Tannhäuser Gate IP Core
(MAC PTP BRIDGE)
Datasheet

Version 1.0

Revision History

Revision	Date	Description
1.0	18/10/25	Initial Version

Copyright reminder

Trademarks:

Xilinx®, Kintex®, and TEMAC® are registered trademarks of Advanced Micro Devices, Inc. LabVIEW® and PXI® are registered trademarks of National Instruments Corporation. All other product names and brands mentioned are the property of their respective owners.

Table of Contents

1. Introduction

The MAC PTP Bridge (Tannhäuser Gate) IP core serves as a high-performance bridge between the Xilinx TEMAC IP Core, SOCe's 1588 Tiny IP Core, and the Xilinx Ethernet PHY IP Core. It features intelligent arbitration of incoming Ethernet frames from both the TEMAC and Tiny IP cores, ensuring lossless transmission to the Ethernet PHY. This architecture enables precise time-stamping and deterministic data flow for time-sensitive networking applications. The block diagram of the IP core is shown in Figure 1.1.

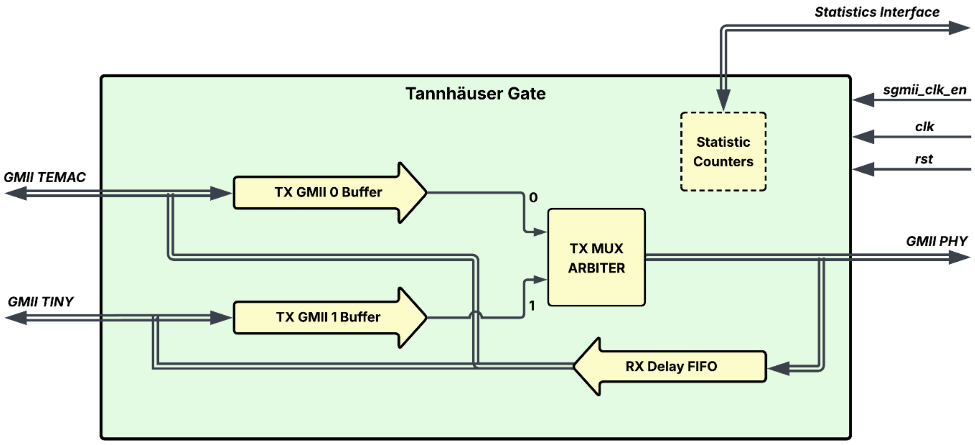


Figure 1-1 Block Diagram of Tannhäuser Gate IP Core

The MAC PTP Bridge (Tannhäuser Gate) IP core features three GMII interfaces, each serving a distinct role in the Ethernet communication pipeline:

- **GMII TEMAC (Slave)**
  Connected to the GMII Master interface of the **Xilinx Tri-Mode Ethernet MAC (TEMAC) IP core.** This interface handles general-purpose Ethernet traffic and may utilize most of the available bandwidth.
- **GMII TINY (Slave)**

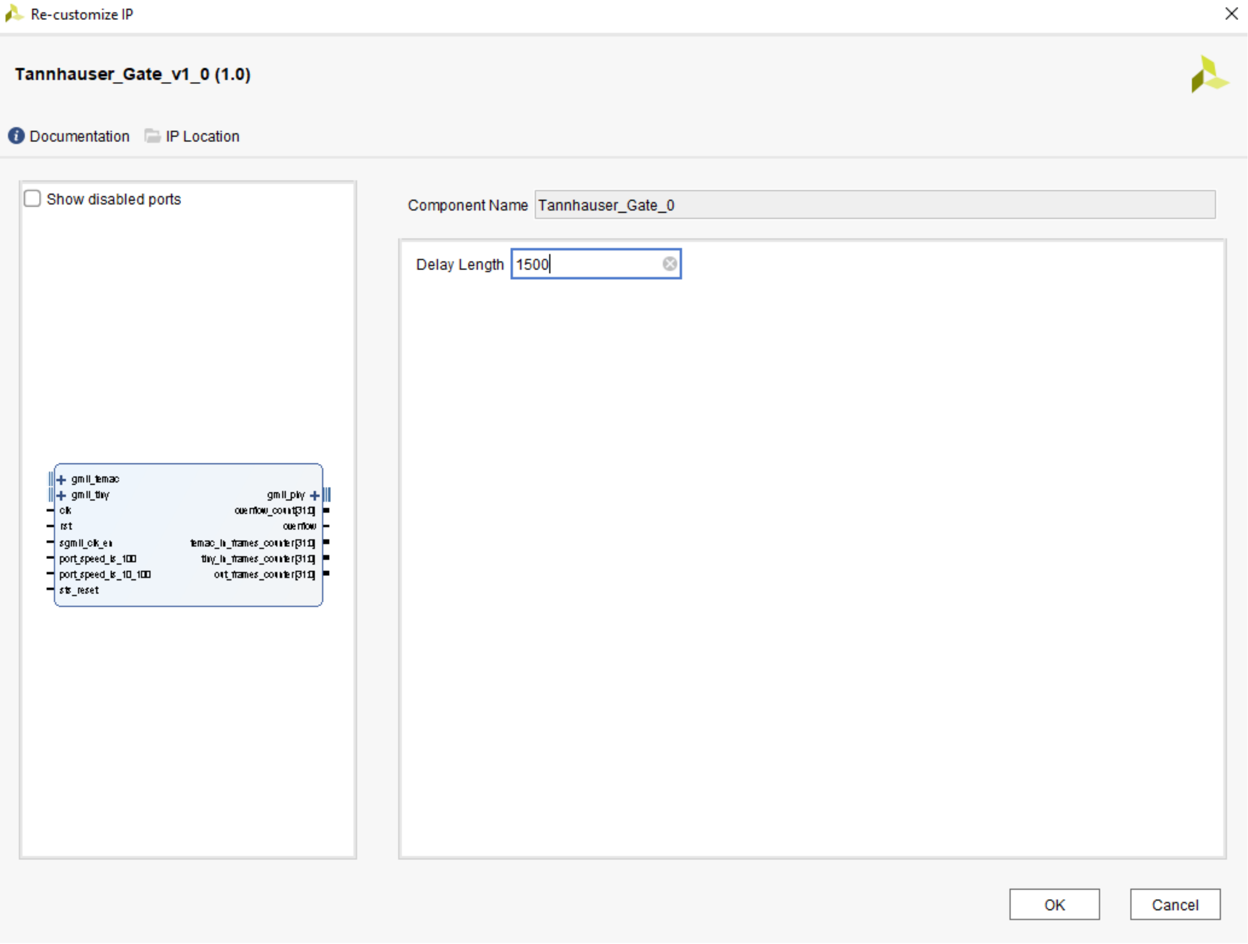
  Connected to the GMII Master interface of **SOCe's 1588 Tiny IP Core**, which operates as a PTP Ordinary Clock (OC). This interface handles low-bandwidth, time-sensitive traffic, typically up to 16 frames per second.

