referred to in this document as *legacy traffic*. Does not have the strict requirement of AV traffic; data can be started, stopped, and delayed without serious consequence, for example, a PC surfing the internet.

For these reasons, an important aspect of AVB technology is to prioritize the audio/video streaming data (AV traffic) over that of standard data transfer (legacy traffic).

AVB Specifications

The *IEEE802.1* Audio Video Task Group is currently working on new specifications combined to define the following technology.

P802.1AS

This specification defines how to synchronize a common time base across an entire AVB network, utilizing functionality from *IEEE1588* (version 2), known as Precise Timing Protocol (PTP). This common time base is in the form of a Real Time Clock ("RTC"), effectively a large counter that consists of a 32-bit nanoseconds field and a 48-bit seconds field. A single device on the network is designated as the clock master (by automatic resolution) using a Best Master Clock Algorithm (BMCA). All other devices resolve to be slaves. Using the *P802.1AS PTP*, all slave devices regularly update their own RTC to match that of the network clock master.

This common time base has various applications:

- It can be used to synchronize common media clocks (audio clocks or video pixel clocks) across the entire network to match audio and video data rates between talkers and listeners.
- It can be used by an Ethernet AVB Endpoint System that is configured as a "talker" to time a class measurement interval for an SR stream. (The class measurement interval for a stream depends upon the SR class associated with the stream: SR class A corresponds to a class measurement interval of 125 microseconds; SR class B corresponds to a class measurement interval of 250 microseconds). The class measurement interval for a stream is used to limit the number of data frames that are placed into the stream's queue per class measurement interval.
- It can be used by higher layer applications (for example *IEEE1722*) to provide presentation timestamps for audio and video data. This is used, for example, to synchronize the *lip sync* on a TV set so a viewer hears the words at the same time as they see the lips move.

The *P802.1AS* specification is implemented in the Ethernet AVB Endpoint using a combination of hardware and software. The hardware components are incorporated into the core, and the software component is provided with the core in the form of drivers. These drivers should be run on an embedded processor (MicroBlaze™ or PowerPC®).

P802.1Qav

This specification defines the mechanism for queuing and forwarding AV traffic from a talker to a listener across the network, which may include several network hops (network bridge devices that the data must pass through).

P802.1Qav is also responsible for enforcing the 75% maximum bandwidth restriction across each link of the network that can be reserved for AV traffic.

P802.1Qat

This specification defines a Stream Reservation Protocol (SRP) which must be used over the AVB network. Every listener that intends to receive audio/video AV traffic from a talker must make a request to reserve that bandwidth. Both the talker and every bridge device that exists between the talker and the listener has the right to decline this request. Only if each device is capable of routing the new AV traffic stream without violating the 75% total bandwidth restriction (when taking into account previously granted bandwidth commitments), will the bandwidth request be successful. However, after granted, this audio/video stream is reliably routed across the network until the reservation is removed.

Note: No hardware components are required for the *P802.1Qat* specification because it is a pure software task. This software is not provided by the Ethernet AVB Endpoint core.

Typical Implementation

Figure 2 illustrates a typical implementation for the Ethernet AVB Endpoint core. Endpoint refers to a talker or listener device from the example network shown in Figure 1 as opposed to an intermediate bridge function, which is not supported.

In the implementation, the Ethernet AVB Endpoint core is shown connected to a Xilinx Tri-Mode Ethernet MAC core, which in turn is connected to an AVB-capable network. All devices attached to this network should be AVB capable to obtain the full Quality of Service advantages for the AV traffic. This AVB network may be a professional or consumer network, as illustrated in Figure 1.

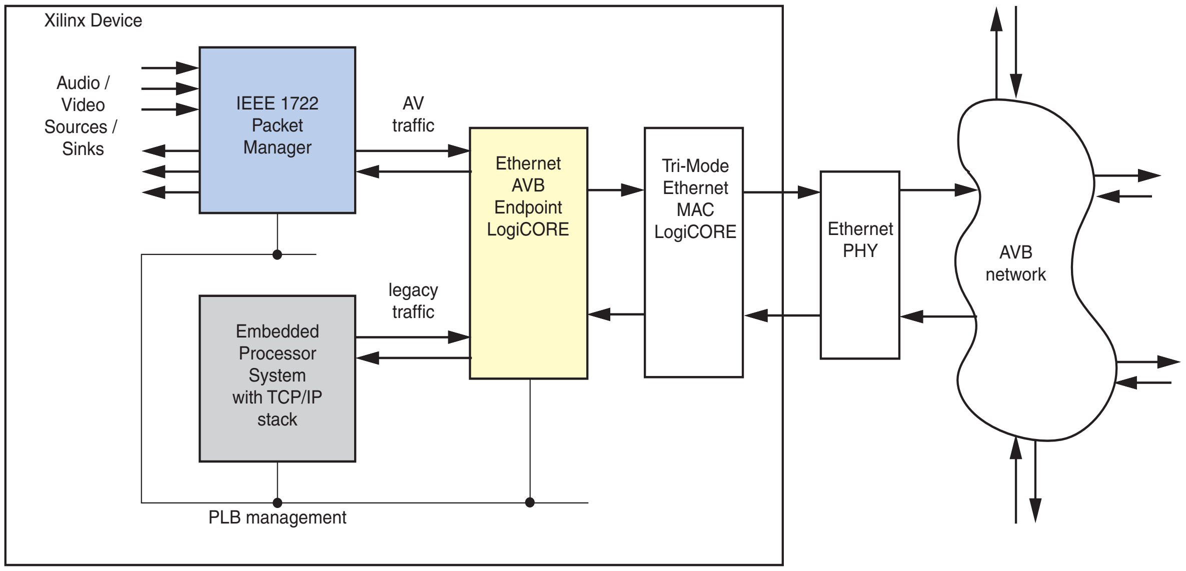
Xilinx Device
Audio / Video Sources / Sinks
IEEE 1722 Packet Manager
AV traffic
Ethernet AVB Endpoint LogiCORE
Tri-Mode Ethernet MAC LogiCORE
Ethernet PHY
AVB network
legacy traffic
Embedded Processor System with TCP/IP stack
PLB management

Figure 2: **Example Ethernet AVB Endpoint System**

Figure 2 illustrates that the Ethernet AVB Endpoint core supports two main data interfaces at the client side:

1. The **AV traffic** interface is intended for the Quality of Service audio/video data. Illustrated are a number of audio/video sources (for example, a DVD player), and a number of audio/video sinks (for example, a TV set). The Ethernet AVB Endpoint gives priority to the **AV traffic** interface over the **legacy traffic** interface, as dictated by *IEEE P802.1Qav* 75% bandwidth restrictions.
2. The **legacy traffic** interface is maintained for *best effort* Ethernet data: Ethernet as we know it today (for example, the PC surfing the internet in the description for Figure 1). Wherever possible, priority is given to the **AV traffic** interface (as dictated by *IEEE P802.1Qav* bandwidth restrictions), but a minimum of 25% of the total Ethernet bandwidth is always available for legacy Ethernet applications.

The **AV traffic** interface in Figure 2 is shown as interfacing to a 1722 Packet Manager block. The *IEEE1722* is also an evolving standard which specifies the embedding of audio/video data streams into Ethernet Packets. The 1722 headers within these packets can optionally include presentation timestamp information. Contact Xilinx for an engineering solution and for more system-level information.

Functional Block Introduction

Figure 3 illustrates the functional blocks of the Ethernet AVB Endpoint core, each of which is introduced in the sections that follow.

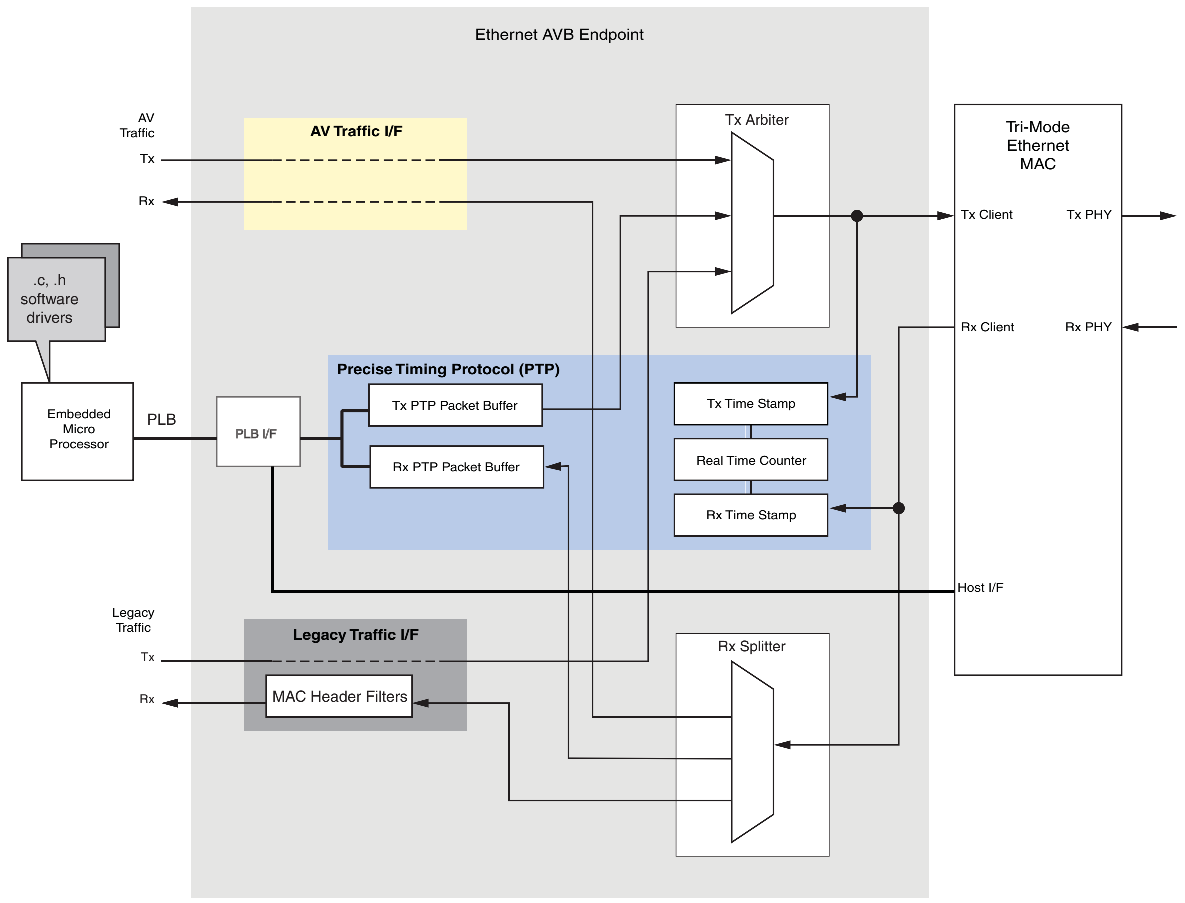

Figure 3: **A Functional Block Diagram of the Ethernet AVB Endpoint Core**

PLB Interface

The core provides a PLB version 4.6 interface as its configuration port to provide easy integration with the Xilinx Embedded Development Kit and access to an embedded processor (MicroBlaze or PowerPC), which is required to run the "Software Drivers." All the configuration and status register address space of the Ethernet AVB Endpoint core can be accessed through the PLB. Additionally, the core provides a logic shim which is connected to the Host I/F of the supported Xilinx Tri-Mode MAC core; this enables all configuration and status registers of the MAC to also be available via the PLB.