- **GMII PHY (Master)**

  Connected to the GMII Slave interface of the Xilinx 1G Base-X or SGMII Ethernet PHY IP Core. This interface transmits the arbitrated and lossless Ethernet frames to the physical layer.

Statistics Interface provides statistical information about the frames passed through each interface and status about the overflow condition.

Figure 1-2 illustrates the IP Core Customization Window as presented in the Vivado IDE:



Re-customize IP

Tannhauser_Gate_v1_0 (1.0)

Component Name: Tannhauser_Gate_0

Delay Length: 1500

Figure 1-2 Tannhauser Gate IP Core customization in Vivado

The *delay_length* parameter defines the internal buffering delay required to ensure proper alignment and lossless transmission of Ethernet frames. This parameter is critical for maintaining timing integrity between the MAC, PTP, and PHY interfaces.

To achieve optimal performance and avoid alignment errors, the delay_length should be configured as *delay_length* = *maximum_frame_size* + *interframe_gap*. This value ensures that the arbitration logic has sufficient buffer space to accommodate the largest possible frame along with the mandatory interframe gap, preventing data loss or timing violations during high-throughput operation. Further details on how this parameter influences frame scheduling and arbitration are provided in the Algorithm chapter.

2. Algorithm

The MAC PTP Bridge IP core employs a deterministic arbitration mechanism to manage frame transmission between the TEMAC and TINY IP cores toward the PHY interface. The arbitration logic ensures lossless delivery and proper alignment of frames, particularly when handling time-sensitive PTP traffic.

Arbitration Behavior

- **TEMAC-Only Operation**

  When the **Xilinx TEMAC IP core** is actively transmitting and **no frame is present from the TINY IP core**, the arbiter **immediately forwards TEMAC frames** to the PHY interface without delay. This guarantees **maximum throughput** and **minimal latency** for general-purpose Ethernet traffic.

- **TINY Frame Detected**

  Upon detecting a frame on the **GMII TINY interface**, the arbiter transitions to a controlled arbitration sequence:

  1. **Tiny Frame Buffering -** The incoming TINY frame is buffered into the **TX GMII 1 Buffer**, and the arbiter initiates a **delay counter** set to the value of delay_length.
  2. **TEMAC Frame Completion -** If a TEMAC frame is currently being transmitted, the arbiter allows it to complete into the **TX GMII 0 Buffer**. **No further TEMAC frames are forwarded** during this delay period; instead, they are buffered.
  3. **Tiny Frame Forwarding -** Once the delay counter reaches the configured delay_length, the buffered TINY frame in **TX GMII 1 Buffer** is forwarded to the **GMII PHY interface**.
  4. **Resumption of TEMAC Transmission -** After the TINY frame is transmitted, the arbiter resumes forwarding buffered TEMAC frames from **TX GMII 0 Buffer** to the PHY.

This arbitration strategy ensures that **PTP frames are prioritized** without compromising the integrity or throughput of TEMAC traffic. The delay_length parameter plays a critical role in aligning frame timing and avoiding transmission conflicts.

After each frame transmission, the arbiter enforces an interframe gap by counting 12 clock cycles, equivalent to 96 bit times, before forwarding the next frame. This guarantees compliance with Ethernet timing requirements and prevents frame collisions.

Interframe Gap Enforcement

After each frame transmission, the arbiter enforces an interframe gap by counting 12 clock cycles, equivalent to 96 bit times, before forwarding the next frame. This guarantees compliance with Ethernet timing requirements and prevents frame collisions.

Deterministic Delay for TINY Frames

When a frame from the TINY IP core is detected, the arbiter initiates a delay counter set to the configured delay_length. During this period:

- If no TEMAC frame is currently being transmitted, the arbiter holds further TEMAC frames, buffering them in the **TX GMII 0 Buffer**.
- Once the delay counter reaches delay_length, the buffered TINY frame in TX GMII 1 Buffer is forwarded to the *GMII PHY interface*.
- After the TINY frame is transmitted, the arbiter resumes forwarding buffered TEMAC frames.