AV Traffic Interface

The AV traffic interface provides a dedicated full-duplex port for the high priority AV data, as described in "AV Traffic Interface," page 13.

Legacy Traffic Interface

The legacy traffic interface provides a dedicated full-duplex port for the legacy data, as described in "Legacy Traffic Interface," page 10.

Unless the core is generated in pcore format for import into EDK, then "MAC Header Filters" are provided on the receiver path. These filters have a greater flexibility than the standard address filter provided in the Tri-Mode Ethernet MAC core (which must be disabled).

Please see the Ethernet AVB Endpoint User Guide (UG492) for EDK pcore information.

Tx Arbiter

Data for transmission over an AVB network can be obtained from three source types:

1. **AV Traffic.** For transmission from the AV Traffic I/F of the core.
2. **Precise Timing Protocol (PTP) Packets**. Initiated by the software drivers using the dedicated hardware "Tx PTP Packet Buffers."
3. **Legacy Traffic**. For transmission from the Legacy Traffic I/F of the core.

The transmitter (Tx) arbiter selects from these three sources in the following manner. If there is AV packet available and the programmed AV bandwidth limitation is not exceeded then the AV packet is transmitted, otherwise the Tx arbiter checks to see if there are any PTP packets to be transmitted, otherwise if there is an available legacy packet then this will be transmitted. To comply with the specifications, this should not be configured to exceed 75%. The arbiter then polices this bandwidth restriction for the AV traffic and ensures that on average, it is never exceeded. Consequently, despite the AV traffic having a higher priority than the legacy traffic, there is always remaining bandwidth available to schedule legacy traffic. The output of the arbiter should be connected directly to the client Tx interface of the connected Ethernet MAC, as illustrated.

Receiver Splitter

The input to the splitter is connected directly to the client Receive (Rx) interface of the connected Ethernet MAC. Received data from an AVB network can be of three types:

- **Precise Timing Protocol (PTP) Packets**: Routed to the dedicated hardware "Rx PTP Packet Buffers," which can be accessed by the "Software Drivers." PTP packets are identified by searching for a specific value in the MAC Length/Type field.
- **AV Traffic**. Routed to the AV Traffic I/F of the core. These packets are identified by searching for MAC packets containing a MAC VLAN field with one of two possible configurable VLAN priority values.
- **Legacy Traffic**: Routed to the Legacy Traffic I/F of the core. All packet types that are not identified as PTP or AV Traffic are considered legacy traffic.

MAC Header Filters

When the core is generated in pcore format for import into the EDK the MAC Header Filters are not included. Please see the Ethernet AVB Endpoint User Guide (UG492) for EDK pcore information.

When present, the MAC Header Filters provided on the receiver legacy traffic path have a greater flexibility than the standard address filter provided in the Tri-Mode Ethernet MAC (which must be disabled). The MAC Header Filters include the ability to filter across any of the initial 16-bytes of an Ethernet frame, including the ability to filter only on the Destination Address, Length/Type Field, VLAN tag (if present), or any bit-wise match combination of the above. Eight individual MAC Header Filters are provided, each of which is separately configured. See "MAC Header Filter Configuration" for additional information.

Precise Timing Protocol Blocks

The various hardware Precise Timing Protocol (PTP) blocks within the core provide the dedicated hardware to implement the *IEEE P802.1AS* specification. However, full functionality is only achieved using a combination of these hardware blocks coupled with functions provided by the "Software Drivers" (run on an embedded processor). In addition, the following hardware block descriptions provide some insight into the software driver functionality.

Note: The following descriptions are simplistic and only provide a conceptual idea of how the PTP protocol operates and how it is implemented in the core as a symbiosis of dedicated hardware and software functions. For a comprehensive understanding about PTP, see the *IEEE P802.1AS* specification.

Tx PTP Packet Buffers

The PTP packet buffer contains pre-initialized templates for seven different PTP packets defined by the *P802.1AS* specification. The buffer contents are read/writable through the PLB and a separate configuration register within the core requests of the Tx Arbiter which of these seven packets is to be transmitted. A dedicated interrupt signal is generated by the core whenever a PTP packet has been transmitted.

The software drivers provided with the core, using the PLB and dedicated interrupts, use this interface to periodically update specific fields within the PTP packets and request transmission of these packets.

Tx Time Stamp

Whenever a PTP packet is transmitted, a sample of the current nanosecond value of the local RTC is taken. This is written into a dedicated field within the Tx PTP Packet Buffer, where it is accessible along side the content of the PTP frame that was just transmitted. By the time the Tx PTP buffer raises its dedicated interrupt, this timestamp is available for the microprocessor to read. This sampling of the RTC is performed in hardware for accuracy.

Rx PTP Packet Buffers

Received PTP Packets will be written to the Rx PTP Packet Buffer by the Rx Splitter. This buffer is capable of storing up to 16 separate PTP frames. Whenever a PTP packet is received, a dedicated interrupt will be generated. The contents of the stored packets can be read via the PLB. The oldest stored frame will always be overwritten by a new frame reception and so a configuration register within the core will contain a pointer to the most recently stored packet.

The software drivers provided with the core, using the PLB and dedicated interrupt, use this interface to decode, and then act on the received PTP packet information.

Rx Time Stamp

When a PTP packet is received, a sample of the current nano-second value of the RTC is taken; this is written into a dedicated field within the Rx PTP Packet Buffer, where it is accessible along side the PTP frame that was just received. By the time the Rx PTP buffer raises its dedicated interrupt, this timestamp is available for the microprocessor to read. This sampling of the RTC is performed in hardware for accuracy.

RTC

A significant component of the PTP network wide timing synchronization mechanism is the Real Time Counter (RTC), which provides the common time of the network. Every device on the network maintains its own local version.

The RTC is effectively a large counter consisting of a 32-bit nanoseconds field (the unit of this field is 1 nanosecond and this field counts the duration of exactly one second, then resets back to zero), and a 48-bit second field (the unit of this field is one second and this field increments when the nanosecond field saturates at 1 second). The seconds field only wraps around when its count fully saturates. For this reason, the entire RTC is designed never to wrap around in our lifetime. The RTC counter is implemented as part of the core in hardware.

Conceptually, this counter is not related to the frequency of the clock used to increment it. A configuration register within the core provides a configurable increment rate for this counter; this increment register simply takes the value of the clock period being used to increment the RTC. However, the resolution of this increment register is very fine, in units of 1/1048576 ($1/2^{20}$) fraction of one nanosecond. For this reason, the RTC increment rate can be adjusted to a very fine degree of accuracy, which provides the following features:

- The RTC can be incremented from any available clock frequency greater than the AVB standards defined minimum of 25 MHz. However, the faster the frequency of the clock, the smaller the step increment and the smoother the overall RTC increment rate. Xilinx recommends clocking the RTC logic at 125 MHz because this is a readily available clock source (obtained from the transmit clock source of the Ethernet MAC at 1 Gbps speed). This frequency significantly exceeds the minimum performance of the *P802.1AS* specification.
- When acting as a clock slave, the rate adjustment of the RTC can be matched to that of the network clock master to an exceptional level of accuracy. The software drivers provided with this core periodically calculate the increment rate error between itself and the master and update the RTC increment value accordingly.

The core also contains a configuration register that allows a large step change to be made to the RTC. This can be used to initialize the RTC after power-up. It is also used to make periodic corrections, as required, by the software drivers when operating as a clock slave; if the increment rates are closely matched, these periodic step corrections are small.

Software Drivers

Software Drivers are delivered with the Ethernet AVB Endpoint core. These drivers provide functions that utilize the dedicated hardware within the core for the Precise Timing Protocol (PTP) *IEEE P802.1AS* specification. Functions include:

- The Best Clock Master Algorithm (BMCA) to determine whether the core should operate in master clock or slave clock mode
- PTP Clock Master functions
- PTP Clock Slave functions, which accurately synchronize the local Real Time Clock (RTC) to match that of the network clock master

If the core is acting as clock master, the software drivers delivered with the core periodically sample the current value of the RTC and transmit this value to every device on the network using the *P802.1* defined PTP packets. The hardware "Tx Time Stamp" logic, using the mechanism defined in P802.1AS, ensures the accuracy of this RTC sample mechanism.

If the core is acting as a clock slave, the local RTC is closely matched to the value and frequency of the network clock master. This is achieved, in part, by receiving the PTP frames transmitted across the network by the clock master (and containing the masters sampled RTC value). The PTP mechanism also tracks the total routing delay across the network between the clock master and itself. The software drivers use this data, in conjunction with recent historical data, to calculate the error between its local RTC counter and that of the RTC clock master. The software periodically calculates an RTC correction value and an updated increment rate, and these values are written to appropriate RTC configuration registers.

Tri-Mode Ethernet MAC Core

Although not part of the Ethernet AVB Endpoint core, a Xilinx Tri-Mode Ethernet MAC core is a requirement of the system, as shown in Figure 3. The IEEE Audio Video Bridging technology stipulates the following configuration requirements on this MAC:

- The MAC must only operate in full-duplex mode
- The MAC must only operate at 100 Mbps and/or 1 Gpbs
- VLAN mode must be enabled (the AV traffic always contains VLAN fields)
- Flow Control is not supported on the network and must be disabled
- Jumbo Frames are not supported and must be disabled
- The built-in Address Filter Module of the MAC must be disabled

Interface Signals

All ports of the core are internal connections in the FPGA fabric. All clock signals are inputs and no clock resources are used by the core. This enables clock circuitry to be fully implemented externally to the core netlist, providing full flexibility for clock sharing with other custom logic.

Clocks and Reset

Table 1 defines the clock and reset signals required by the Ethernet AVB Endpoint core.

Table 1: **Clocks and Resets**

Signal	Direction	Description
reset	Input	Asynchronous reset for the entire core
rtc_clk	Input	Reference clock used to increment the "RTC." The minimum frequency is 25 MHz. Xilinx recommends a 125 MHz clock source.
tx_clk	Input	The MAC transmitter clock, provided by the Tri-Mode Ethernet MAC core.
tx_clk_en	Input	A clock enable signal; this must be used as a qualifier for `tx_clk`.
rx_clk	Input	The MAC receiver clock, provided by the Tri-Mode Ethernet MAC core.
rx_clk_en	Input	A clock enable signal; this must be used as a qualifier for `rx_clk`.
host_clk	Input	An input clock for the management interface of the connected Tri-Mode Ethernet MAC core. This clock can be independent, or can be shared with `PLB_clk`.
PLB_clk	Input	The input clock reference for the PLB bus.
tx_reset	Output	Output reset signal for logic on the Legacy Traffic and AV Traffic transmitter paths. This reset signal is synchronous to `tx_clk`; the reset is asserted when a transmitter path reset request is made to the "Software Reset Register."
rx_reset	Output	Output reset signal for logic on the Legacy Traffic and AV Traffic receiver paths. This reset signal is synchronous to `rx_clk`; the reset is asserted when a receiver path reset request is made to the "Software Reset Register."

Legacy Traffic Interface

Transmitter Path Signals

Table 2 defines the core client-side legacy traffic transmitter signals. These signals are used to transmit data from the legacy client logic into the core. All signals are synchronous to the MAC transmitter clock, `tx_clk`, which must be qualified by the corresponding clock enable, `tx_clk_en` (see "Clocks and Resets").

Table 2: **Legacy Traffic Signals: Transmitter Path**

Signal	Direction	Description
legacy_tx_data[7:0]	Input	Frame data to be transmitted is supplied on this port.
legacy_tx_data_valid	Input	A data valid control signal for data on the `legacy_tx_data[7:0]` port.
legacy_tx_underrun	Input	Asserted by the client to force the MAC to corrupt the current frame.
legacy_tx_ack	Output	Handshaking signal asserted when the current data on `legacy_tx_data[7:0]` has been accepted.

Transmitter Path Operation

Figure 4 illustrates the timing of a normal frame transfer. When the legacy client initiates a frame transmission, it places the first column of data onto the `legacy_tx_data[7:0]` port and asserts a logic 1 onto `legacy_tx_data_valid`.

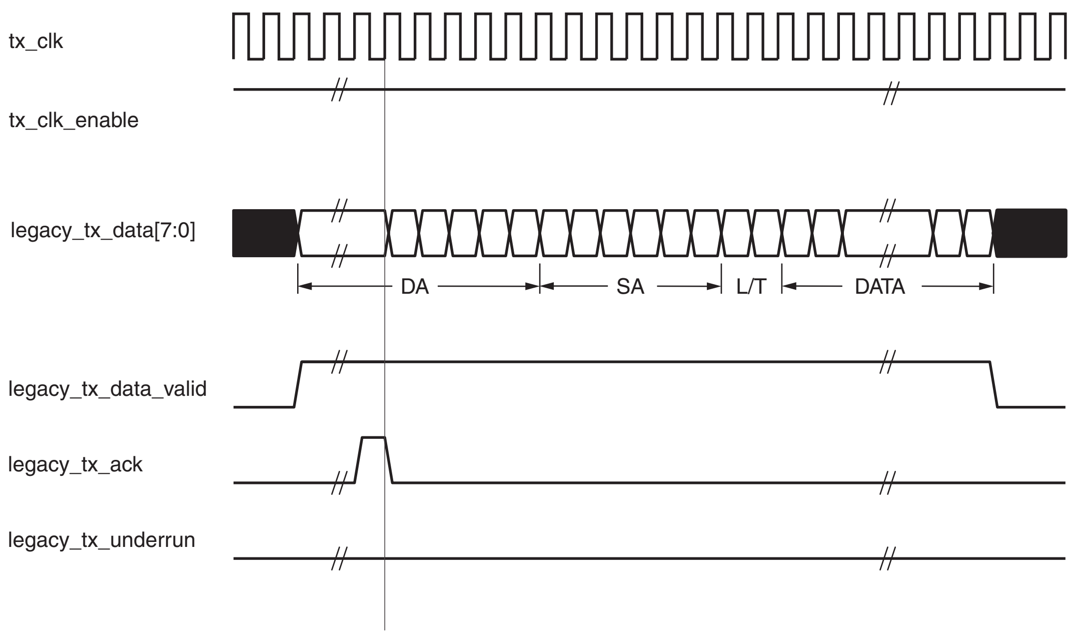

Figure 4: **Normal Frame Transmission across the Legacy Traffic Interface**

After the Ethernet AVB Endpoint core reads the first byte of data, it asserts the `legacy_tx_ack` signal. On the next and subsequent rising clock edges, the client must provide the remainder of the data for the frame. The end of frame is signalled to the core by taking the `legacy_tx_data_valid` to logic 0.

This interface is intentionally *identical* to the client transmitter interface of the supported Xilinx Tri-Mode Ethernet MAC core (there is a one-to-one correspondence between signal names after the `legacy_` prefix has been removed). This provides backwards compatibility–all existing MAC client-side designs can connect to the legacy Ethernet port unmodified.

Legacy Traffic Receiver Path Signals

Table 3 defines the core client side legacy traffic receiver signals. These signals are used by the core to transfer data to the client. All signals are synchronous to the MAC receiver clock, `rx_clk`, which must be qualified by the corresponding clock enable, `rx_clk_en` (see "Clocks and Resets").

Table 3: **Legacy Traffic Signals: Receiver Path**

Signal	Direction	Description
legacy_rx_data[7:0]	Output	Legacy frame data received is supplied on this port.
legacy_rx_data_valid	Output	Control signal for the `legacy_rx_data[7:0]` port.
legacy_rx_frame_good	Output	Asserted at the end of frame reception to indicate that the frame should be processed by the MAC client.
legacy_rx_frame_bad	Output	Asserted at the end of frame reception to indicate that the frame should be discarded by the MAC client–either the frame contained an error, or it was intended for the PTP or AV traffic channel.
legacy_rx_filter_match[7:0]	Output	This output is not present when the core is generated in pcore format for import into EDK since the "MAC Header Filters." are not included. When present, each bit in the bus corresponds to one of the unique"MAC Header Filters." A bit is asserted, in alignment with the `legacy_rx_data_valid` signal, if the corresponding filter number obtained a match.

Legacy Traffic Receiver Path Operation

Figure 5 illustrates the timing of a normal inbound frame transfer that has been accepted by the "MAC Header Filters." The legacy client must be prepared to accept data at any time; there is no buffering within the core to allow for latency in the receive client. After frame reception begins, data is transferred on consecutive clocks to the receive client until the frame is complete. The core asserts the `legacy_rx_frame_good` to indicate that the frame was intended for the legacy traffic client and was successfully received without error.

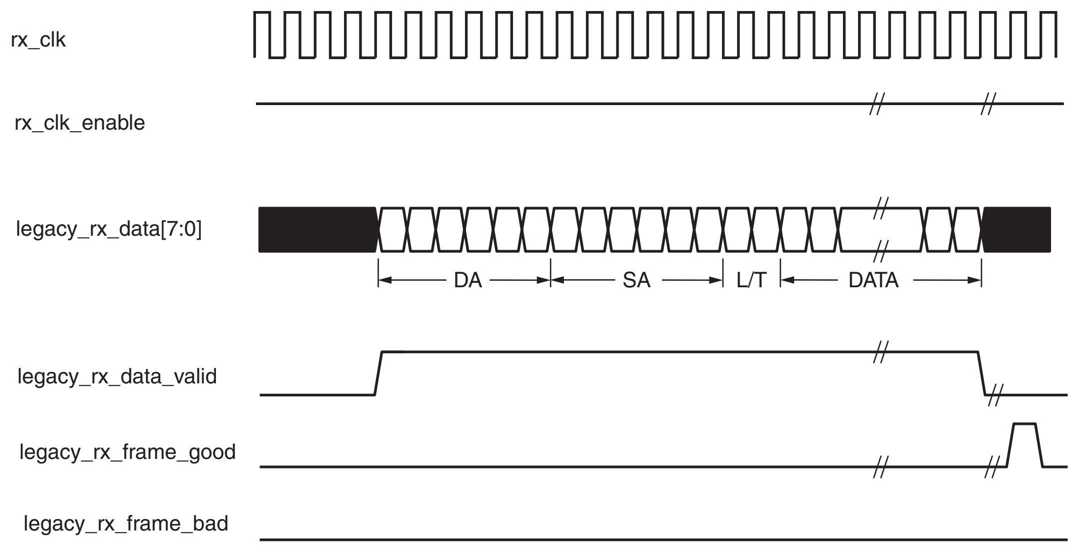

Figure 5: **Normal Frame Reception across the Legacy Traffic Interface**

Reception of any frame in which the `legacy_rx_frame_bad` is asserted (in place of `legacy_rx_frame_good`) indicates that this frame must be discarded; it was either received with errors or was not intended for the legacy traffic interface.

If the "MAC Header Filters" are enabled and do not obtain a match, the frame data does not appear on this interface (`legacy_rx_data_valid` and `legacy_rx_frame_good/legacy_rx_good_bad` are not asserted).

This interface is intentionally *identical* to the client receiver interface of the supported Xilinx Tri-Mode Ethernet MAC core (there is a one-to-one correspondence between signal names after the `legacy_` prefix is removed). This provides backward compatibility–all existing client-side designs should be able to connect to the legacy Ethernet port unmodified.

AV Traffic Interface

Transmitter Path Signals

Table 4 defines the core client-side AV traffic transmitter signals, used to transmit data from the AV client logic into the core. All signals are synchronous to the MAC transmitter clock, `tx_clk`, which must be qualified by the corresponding clock enable, `tx_clk_en` (see "Clocks and Resets").

Table 4: **AV Traffic Signals: Transmitter Path**

Signal	Direction	Description
av_tx_data[7:0]	Input	Frame data to be transmitted is supplied on this port.
av_tx_valid	Input	A data valid control signal for data on the `av_tx_data[7:0]` port.
av_tx_done	Input	Asserted by the AV client to indicate that further frames, following the current frame, are/are not held in a queue.
av_tx_ack	Output	Handshaking signal asserted when the current data on `av_tx_data[7:0]` has been accepted.

Transmitter Path Operation

Figure 6 illustrates the timing of a normal frame transfer. When the AV client initiates a frame transmission, it places the first column of data onto the `av_tx_data[7:0]` port and asserts a logic 1 onto `av_tx_valid`.

After the Ethernet AVB Endpoint core reads the first byte of data, it asserts the `av_tx_ack` signal. On the next and subsequent rising clock edges, the client must provide the remainder of the data for the frame. The end of frame is signalled to the core by taking the `av_tx_valid` to logic 0.