This mechanism ensures a constant transmission delay for TINY frames, calculated as:

$$\textit{TINY Frame Delay} = \textit{delay\_length} \times \textit{clock period} + \textit{external delay}$$

Variable Delay for TEMAC Frames

The delay experienced by TEMAC frames may vary depending on the arbitration state:

- If no TINY frame is present, TEMAC frames are forwarded immediately.
- If a TINY frame is being processed, TEMAC frames are buffered and transmitted after the TINY frame completes, introducing variable delay.

Receive Path Delay Compensation

To align incoming frames from the PHY interface with the internal arbitration timing, the IP core includes an **RX Delay FIFO**. This FIFO introduces a fixed delay equal to:

$$\textit{RX Delay} = \textit{delay\_length} \times \textit{clock period} + \textit{external delay}$$

This ensures that all incoming frames from the PHY experience a consistent delay, matching the transmission path and maintaining synchronization across the system.

3. Port Description

The ports of Tannhauser IP Core are described in table 3-1, 3-2 and 3-3.

Table 3-1 General purpose ports

Name	Direction	Description
clk	In	Clock Source (userclk2_out of PHY)
sgmii_clk_en	In	Clock Enable for internal logic (generated by PHY)
rst	In	Resets entire IP core
Port_speed_is_100	In	PHY is configured in 100Mbit/s rate (Unused)
Port_speed_is_10_100	In	PHY is configured in 100 or 10 Mbit/s rate (Unused)

Table 3-2 GMII Interfaces ports

Name	Direction	Description
rx_data_tiny[7:0]	In	gmii tx data of Tiny IP Core
rx_dv_tiny	In	gmii tx data valid signal of Tiny IP Core
rx_data_temac[7:0]	In	gmii tx data of TEMAC IP Core
rx_dv_temac	In	gmii tx data valid signal of TEMAC IP Core
gmii_rxd_in[7:0]	In	gmii rx data of PHY
gmii_rx_dv_in	In	gmii rx data valid signal of PHY
gmii_rxd_out [7:0]	Out	gmii rx data of Tiny and TEMAC IP Cores
gmii_rx_dv_out	Out	gmii rx data valid signal of Tiny and TEMAC IP Cores
tx_data_com[7:0]	Out	gmii tx data of PHY
tx_en_com	Out	gmii tx data valid signal of PHY

Table 3-3 Statistics Interface ports

Name	Direction	Description
overflow	Out	Latches, when an overflow occurs in *TX GMII 0 Buffer*
overflow_count [31:0]	Out	Number of bytes entered in *TX GMII 0 Buffer* during Overflow
temac_in_frames_counter [31:0]	Out	Number of frames entered in from *GMII TEMAC* interface
tiny_in_frames_counter [31:0]	Out	Number of frames entered in from *GMII TINY* interface
out_frames_counter [31:0]	Out	Number of frames transmitted to *GMII PHY* interface
sts_reset	In	Resets statistic counters and overflow flag

4. Testing and Validation

To evaluate the performance and validate the functionality of the IP core, the configuration shown in Figure 4-1 is used:

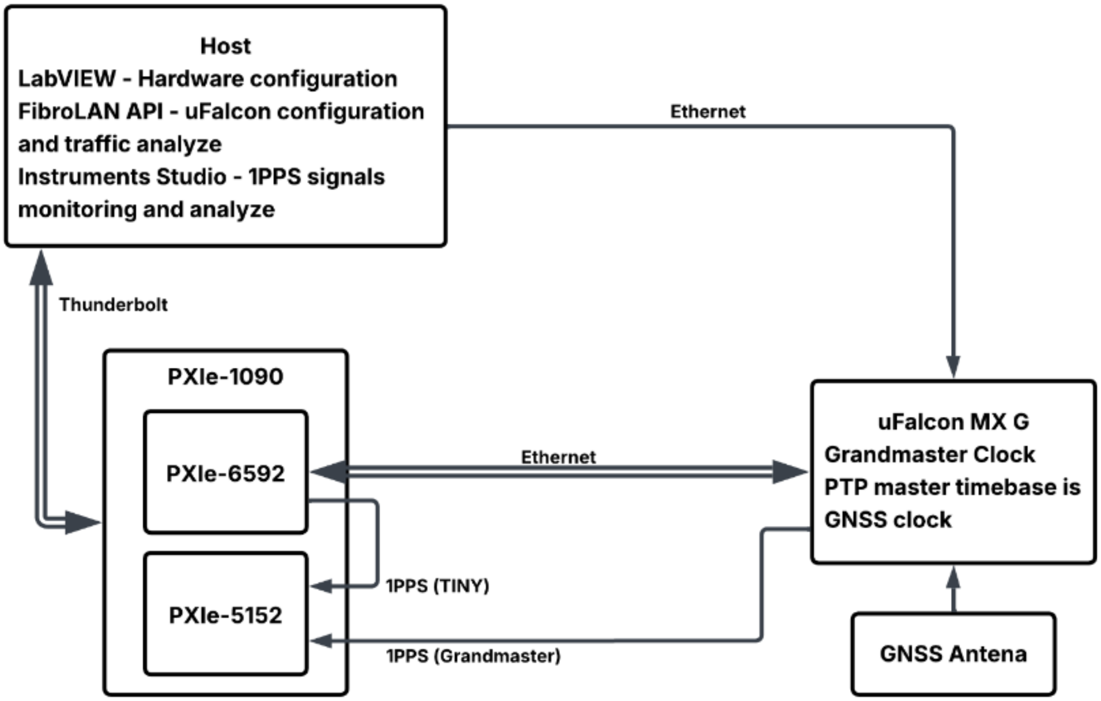

Host
LabVIEW - Hardware configuration
FibroLAN API - uFalcon configuration and traffic analyze
Instruments Studio - 1PPS signals monitoring and analyze
Ethernet
Thunderbolt
PXIe-1090
PXIe-6592
PXIe-5152
Ethernet
uFalcon MX G
Grandmaster Clock
PTP master timebase is GNSS clock
1PPS (TINY)
1PPS (Grandmaster)
GNSS Antena