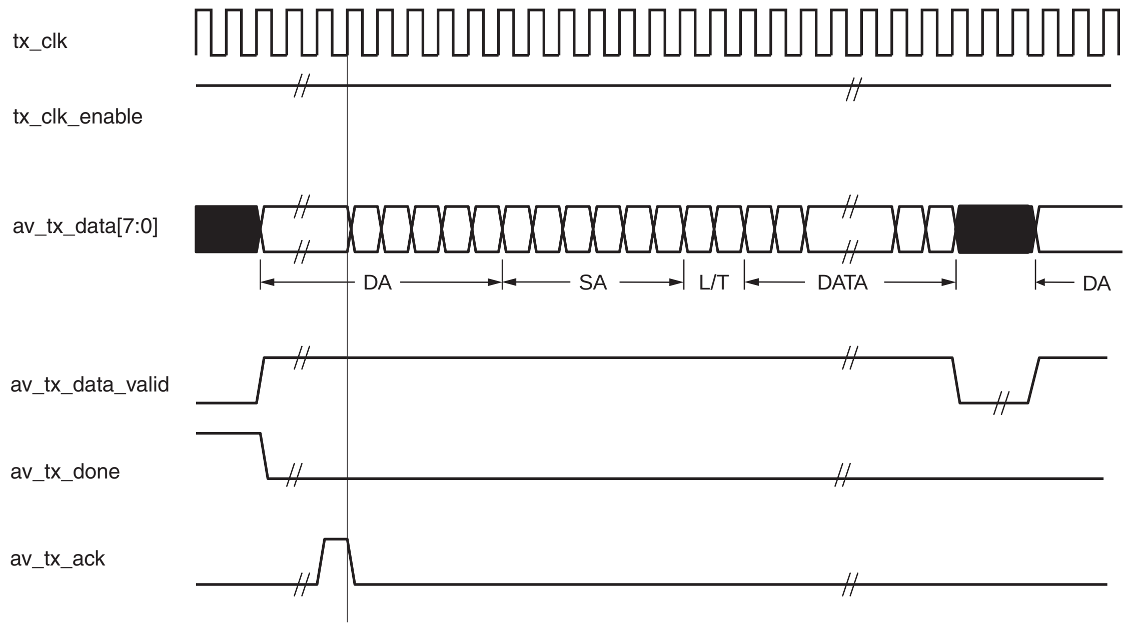

Figure 6: **Normal Frame Transmission across the AV Traffic Interface**

In Figure 6, following the end of frame transmission, the `av_tx_done` signal is held low, which indicates to the "Tx Arbiter" that another AV frame is queued. Unless the configurable bandwidth restrictions have been exceeded, this *parks* the "Tx Arbiter" onto the AV traffic queue. Figure 6 then illustrates the client asserting the `av_tx_valid` signal to request a subsequent frame, and the frame transmission cycle of Figure 6 repeats. However, if no further AV traffic frames are queued, the `av_tx_done` signal should be set to logic 1 immediately following the end of frame transmission. This then allows the "Tx Arbiter" to schedule legacy traffic transmission (if any legacy frames are queued).

If, following the end of frame transmission, the bandwidth allocation for AV traffic has exceeded, the "Tx Arbiter" switches to service the legacy traffic regardless of the state of the `av_tx_done` signal.

For this reason, the `av_tx_done` signal should be considered an aid to the "Tx Arbiter" to help make best use of the available network bandwidth. Asserting this signal after all AV traffic has been serviced immediately allows the "Tx Arbiter" to service the legacy traffic for the remainder of the current isochronous cycle. This helps achieve in excess of the 25% minimum allocation for the legacy traffic. However, holding off the assertion of `av_tx_done` will not act as cheat mode to exceed the maximum bandwidth allocation for the AV traffic.

This interface is intentionally very similar to the "Legacy Traffic Interface." Note, however, that the legacy traffic does not contain a signal that is equivalent to `av_tx_done`.

AV Traffic Receiver Path Signals

Table 5 defines the core client side AV traffic receiver signals, used by the core to transfer data to the AV client. All signals are synchronous to the MAC receiver clock, `rx_clk`, which must be qualified by the corresponding clock enable, `rx_clk_en` (see "Clocks and Resets").

Table 5: **AV Traffic Signals: Receiver Path**

Signal	Direction	Description
av_rx_data[7:0]	Output	AV frame data received is supplied on this port.
av_rx_valid	Output	Control signal for the `av_rx_data[7:0]` port
av_rx_frame_good	Output	Asserted at the end of frame reception to indicate that the frame should be processed by the MAC client.
av_rx_frame_bad	Output	Asserted at the end of frame reception to indicate that the frame should be discarded by the MAC client; either the frame contained an error, or it was intended for the PTP or legacy traffic channel.

AV Traffic Receiver Path Operation

Figure 7 illustrates the timing of a normal inbound frame transfer. The AV client must be prepared to accept data at any time; there is no buffering within the core to allow for latency in the receive client. After frame reception begins, data is transferred on consecutive clocks to the receive client until the frame is complete. The core asserts the `av_rx_frame_good` to indicate that the frame was intended for the AV traffic client, and was successfully received without error.

It should be noted that all frames (including PTP and legacy traffic) appear on this interface. However, all frames that were not AV traffic frames are indicated by the assertion of the `av_rx_frame_bad` signal, and should be discarded.

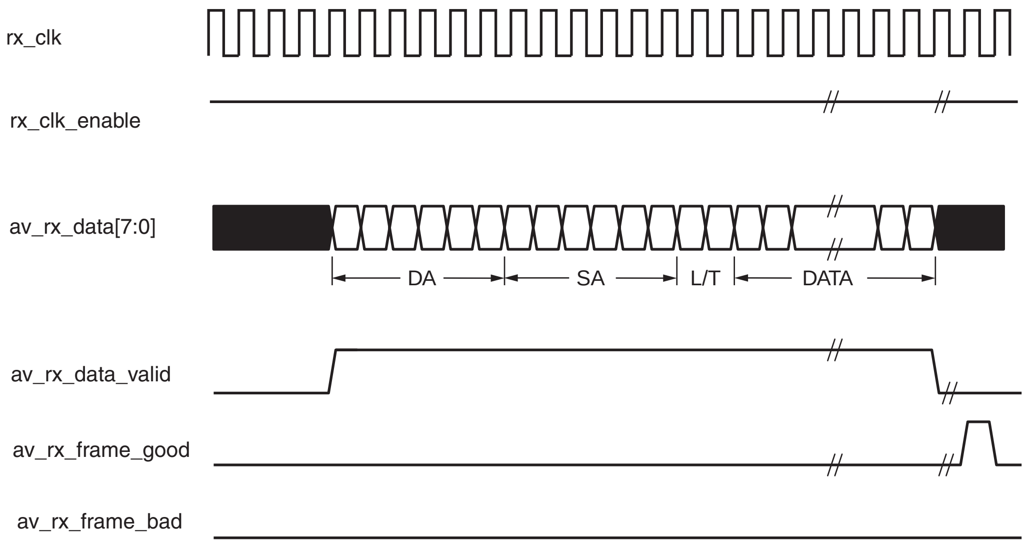

Figure 7: **Normal Frame Reception across the AV Traffic Interface**

Tri-Mode Ethernet MAC Client Interface

Tables 6, 7, and 8 show the ports of the core that connect directly to the port signals of the Tri-Mode Ethernet MAC core solution. The TEMAC solution is made up of three separate IP cores for 10/100/1000 Mbps, 1000 Mbps, and 10/100 Mbps. The interface for all three of these IP cores are the same and will be referred as the Tri-Mode Ethernet MAC signals in the rest of this document.For detailed information about the Tri-Mode Ethernet MAC ports, see the Tri-Mode Ethernet MAC User Guide (UG138).

MAC Transmitter Interface

These signals connect directly to the identically named Tri-Mode Ethernet MAC signals and are synchronous to `tx_clk`.

Table 6: **Tri-Mode Ethernet MAC Transmitter Interface**

Signal	Direction	Description
tx_data[7:0]	Output	Frame data to be transmitted is supplied on this port
tx_data_valid	Output	A data valid control signal for data on the `tx_data[7:0]` port
tx_underrun	Output	Asserted to force the MAC to corrupt the current frame
tx_ack	Input	Handshaking signal asserted when the current data on `tx_data[7:0]` has been accepted by the MAC.

MAC Receiver Interface

These signals connect directly to the identically named Tri-Mode Ethernet MAC signals and are synchronous to `rx_clk`.

Table 7: **Tri-Mode Ethernet MAC Receiver Interface**

Signal	Direction	Description
rx_data[7:0]	Input	Frame data received is supplied on this port.
rx_data_valid	Input	Control signal for the `rx_data[7:0]` port
rx_frame_good	Input	Asserted at the end of frame reception to indicate that the frame should be processed by the Ethernet AVB Endpoint core.
rx_frame_bad	Input	Asserted at the end of frame reception to indicate that the frame should be discarded by the MAC client.

MAC Management Interface

This interface is not present when the core is generated in pcore format for import into EDK. Please see the Ethernet AVB Endpoint User Guide (UG492) for EDK pcore information.

These signals connect directly to the identically named Tri-Mode Ethernet MAC signals and are synchronous to `host_clk`. All MAC configuration and MDIO register space is address mapped into the PLB of the Ethernet AVB Endpoint core. A logic shim automatically drives this interface to access the MAC when the appropriate PLB address space is accessed.

Table 8: **Tri-Mode Ethernet MAC Host Interface (Configuration/Status)**

Signal	Direction	Description
host_opcode[1:0]	Output	Defines the MAC operation (configuration or MDIO, read or write)
host_addr[9:0]	Output	Address of the MAC register to access
host_wr_data[31:0]	Output	Data to be written to the MAC register
host_rd_data_mac[31:0]	Input	Data read from the MAC register (connect to the `host_rd_data[31:0]` signal of the MAC)
host_rd_data_stats[31:0]	Input	Data read from the Ethernet Statistics core (connect to the `host_rd_data[31:0]` signal of the Ethernet Statistics core, if present). If the statistics core is not used, then connect to logic 0.
host_miim_sel	Output	When asserted, the MAC will access the MDIO port, when not asserted, the MAC will access configuration registers
host_req	Output	Used to initiate a transaction onto the MDIO
host_miim_rdy	Input	When high, the MAC has completed its MDIO transaction
host_stats_lsw_rdy	Input	Signal provided by the Ethernet Statistics core to indicate that the lower 32-bits of the statistic counter value is present on the `host_rd_data_stats[31:0]` port. If the statistics core is not used, then connect to logic 0.
host_stats_msw_rdy	Input	Signal provided by the Ethernet Statistics core to indicate that the upper 32-bits of the statistic counter value is present on the `host_rd_data_stats[31:0]` port. If the statistics core is not used, then connect to logic 0.

Processor Local Bus Interface

The Processor Local Bus (PLB) bus on the Ethernet AVB Endpoint core is designed to be integrated directly in the Xilinx Embedded Development Kit (EDK) where it can be easily integrated and connected to the supported embedded processors (MicroBlaze or PowerPC). As a result, the PLB interface does not require in-depth understanding, and the following information is provided for reference only. See the EDK documentation for further information.

The PLB interface, defined by IBM, can be complex and support many usage modes (such as multiple bus masters). It can support single or burst read/writes, and can support different bus widths and different peripheral data widths.

The general philosophy of the Ethernet AVB Endpoint core has been to implement a PLB interface, which is as simple as possible. The following features are provided:

- 32-bit data width
- Implements a simple PLB slave
- Supports single read/writes only (no burst or page modes)

PLB Interface

The following signals are present on the PLB bus; see the IBM PLB specification for detailed information. Shaded rows are unused by this core; inputs are ignored and outputs are tied to a constant. These signals are synchronous to `PLB_clk` (see "Clocks and Resets").

Table 9: **PLB Signals**

PIN name	Direction	Description
PLB_reset	Input	Reset for the PLB, synchronous to PLB_clk
PLB_ABus[0:31]	input	PLB address bus
PLB_UABus[0:31]	Input	PLB upper address bus
PLB_PAvaild	Input	PLB primary address valid indicator
PLB_SAValid	Input	Unused. PLB secondary address valid indicator.
PLB_rdPrim	Input	Unused. PLB secondary to primary read request indicator.
PLB_wrPrim	Input	Unused. PLB secondary to primary write request indicator.
PLB_masterID[0:log2(NUM_MASTERS)]	Input	PLB current master identifier
PLB_abort	Input	PLB abort request indicator
PLB_busLock	Input	Unused. PLB bus lock.
PLB_RNW	Input	PLB read not write
PLB_BE[0:3]	Input	PLB byte enables
PLB_MSize[0:1]	Input	PLB master data bus size
PLB_size[0:3]	Input	PLB transfer size. Only support size 0.
PLB_type[0:2]	Input	PLB transfer type. Only support type 0.
PLB_TAttribute[0:15]	Input	Unused. PLB transfer attribute bus.
PLB_lockErr	Input	Unused. PLB lock error indicator.
PLB_wrDBus[0:31]	Input	PLB write data bus
PLB_wrBurst	Input	PLB write burst transfer indicator
PLB_rdBurst	Input	PLB read burst transfer indicator

Table 9: **PLB Signals *(Cont'd)***

PIN name	Direction	Description
PLB_rdPendReq	Input	Unused. PLB pending read request priority.
PLB_wrPendReq	Input	Unused. PLB pending write request priority.
PLB_rdPendPri[0:1]	Input	Unused. PLB pending read bus request indicator.
PLB_wrPendPri[0:1]	Input	Unused. PLB pending read bus request indicator.
PLB_reqPri[0:1]	Input	Unused. PLB request priority.
Sl_addrAck	Output	Slave address acknowledge
Sl_SSize[0:1]	Output	Slave data bus size
Sl_wait	Output	Slave wait indicator
Sl_rearbitrate	Output	Not used, tied to logic 0. Slave rearbitrate bus indicator.
Sl_wrDack	Output	Slave write data acknowledge
Sl_wrComp	Output	Slave write transfer complete indicator
Sl_WrBTerm	Output	Slave terminate write burst transfer
Sl_rdBus[0:31]	Output	Slave read data bus
Sl_rdWdAddr[0:3]	Output	Slave read word address
Sl_rdDAck	Output	Slave read data acknowledge
Sl_rdComp	Output	Slave read transfer complete indicator
Sl_rdBTerm	Output	Slave terminate read burst transfer
Sl_MBusy[0:NUM_MASTERS-1]	Output	Slave busy indicator
Sl_MWrErr[0:NUM_MASTERS-1]	Output	Unused, tied to logic 0. Slave write error indicator.
Sl_MRdErr[0:NUM_MASTERS-1]	Output	Unused, tied to logic 0. Slave read error indicator.
Sl_MIRQ[0:NUM_MASTERS-1]	Output	Unused, tied to 0s. Slave interrupt indicator.

PLB Operation

Single Read Transaction

Figure 8 illustrates a single read data transfer on the PLB. Note the following:

- Wait states may be added to the Address cycle by asserting `Sl_wait` and delaying `Sl_addrAck`
- Wait states may be inserted in the Read fetch by delaying the assertion of `Sl_rdDAck.`

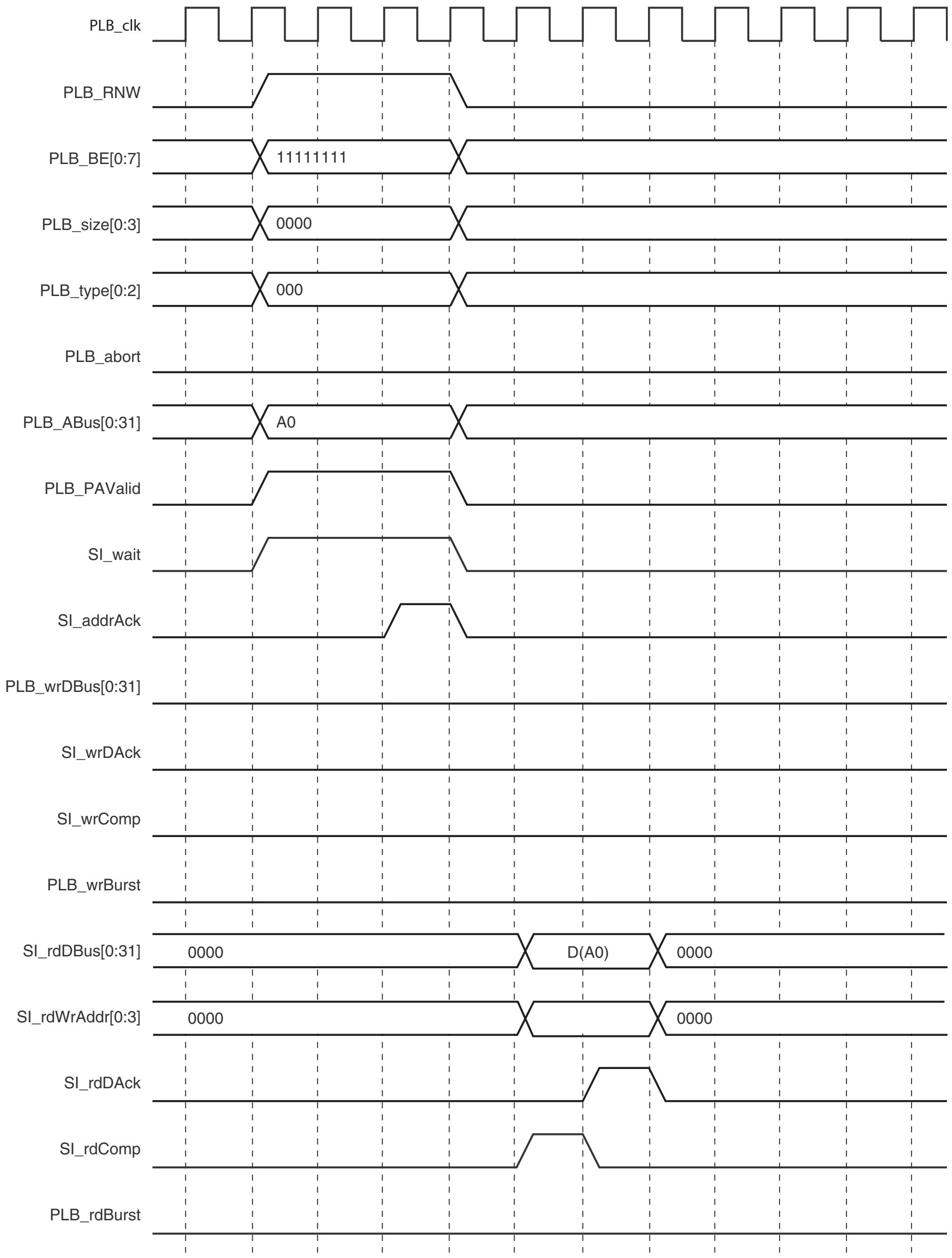

Figure 8: **Single Read Transaction**

Single Write Transaction

Figure 9 illustrates a single write data transfer on the PLB. Note the following:

- Wait states may be added to the Address cycle by asserting `Sl_wait` and delaying `Sl_addrAck`.
- Wait states may be inserted in the Write sample by delaying the assertion of `Sl_wrDAck`.

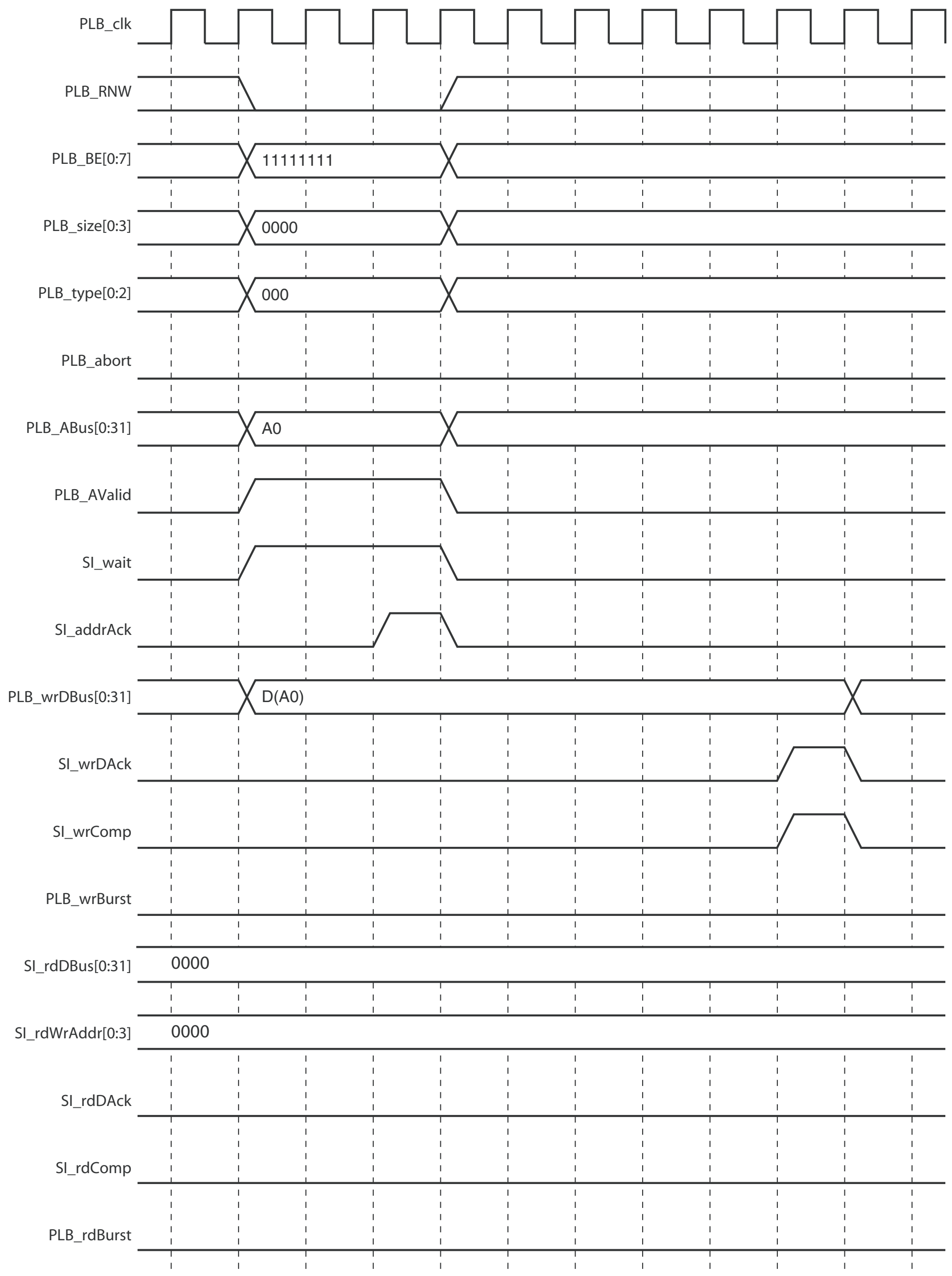

Figure 9: **Single Write Transaction**

Interrupt Signals

Table 10 defines the interrupt signals asserted by the core. All interrupts are active high and are automatically asserted. All interrupts, required by the "Software Drivers" delivered with the core, are cleared by software access to an associated configuration register. It is recommended that these interrupts are routed to the input of an EDK Interrupt Controller module as part of the embedded processor subsystem.