Figure 4-1 Test System architecture

Test Setup Equipment:

- **PXIe-1090** – 2-slot PXI Express chassis by National Instruments, providing high-performance modular instrumentation support.
- **PXIe-6592** – High-speed serial instrument from National Instruments, featuring an embedded Xilinx Kintex-7 410T FPGA for flexible digital interfacing and protocol emulation.
- **PXIe-5152** – Dual-channel high-speed digitizer (oscilloscope) by National Instruments, used for signal capture and analysis.
- **uFalcon MX G** – Grandmaster clock source manufactured by FibroLAN, used for precise IEEE 1588 PTP synchronization. Additionally, it provides statistical information and supports throughput calculation during performance evaluation.

The **uFalcon MX G** can be configured via the FibroLAN API through a web browser interface. It allows users to configure Ethernet ports, monitor real-time network traffic, and set up Precision Time Protocol (PTP) clocks on each port. In addition to acting as a Grandmaster clock, it also provides statistical data and supports throughput calculation, making it essential for performance validation.

The **PXIe-5152** oscilloscope is controlled using **NI InstrumentStudio™**, which offers an intuitive graphical interface for configuring and managing acquisition parameters without requiring LabVIEW programming knowledge.

The PXIe-6592 is a high-speed serial instrument from National Instruments, equipped with a Xilinx Kintex-7 410T FPGA. This FPGA provides a flexible and reconfigurable platform for implementing custom digital logic, protocol emulation, and high-speed data processing. It serves as a key component in the test setup, enabling precise control and real-time interaction with the IP core under evaluation.

LabVIEW FPGA is used for developing custom FPGA logic in a graphical environment, eliminating the need for traditional HDL coding. The graphical design is automatically synthesized into HDL and integrated into the standard FPGA toolchain. Furthermore, designers can incorporate custom logic developed in third-party EDA tools or written in HDL using IP Integration Nodes (IPINs) or Socketed Component-Level IP (CLIP) interfaces.

The FPGA design used for testing the **Tannhauser Gate IP core** on the PXIe-6592 is illustrated in Figure 4-2:

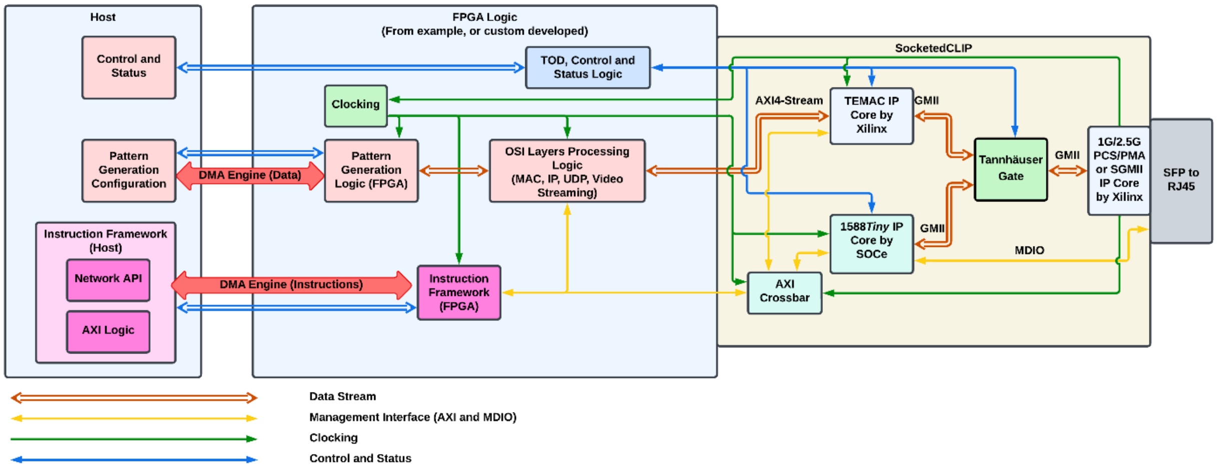

Host
Control and Status
Pattern Generation Configuration
Instruction Framework (Host)
Network API
AXI Logic
DMA Engine (Data)
DMA Engine (Instructions)
FPGA Logic
(From example, or custom developed)
TOD, Control and Status Logic
Clocking
Pattern Generation Logic (FPGA)
OSI Layers Processing Logic
(MAC, IP, UDP, Video Streaming)
Instruction Framework (FPGA)
SocketedCLIP
AXI4-Stream
TEMAC IP Core by Xilinx
GMII
Tannhäuser Gate
GMII
1G/2.5G PCS/PMA or SGMII IP Core by Xilinx
SFP to RJ45
1588*Tiny* IP Core by SOCe
GMII
MDIO
AXI Crossbar
Data Stream
Management Interface (AXI and MDIO)
Clocking
Control and Status