Table 10: **Interrupt Signals**

Signal	Direction	Description
interrupt_ptp_timer	Output	This interrupt is asserted every 1/128 seconds as measured by the "RTC." This acts as a timer for the PTP software algorithms.
interrupt_ptp_tx	Output	This is asserted following the transmission of any PTP packet from the "Tx PTP Packet Buffers."
interrupt_ptp_rx	Output	This is asserted following the reception of any PTP packet into the "Rx PTP Packet Buffers."

PTP Signals

Table 11 defines the signals output from the core by the "Precise Timing Protocol Blocks." These signals are provided for reference only and may be used by an application. For example, the 1722 Packet Managers, as illustrated in Figure 2, require the following:

- `clk8k`. The period of this clock can be used to help control traffic shaping.
- `rtc_nanosec_field` and `rtc_sec_field`. Used in the 1722 presentation timestamp logic.

Table 11: **PTP Signals**

Signal	Direction	Description
rtc_nanosec_field[31:0]	Output	This is the synchronized nanoseconds field from the "RTC."
rtc_sec_field[47:0]	Output	This is the synchronized seconds field from the "RTC".
clk8k	Output	This is an 8 KHz clock which is derived from, and synchronized in frequency, to the "RTC". The period of this clock, 125 µs, can be useful in timing SR class measurement intervals.
rtc_nanosec_field_ 1722[31:0]	Output	The IEEE1722 specification contains a different format for the "RTC.", provided here as an extra port. This is derived and is in sync with the IEEE802.1 AS "RTC."

PLB Address Map and Register Definitions

Figure 10 displays an overview of the Address Space occupied by the Ethernet AVB Endpoint core on the PLB. Common across all addressable space, each unique PLB address value references a single *byte* of data.

The variable PLB_base_address illustrated in Figure 10 and in the tables that follow represent the starting address of the AVB core within the entire PLB address space. The starting address is selected from the CORE Generator™ tool GUI.

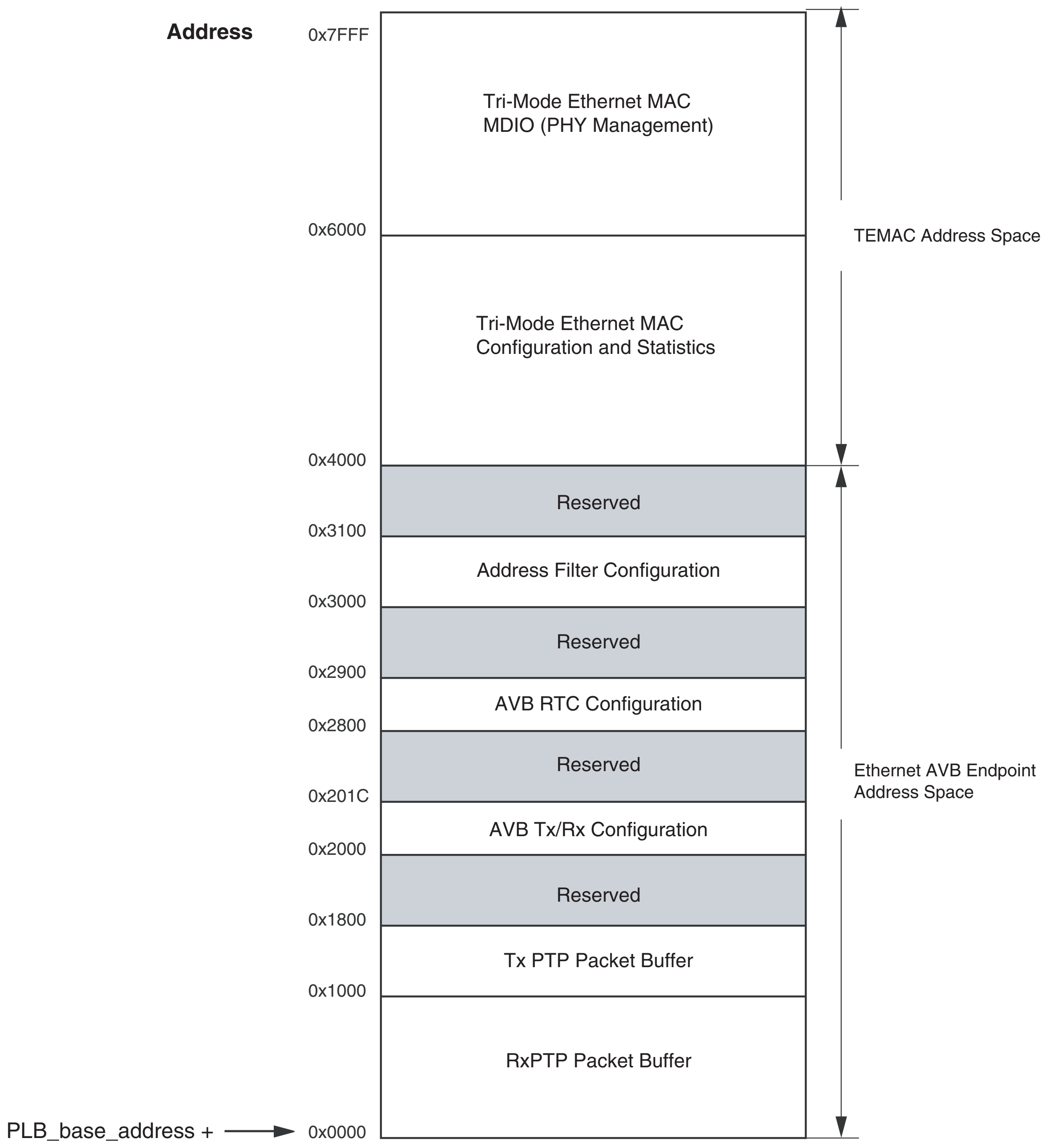

Figure 10: **PLB Address Space of Ethernet AVB Endpoint and Connected Tri-Mode Ethernet MAC**

Ethernet AVB Endpoint Address Space

Rx PTP Packet Buffer Address Space

The Address space of the "Rx PTP Packet Buffers" is 4k bytes, from PLB_base_address to (PLB_base_address + 0x0FFF). This represents the size of a single Virtex-5 FPGA block RAM pair (4k bytes). Every byte of this block RAM can be read from the PLB.

This address space is divided equally into 16 separate buffers of 256 bytes, each of which is capable of storing a unique PTP frame. When received, a PTP frame is written into one of these buffers; then the buffer write pointer increments and points to the next buffer in preparation for subsequent PTP frame reception.

Within each buffer, the entire PTP frame is written in (from MAC Destination Address through to the last byte from the data field), starting at the base address of that buffer. Following PTP frame reception, the "Rx Time Stamp" captured for that frame is written into the top 4 bytes of the buffer used.

Tx PTP Packet Buffer Address Space

The Address space of the "Tx PTP Packet Buffers" is continuous from (PLB_base_address + 0x1000) to (PLB_base_address + 0x17FF), representing the size of a single Virtex-5 FPGA block 18k RAM (2k bytes). Every byte of this block RAM is accessible through the PLB. This address space is divided equally into 8 separate buffers of 256 bytes, each of which is capable of storing a unique PTP frame: 7 of these buffer locations are pre-initialized with standard PTP frame syntax; however, each byte can be modified via the PLB.

Within each single buffer, the initial byte is used as a length field, used to indicate to the core logic the number of bytes to be transmitted for that frame. An entire PTP frame (from MAC Destination Address through to the last byte from the data field) is then stored, starting at the 8th address of that particular buffer. Following PTP frame transmission from one of these buffers, the "Tx Time Stamp" captured for that frame is written into the top 4 bytes of the buffer just used.

Ethernet Audio Video End Point Configuration Registers

Tx PTP Packet Control Register

Table 12 defines associated control register of the "Tx PTP Packet Buffers," used by the "Software Drivers" to request the transmission of the PTP frames.

Table 12: **Tx PTP Packet Buffer Control Register (Address at PLB_base_address + 0x2000)**

Bit Number	Default	Access	Description
7-0	0	WO	**tx_send_frame Bits**. The Tx PTP Packet Buffer is split into 8 regions of 256 bytes, each of which can contain a separate PTP frame. There is 1 tx_send_frame bit for each of the 8 regions. Each bit, when written to '1', causes a request to be made to the "Tx Arbiter." When access is granted the frame contained within the respected region is transmitted. If read, always returns 0.
15-8	0	RO	**tx_frame_waiting Indication**. The Tx PTP Packet Buffer is split into 8 regions of 256 bytes, each of which can contain a separate PTP frame. There is 1 tx_frame_waiting bit for each of the 8 regions. Each bit, when logic 1, indicates that a request has been made for frame transmission to the "Tx Arbiter," but that a grant has not yet occurred. When the frame has been successfully transmitted, the bit is set to logic 0. This bit allows the microprocessor to run off a polling implementation as opposed to the Interrupts.
18-16	0	RO	**tx_packet**. Indicates the number (block RAM bin position) of the most recently transmitted PTP packet.
31-19	0	RO	Unused

Note: A read or a write to this register clears the `cpu_wr_int` interrupt (asserted after each successful PTP packet transmission).

Rx PTP Packet Control Register

Table 13 defines the associated control register of the "Rx PTP Packet Buffers," used by the "Software Drivers" to monitor the position of the most recently received PTP frame.

Table 13: **Rx PTP Packet Buffer Control Register (Address at PLB_base_address + 0x2004)**

Bit Number	Default	Access	Description
0	0	WO	**rx_clear**. When written with a '1,' forces the buffer to empty, in practice moving the write address to the same value as the read address. If read, always returns 0.
7-1	0	RO	Unused
11-8	0	RO	**rx_packet**. Indicates the number (block RAM bin position) of the most recently received PTP packet.
31-12	0	RO	Unused

Note: A read or a write to this register clears the `cpu_rd_int` interrupt (asserted after each successful PTP packet reception).