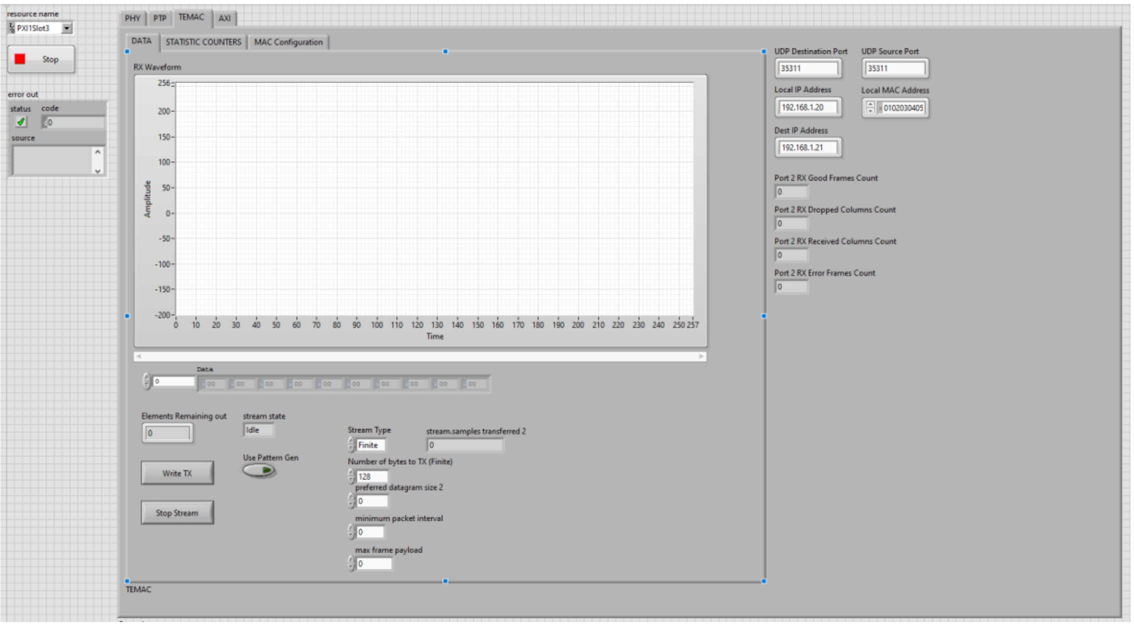
Figure 4-2 PXIe-6592 Application Design

FPGA Design and Host Interaction:

The **UDP**, **IP**, and **pattern generation logic** are implemented using **LabVIEW FPGA**, enabling rapid development through graphical programming. Meanwhile, the **Tannhäuser Gate**, TINY (SOCe 1588 IP Core), **Xilinx TEMAC**, and **Base-X/SGMII IP cores** are integrated into the design using **Socketed Component-Level IP (CLIP)**, allowing seamless inclusion of HDL-based modules within the LabVIEW FPGA environment.

An **Instructions Framework** is employed to facilitate host-side access to internal registers of the IP cores via the AXI interface, enabling dynamic configuration and monitoring during runtime. Additionally, the PPS (Pulse Per Second) output signal from the TINY IP core is routed to the PFI2 port of the PXIe-6592, allowing precise timing analysis and synchronization.

The **host system** uses **LabVIEW** to deploy the compiled FPGA bitfile to the **High-Speed Serial (HSS)** device, configure the pattern generation process, initialize IP core parameters, and monitor system status in real time. The **LabVIEW Main VI Front Panel**, shown in Figure 4-3, provides a user-friendly interface for controlling and observing the system's operation.



resource name: PXI1Slot3

error out
status: ✓
code: 0
source:

Data: 0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00

Elements Remaining out: 0

stream state: Idle

Use Pattern Gen

Stream Type: Finite

stream.samples transferred 2: 0

Number of bytes to TX (Finite): 128

preferred datagram size 2: 0

minimum packet interval: 0

max frame payload: 0

UDP Destination Port: 35311

UDP Source Port: 35311

Local IP Address: 192.168.1.20

Local MAC Address: 0102030405

Dest IP Address: 192.168.1.21

Port 2 RX Good Frames Count: 0

Port 2 RX Dropped Columns Count: 0

Port 2 RX Received Columns Count: 0

Port 2 RX Error Frames Count: 0

Figure 4-3 LabVIEW MAIN VI Front Panel

Pattern Generation Logic Configuration:

The **Pattern Generation logic** implemented in the FPGA includes several configurable parameters that allow flexible control over traffic generation:

- User Pattern Gen – When enabled, the FPGA asserts the data valid signal from downstream nodes. If disabled, data validity depends on the presence of data in the DMA FIFO sent from the host.
- Stream Type – Specifies whether the stream is Continuous or Finite. In the scenarios described below, a Continuous stream is selected.
- Preferred Datagram Size – Defines the preferred size of data chunks to encapsulate in each frame. This parameter is used when forming Ethernet frames.
- Minimum Packet Interval – Sets the minimum time interval between two consecutive frames.
- Maximum Payload Length – Specifies the maximum size of the payload (i.e., the largest data chunk per frame). A typical value is 1472 bytes, based on a 1518-byte Ethernet frame size and a 46-byte header.
- Number of Bytes – Used in Finite stream mode to define the total number of bytes to transmit.

In **Continuous** stream mode, data is continuously passed to upstream nodes until explicitly stopped via the API.