Rx Filtering Control Register

Table 14 defines the associated control register of the "Receiver Splitter." The Rx path is capable of identifying the AV packets using configurable VLAN priority.

Table 14: **Rx Filtering Control Register (Address at PLB_base_address + 0x2008)**

Bit Number	Default	Access	Description
2-0	3	R/W	**VLAN Priority A**. If a tagged packet is received with a VLAN priority field matching either of the Priority A or B values, then the packet will be considered as an AV frame: it will be passed to the AV I/F. Otherwise it will be passed to the Legacy I/F.
7-3	0	RO	Unused
10-8	2	R/W	**VLAN Priority B**. If a tagged packet is received with a VLAN priority field matching either of the Priority A or B values, then the packet will be considered as an AV frame: it will be passed to the AV I/F. Otherwise it will be passed to the Legacy I/F.
15-11	0	RO	Unused
16	1	R/W	Promiscuous Mode for the MAC Header Filters: If this bit is set to 1, the MAC Header Filter is set to operate in promiscuous mode. All frames are passed to the Legacy Traffic Interface. If set to 0, only frames with matching MAC headers are passed to the Legacy Traffic Interface.
31-17	0	RO	Unused

Tx Arbiter Send Slope Control Register

The SendSlope variable is defined in *IEEE802.1Qav* to be the rate of change of credit, in bits per second, when the value of credit is decreasing (during AV packet transmission). Together with "Tx Arbiter Idle Slope Control Register," "RTC Offset Control Registers," and "RTC Offset Control Registers," these registers define the maximum limit of the bandwidth reserved for AV traffic, as enforced by the "Tx Arbiter." The default values allow the maximum bandwidth proportion of 75% for the AV traffic. See the IEEE specification or the Ethernet AVB Endpoint User Guide (UG492) for further information.

Table 15: **Tx Arbiter Send Slope Control Register (Address at PLB_base_address + 0x200C)**

Bit Number	Default	Access	Description
31-20	0	RO	Unused
19-0	2048	R/W	The value of sendSlope

Tx Arbiter Idle Slope Control Register

The idleSlope variable is defined in *IEEE802.1Qav* to be the rate of change of credit, in bits per second, when the value of credit is increasing (whenever there is no AV packet transmission). Together with "Tx Arbiter Send Slope Control Register," "RTC Offset Control Registers," and "RTC Offset Control Registers," four registers define the maximum limit of the bandwidth reserved for AV traffic: this will be enforced by the "Tx Arbiter." The default values allow the maximum bandwidth proportion of 75% for the AV traffic. See the IEEE specification or the Ethernet AVB Endpoint User Guide (UG492) for further information.

Table 16: **Tx Arbiter Idle Slope Control Register (Address at PLB_base_address + 0x2010)**

Bit Number	Default	Access	Description
31-20	0	RO	Unused
19-0	6144	R/W	The value of idleSlope

RTC Offset Control Registers

Table 17 describes the offset control register for the nano-seconds field of the "RTC," used to force step changes into the counter. When in PTP clock master mode, this can be used to set the initial value following power-up. When in PTP clock slave mode, the "Software Drivers" use this register to implement the periodic step corrections.

This register and the registers defined in Table 18 and in Table 19 are linked. These three offset values will be loaded into the RTC counter logic simultaneously following a write to this nanosecond offset register.

Table 17: **RTC Nano-seconds Field Offset (PLB_base_address + 0x2800)**

Bit Number	Default	Access	Description
29-0	0	R/W	**30-bit offset value for the RTC nano seconds**. Used by the microprocessor to initialize the RTC, then afterwards to perform the regular RTC corrections (when in slave mode).
31-30	0	RO	Unused

Table 18 describes the offset control register for the lower 32-bits of seconds field of the "RTC," used to force step changes into the counter. When in PTP clock master mode, this can be used to set the initial value following power-up. When in PTP clock slave mode, the "Software Drivers" use this register to implement the periodic step corrections.

This register and the registers defined in Table 17 and in Table 19 are linked. These three offset values will be loaded into the RTC counter logic simultaneously following a write to the nanosecond offset register defined in Table 17.

Table 18: **Seconds Field Offset Bits [31:0] (PLB_base_address + 0x2808)**

Bit Number	Default	Access	Description
31-0	0	R/W	**32-bit offset value for the RTC seconds field (bits 31-0)**. Used by the microprocessor to initialize the RTC, then afterwards to perform the regular RTC corrections (when in slave mode).

Table 19 describes the offset control register for the upper 16-bits of seconds field of the "RTC," used to force step changes into the counter. When in PTP clock master mode, this can be used to set the initial value following power-up. When in PTP clock slave mode, the "Software Drivers" use this register to implement the periodic step corrections.

This register and the registers defined in Table 17 and in Table 18 are linked. These three offset values will be loaded into the RTC counter logic simultaneously following a write to the nanosecond offset register defined in Table 17.

Table 19: **Seconds Field Offset Bits [47:32] (PLB_base_address + 0x280C)**

Bit Number	Default	Access	Description
15-0	0	R/W	**16-bit offset value for the RTC seconds field (bits 47-32)**. Used by the microprocessor to initialize the RTC, then afterwards to perform the regular RTC corrections (when in slave mode).
31-16	0	RO	Unused

RTC Increment Value Control Register

Table 20 describes the RTC Increment Value Control Register, which provides a configurable increment rate for the "RTC" counter. This increment register should simply take the value of the clock period being used to increment the RTC; however, the resolution of this increment register is very fine (in units of 1/1048576 ($1/2^{20}$) fraction of one nanosecond) and for this reason the RTC increment rate can be adjusted to a very fine degree of accuracy, providing the following features:

- The RTC can be incremented from any available clock frequency that is greater than the P802.1AS defined minimum of 25 MHz.
- When acting as a clock slave, the rate adjustment of the RTC can be matched to that of the network clock master to an exceptional level of accuracy

Table 20: **RTC Increment Value Control Register (PLB_base_address + 0x2810)**

Bit Number	Default	Access	Description
25-0	0	R/W	Per rtc_clk clock period Increment Value for the RTC.
31-26	0	RO	Unused

Current RTC Value Registers

Table 21 describes the nanoseconds field value register for the nano-seconds field of the "RTC." When read, this returns the latest value of the counter. This register and the registers defined in Table 22 and in Table 23 are linked. When this nanoseconds value register is read, the entire RTC (including the seconds field) is sampled.

Table 21: **Current RTC Nanoseconds Value (PLB_base_address + 0x2814)**

Bit Number	Default	Access	Description
29-0	0	RO	Current Value of the synchronized RTC nanoseconds field. Note: A read from this register will sample the entire RTC counter (synchronized) so that the Epoch and Seconds field will be held static for a subsequent read.
31-30	0	RO	Unused

Table 22 describes the lower 32-bits of the seconds value register for the seconds field of the "RTC." When read, this returns the latest value of the counter. This register and the registers defined in Table 21 and in Table 23 are linked. When the nanoseconds value register is read (see Table 21), the entire RTC is sampled.

Table 22: **Current RTC Seconds Field Value bits [31:0] (PLB_base_address + 0x2818)**

Bit Number	Default	Access	Description
31-0	0	RO	Sampled Value of the synchronized RTC Seconds field (bits 31-0).

Table 23 describes the upper 16-bits of the seconds value register for the seconds field of the "RTC." When read, this returns the latest value of the counter. This register and the registers defined in Table 21 and in Table 22 are linked. When the nanoseconds value register is read (see Table 21), the entire RTC is sampled.

Table 23: **Current RTC Seconds Field Value Bits [47:32] (PLB_base_address + 0x281C)**

Bit Number	Default	Access	Description
15-0	0	RO	Sampled Value of the synchronized RTC Seconds field (bits 47-32).
32-16	0	RO	Unused

RTC Interrupt Clear Register

Table 24 describes the control register defined for the `interrupt_ptp_timer` signal, the periodic interrupt signal which is raised by the "RTC."

Table 24: **RTC Interrupt Clear Register (PLB_base_address + 0x2820)**

Bit Number	Default	Access	Description
0	0	WO	Write ANY value to bit 0 of this register to clear the `interrupt_ptp_timer` Interrupt signal. This bit will always return 0 on read.
31-1	0	RO	Unused

Phase Adjustment Register

Table 25 describes the Phase Adjustment Register which has units of nanoseconds. This value is added to the synchronized value of the RTC nanoseconds field, and the RTC timing signals are then derived from the result. This phase offset is therefore applied to the `clk8k` signal.

As an example, writing the value of the decimal 62500 (half of an 8 KHz clock period) to this register would invert the `clk8k` signal with respect to a value of 0. For this reason, this register can provide fine grained phase alignment of these signals to a 1 ns resolution.

Table 25: **RTC Phase Adjustment Register (PLB_base_address + 0x2824)**

Bit Number	Default	Access	Description
29-0	0	R/W	ns value relating to the phase offset for all RTC derived timing signals (clk8k).
31-30	0	RO	Unused

Software Reset Register

Table 26 describes the Software Reset Register. This register contains unique bits which can be written to in order to request the reset of a particular section of logic from within the Ethernet AVB Endpoint core. A single bit can be written to in a single CPU transaction in order to reset just that particular function; several to all bits can be written to in a single CPU transaction in order to reset several to all of the available reset functions.

Table 26: **Software Reset Register (Address at PLB_base_address + 0x2828)**

Bit Number	Default	Access	Description
0	0	WO	Transmitter path reset. When written with a '1', forces the entire transmitter path of the core to be reset. This also asserts the `tx_reset` signal of Table 1. This reset does not affect transmitter configuration settings. If read, always returns 0.
1	0	WO	Receiver path reset. When written with a '1', forces the entire receiver path of the core to be reset. This also asserts the `rx_reset` signal of Table 1. This reset does not affect receiver configuration settings. If read, always returns 0.
2	0	WO	PTP Transmitter logic reset. When written with a '1', forces the PTP transmitter logic of the core to be reset. This is a subset of the full transmitter path reset of bit 0. This reset does not affect PTP transmitter configuration settings. If read, always returns 0.
3	0	WO	PTP Receiver logic reset. When written with a '1', forces the PTP receiver logic of the core to be reset. This is a subset of the full receiver path reset of bit 1. This reset does not affect PTP receiver configuration settings. If read, always returns 0.
31-4	0	RO	Unused

MAC Header Filter Configuration

Unless the core is generated in pcore format for import into EDK, then "MAC Header Filters" are provided on the receiver path. Please see the Ethernet AVB Endpoint User Guide (UG492) for EDK pcore information.

When the "MAC Header Filters" are not provided, the following address locations will return 0's for a read and all writes will be ignored.