The relationship between **Minimum Packet Interval** and **Preferred Datagram Size** directly affects the effective bandwidth. For example, if the datagram size is **466 bytes** and the minimum packet interval is 2048 clock cycles, the TEMAC bandwidth utilization is calculated as:

$(466+46)\times100/2048\approx25\%$

The **FibroLAN API** includes a Throughput Calculation tab, which visualizes this relationship and assists in optimizing transmission parameters. An example of this interface is shown in Figure 4-4:

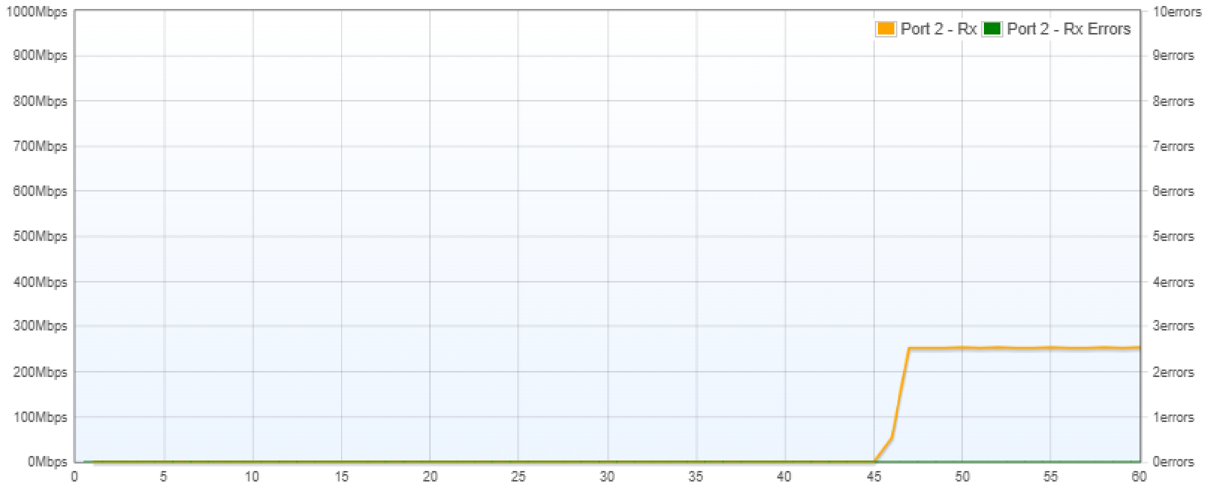


Figure 4-4: 25% bandwidth utilization example.

The **TINY IP core** is configured to transmit 16 *delay_request* messages per second, representing a worst-case scenario. In the absence of pattern generation logic, this results in a bandwidth utilization of only 4096 kbps.

Scenario 1:

- **Prefered Datagram Size** : 1472 bytes
- **Minimum Packet Interval** : 15,180
- **Bandwidth Utilization**: 10%

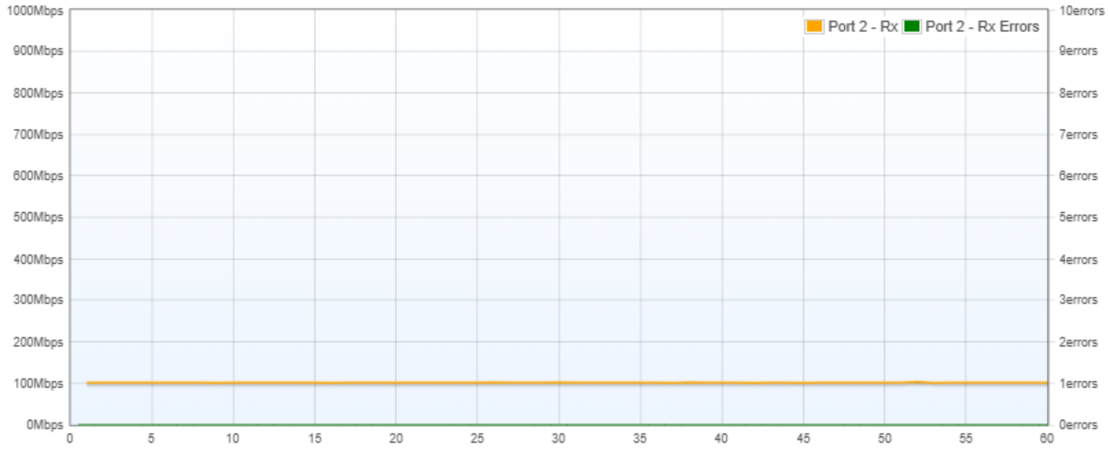


Figure 4-5 Scenario 1, 1min continuous test

Scenario 2:

- **Prefered Datagram Size** : 1472 bytes
- **Minimum Packet Interval** : 4554
- **Bandwidth Utilization**: 33%

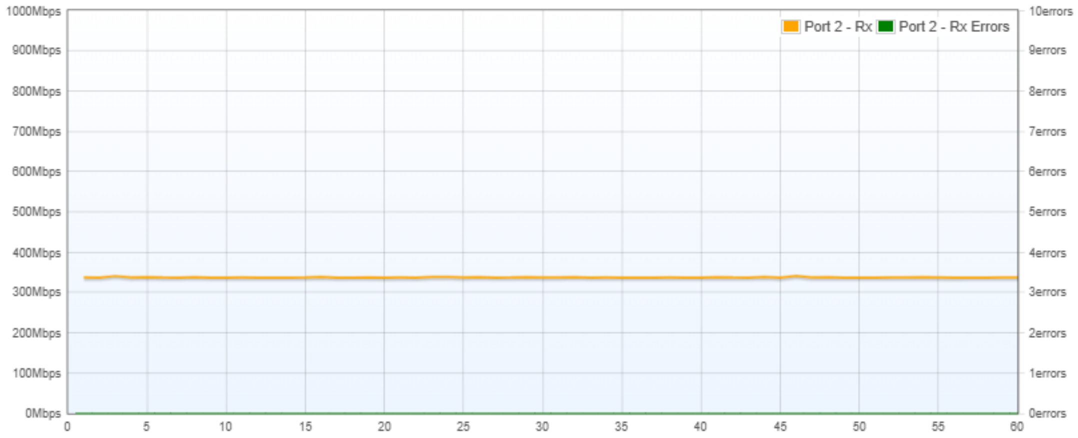


Figure 4-6 Scenario 2, 1min continuous test

Scenario 3:

- **Prefered Datagram Size** : 1472 bytes
- **Minimum Packet Interval** : 1687
- **Bandwidth Utilization**: 90

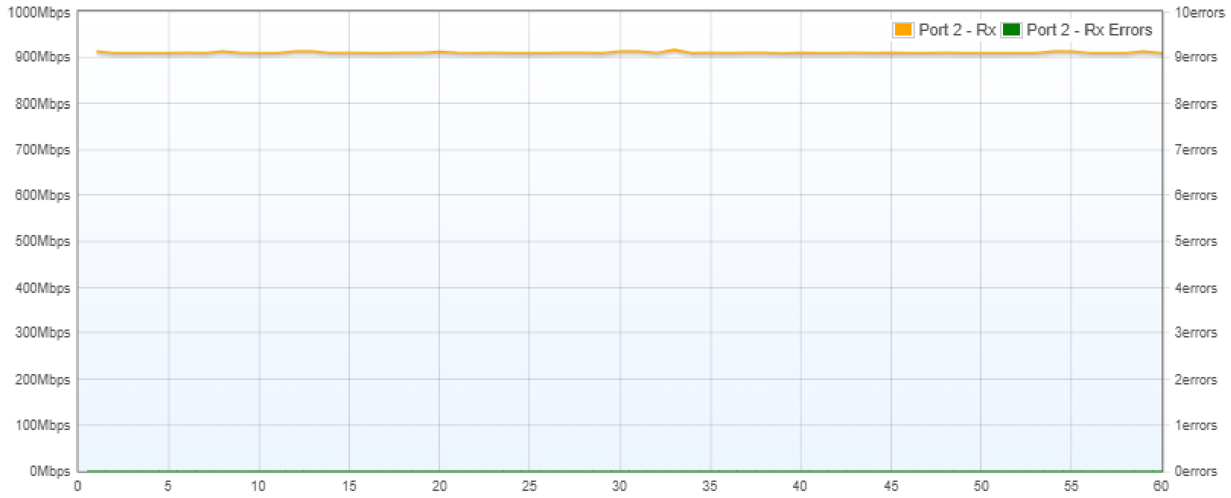


Figure 4-7 Scenario 3, 1min continuous test

Scenario 4:

- **Prefered Datagram Size** : 1472 bytes
- **Minimum Packet Interval** : 1550
- **Bandwidth Utilization**: 98%

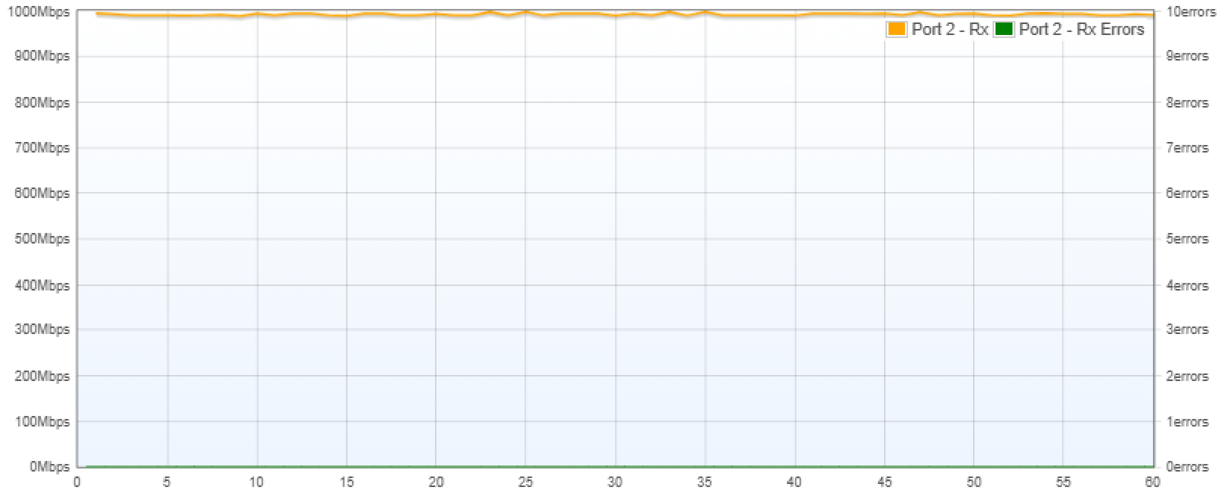


Figure 4-8 Scenario 4, 1min continuous test

Scenario 5:

- **Prefered Datagram Size** : 1472 bytes
- **Minimum Packet Interval** : 1520
- **Bandwidth Utilization**: 99%