When provided, the "MAC Header Filters" present on the Rx Legacy traffic path are capable of providing match recognition logic against eight unique MAC frame headers. Each of the eight individual filters require eight memory mapped registers to configure them, as defined in Table 27. Each individual filter contains its own set of these eight registers. When interpreting Table 27, the variable *filter#* should be replaced with an integer number between 0 and 7, which represent the eight individual filters.

Table 27: **MAC Header Filter Configuration Registers**

Address	Default	Access	Description
PLB_base_address + 0x3000 + (filter# * 0x20) + 0x0	0xFFFFFFFF	R/W	Match Pattern: Ethernet frame bits 0 to 31 32 bit pattern to match against the Ethernet frame bits 0 to 31. Specifically, match pattern bits: [31:0]: MAC Destination Address Field bits [31:0]
PLB_base_address + 0x3000 + (filter# * 0x20) + 0x4	0x0000FFFF	R/W	Match Pattern: Ethernet frame bits 32 to 63 32 bit pattern to match against the Ethernet frame bits 32 to 63. Specifically, match pattern bits: [15:0]: MAC Destination Address Field bits [47:32] [31:16]: MAC Source Address Field bits [15:0]
PLB_base_address + 0x3000 + (filter# * 0x20) + 0x8	0x00000000	R/W	Match Pattern: Ethernet frame bits 64 to 95 32 bit pattern to match against the Ethernet frame bits 64 to 95. Specifically, match pattern bits: [31:0]: MAC Source Address bits [47:16]
PLB_base_address + 0x3000 + (filter# * 0x20) + 0xC	0x00000000	R/W	Match Pattern: Ethernet frame bits 96 to 127 32 bit pattern to match against the Ethernet frame bits 96 to 127. For frames with a VLAN tag, match pattern bits[31:0] can be matched against the full VLAN field For frames without a VLAN, match pattern bits[15:0] can be matched against the Length/Type field.
PLB_base_address + 0x3000 + (filter# * 0x20) + 0x10	0xFFFFFFFF	R/W	Match Enable: Ethernet frame bits 0 to 31 There is a 1-to-1 correspondence between all bits in this register and all bits in the "Match Pattern: Ethernet frame bits 0 to 31" register. For each bit: • logic 1 enables the match: the corresponding bit in the Match Pattern will be compared • logic 0 disables the match: the corresponding bit in the Match Pattern will be a don't-care.

Table 27: **MAC Header Filter Configuration Registers *(Cont'd)***

Address	Default	Access	Description
PLB_base_address + 0x3000 + (filter# * 0x20) + 0x14	0x0000FFFF	R/W	Match Enable: Ethernet frame bits 32 to 63 There is a 1-to-1 correspondence between all bits in this register and all bits in the "Match Pattern: Ethernet frame bits 32 to 63" register. For each bit: • logic 1 enables the match: the corresponding bit in the Match Pattern will be compared • logic 0 disables the match: the corresponding bit in the Match Pattern will be a don't-care.
PLB_base_address + 0x3000 + (filter# * 0x20) + 0x18	0x00000000	R/W	Match Enable: Ethernet frame bits 64 to 95 There is a 1-to-1 correspondence between all bits in this register and all bits in the "Match Pattern: Ethernet frame bits 64 to 95" register. For each bit: • logic 1 enables the match: the corresponding bit in the Match Pattern will be compared • logic 0 disables the match: the corresponding bit in the Match Pattern will be a don't-care.
PLB_base_address + 0x3000 + (filter# * 0x20) + 0x1C	0x00000000	R/W	Match Enable: Ethernet frame bits 96 to 127 There is a 1-to-1 correspondence between all bits in this register and all bits in the "Match Pattern: Ethernet frame bits 96 to 127" register. For each bit: • logic 1 enables the match: the corresponding bit in the Match Pattern will be compared • logic 0 disables the match: the corresponding bit in the Match Pattern will be a don't-care.

Tri-Mode Ethernet MAC Address Space

Unless the core is generated in pcore format for import into EDK, then the address space of the Ethernet MAC is incorporated into the address space of the Ethernet AVB Endpoint core as illustrated in Figure 10. When not included, this address space will return 0s for a read and all writes will be ignored.

When the address space of the Ethernet MAC is incorporated, the MAC Address space is split into two sections:

- "MAC Configuration and Statistics"
- "MAC MDIO Registers"

MAC Configuration and Statistics

Table 28 defines the statistic registers and configuration registers of the Tri-Mode Ethernet MAC core. These are listed with their assigned addresses. See the Tri-Mode Ethernet MAC User Guide (UG138) and Ethernet Statistics User Guide (UG170) for further descriptions of these registers.

Table 28: **Configuration Registers**

Address	Description
(PLB_base_address + 0x4000) to (PLB_base_address + 0x41FF)	The Ethernet Statistics core is capable of providing up to 64 configurable statistics for the Ethernet MAC (let these be numbered by STATISTIC_NUMBER, from 0 to 63). Each statistic will return a 64-bit counter value. Therefore Address of STATISTIC_NUMBER = (PLB_base_address + 0x4000 + [STATISTIC_NUMBER * 8])
PLB_base_address + 0x5000	Receiver Configuration (Word 0)
PLB_base_address + 0x5200	Receiver Configuration (Word 1)
PLB_base_address + 0x5400	Transmitter Configuration
PLB_base_address + 0x5600	Flow Control Configuration
PLB_base_address + 0x5800	MAC Speed Configuration
PLB_base_address + 0x5A00	Management Configuration

MAC Address Filter Registers

The Address Filter, optionally present in the Tri-Mode Ethernet MAC core, must not used. Instead, new "MAC Header Filters" have been added to the Receiver Legacy Traffic path, which is capable of providing match recognition logic for eight unique MAC frame headers. For additional information, see "MAC Header Filter Configuration."

MAC MDIO Registers

The Tri-Mode Ethernet MAC has MDIO master capability. To access an MDIO register via the Ethernet MAC, construct the address as follows:

```
MDIO register address = PLB_base_address + 0x6000 + (MDIO_ADDRESS *8)
```

where `MDIO_ADDRESS` is a 10-bit binary address, constructed from the 5-bit MDIO Physical Address (PHYAD) and the 5-bit MDIO Register Address (REGAD) as follows:

```
MDIO_ADDRESS <= {PHYAD, REGAD}
```

See the Tri-Mode Ethernet MAC User Guide (UG138) and IEEE802.3 Specification for further MDIO information.

Device Utilization

Table 29 provides approximate utilization figures for the core. These figures were obtained by implementing the core in a Virtex-6 and a Spartan-3A DSP device; other device families have similar utilization figures as indicated.

When the core is generated in pcore format for import into the EDK, it is intended to be connected to the XPS LocalLink Tri-Mode Ethernet MAC (xps_ll_temac) core. This core can optionally contain its own address filter and PLB Configuration logic and so this functionality is omitted from the Ethernet AVB Endpoint core; consequently, the AVB core uses less device resources in this configuration.

When the core is not generated in pcore format, it is intended to be connected to the LogiCORE IP Tri-Mode Ethernet MAC (TEMAC) or an Embedded Tri-Mode Ethernet MAC. The AVB Endpoint core is then generated to include the "MAC Management Interface" and the "MAC Header Filters".

Table 29: **Device Utilization**

Parameter Values		**Device Resources**				
Family	**EDK pcore format**	**Slices**	**LUTs**	**FFs**	**Block RAMs**	**DSP / Multipliers**
Virtex-6, Virtex-6 -1L, Virtex-5, Spartan-6	No	710	2060	1910	2	4
	Yes	490	1490	1420	2	4
Spartan-3, Spartan-3E, Spartan-3A, Spartan-3ADSP	No	1640	1940	1860	3	7
	Yes	1120	1260	1440	3	7

Support

Xilinx provides technical support for this LogiCORE IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY.*

Ordering Information

The Ethernet AVB Endpoint core is provided under the Xilinx Core Site License and can be generated using the ISE CORE Generator system, which is included with the Xilinx ISE Design Suite software.

Two free evaluation licenses are available:

- The Simulation Only license is provided with the CORE Generator ISE Design Suite software.
- The Full System Hardware Evaluation license, which lets you test your designs in hardware for a limited period of time, can be accessed from the "Evaluate" link on the Ethernet AVB Endpoint product page. (www.xilinx.com/products/ipcenter/DO-DI-EAVB-EPT.htm)

For full access to all core functionality, both in simulation and in hardware, you must purchase a license for the core.

Please contact your local Xilinx sales representative for pricing and availability of Xilinx LogiCORE IP modules and software. Information about additional Xilinx LogiCORE IP modules is available on the Xilinx Intellectual Property page IP Center.

References

Ethernet AVB Endpoint User Guide (UG492)

Tri-Mode Ethernet MAC User Guide (UG138)

Ethernet Statistics User Guide (UG170)

IEEE802.3 Specification

EDK documentation

List of Acronyms

The following table describes acronyms used in this manual.

Acronym	Spelled Out
AV	Audio Video
AVB	Audio Video Bridging
BMCA	Best Master Clock Algorithm
EDK	Embedded Development Kit
FF	flip-flop
FPGA	Field Programmable Gate Array.
Gbps	Gigabits per second
GUI	Graphical User Interface
HDL	Hardware Description Language
I/F	Interface
I/O	Input Output
IES	Incisive Enterprise Simulator
IP	Intellectual Property
ISE	Integrated Software Environment
LUT	Look Up Table
MAC	Media Access Controller
Mbps	Megabits per second
MDIO	Management Data Input/Output
PLB	Processor Local Bus
PTP	Precise Timing Protocol
R/W	Read/Write
RAM	Random Access Memory
RO	Read Only
RTC	Real Time Clock
Rx	Receive
SRP	Stream Reservation Protocol
TEMAC	Tri-Mode Ethernet MAC
Tx	Transmit

Acronym	Spelled Out
VHDL	VHSIC Hardware Description Language (VHSIC an acronym for Very High-Speed Integrated Circuits)
VLAN	Virtual LAN (Local Area Network)

Revision History

Date	Version	Revision
9/18/08	1.1	Initial Xilinx release.
10/7/08	1.1.1	LUTs and FF values adjusted.
4/24/09	1.2	Update core to version 1.2; Xilinx tools v11.1; Added Spartan-3, Spartan-3E and Spartan-3A devices; Mentor Graphics ModelSim v6.4b; Cadence IUS v8.1-s009; Synopsys 2008.09.
6/24/09	2.1	Update core to version 2.1; Xilinx tools v11.2; Added support for Virtex-6 and Spartan-6 devices.
9/16/09	2.2	Update core to version 2.2; Xilinx tools v11.3; Added support for Virtex-6 HXT and Virtex-6 Lower Power devices. Added pcore generation support.
4/19/10	2.3	Update core to version 2.3; Xilinx tools v12.1.
7/23/10	2.4	Update core to version 2.4; Xilinx tools v12.2.

Notice of Disclaimer

Xilinx is providing this product documentation, hereinafter "Information," to you "AS IS" with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.