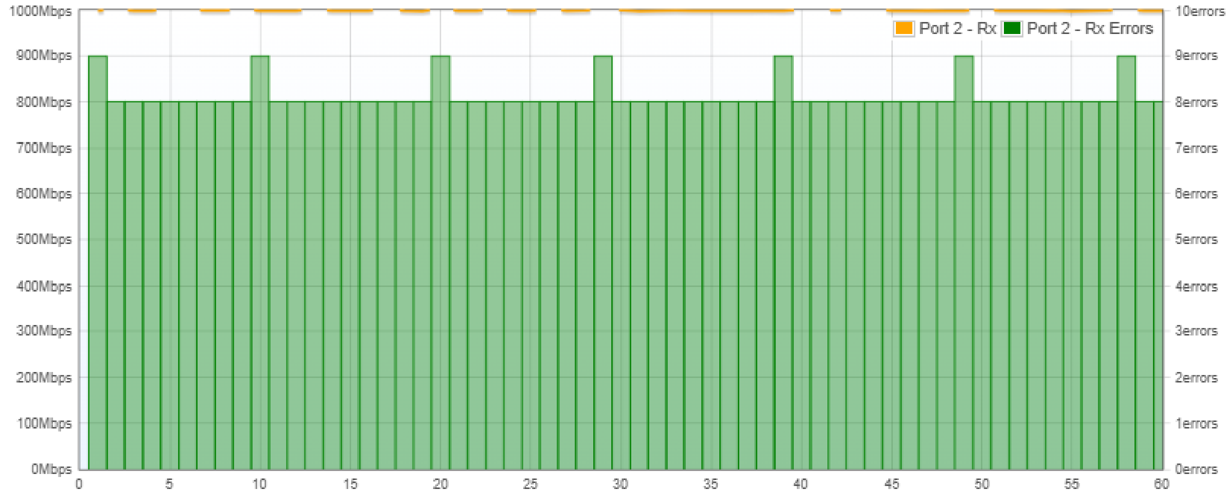
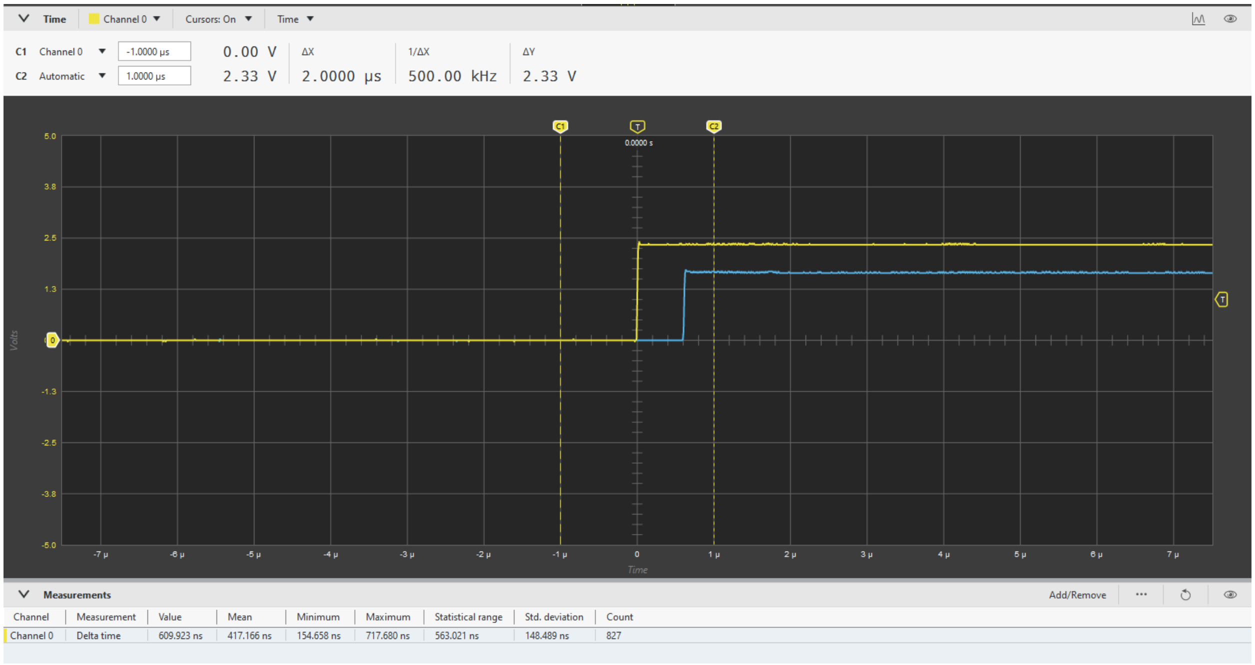


Figure 4-9 Scenario 5, 1min continuous test

Figure 4-10 illustrates the alignment of PPS (Pulse Per Second) signals from the Grandmaster clock (yellow trace) and the TINY IP core (blue trace) across five distinct test scenarios.



					ΔX	1/ΔX	ΔY
C1	Channel 0	-1.0000 µs	0.00 V				
C2	Automatic	1.0000 µs	2.33 V		2.0000 µs	500.00 kHz	2.33 V

Channel	Measurement	Value	Mean	Minimum	Maximum	Statistical range	Std. deviation	Count
Channel 0	Delta time	609.923 ns	417.166 ns	154.658 ns	717.680 ns	563.021 ns	148.489 ns	827

Figure 4-10 PPS Alignment and Statistics

The statistical data displayed at the bottom of Figure 4-10 confirms that synchronization between the Grandmaster and the TINY IP core is achieved at a sub-microsecond level.

Based on the test results, the IP core demonstrates the capability to utilize up to 98% of the Xilinx TEMAC bandwidth while maintaining high-precision synchronization with the PTP Grandmaster.

5. Resources

The IP core was developed targeting the **Xilinx Kintex-7 XC7K410T-2FFG900** FPGA. The post-synthesis resource utilization is summarized in Table 5-1.

Table 5-1 Utilization

Resource Name	LUT	Register	F7 Mux	F8 Mux	BRAM	DSPs
Usage	2790	525	190	90	0.5	0