Proposed Text for Annex 90A

Richard Tse, Microchip Technology

Steve Gorshe, Microchip Technology

Marek Hajduczenia, Charter Communications

Presented at P802.3cx teleconference, May 25, 2021

Supporters

- Andras de Koos, Microchip Technology
- Clark Carty, Cisco Systems
- Denny Wong, Xilinx
- Mark Bordogna, Intel
- Sriram Natarajan, Cisco Systems
- Ulf Parkholm, Ericsson

Annex 90A (informative) Timestamping Accuracy Considerations

90A.1 High Accuracy Timestamping Introduction

This annex provides information on supporting high accuracy timestamping for time synchronization protocol (TimeSync) Client implementations compliant with Clause 90. This timestamping may be used for time synchronization protocols including IEEE Std 1588 and IEEE Std 802.1AS.

90A.2 High Accuracy Timestamping Background

Ethernet support for time synchronization protocols (Clause 90) was not originally specified to support high accuracy timestamping. Thus, implementation flexibility permitted by this standard prior to the addition of certain registers (IEEE Std 802.3cx support, Timestamp reference, first symbol after SFD, Multilane support, and TX/RX num_unit_change support registers shown in Table 45-235 and the fine resolution path data delay registers located throughout subclauses 45.2.1 to 45.2.6) could lead to timestamp accuracy impairments that might not satisfy high accuracy timing requirements.

Timestamping accuracy can be impaired when two TimeSync Clients do not account for a varying physical layer device (PHY) path data delay in the same manner. Examples of PHY functions that cause variation in the PHY path data delay include alignment marker (AM) or codeword marker (CWM) insertion/removal, Idle insertion/removal, and multi-physical coding sublayer (PCS) lane distribution/merging.

Timestamping accuracy can also be impaired when two TimeSync Clients do not use the same message timestamp point. As specified in 90.7, this standard gives two options for the message timestamp point (the beginning of the start of frame delimiter, the SFD, and the beginning of the first symbol after the SFD) but recommends using the beginning of the first symbol after SFD, which is consistent with IEEE Std 1588 and IEEE Std 802.1AS.

Table 90A-1 shows the magnitude of potential timestamp accuracy impairments that could be generated by the aforementioned causes.

Table 90A-1 – Magnitude of Potential Timestamp Accuracy Impairments

Ethernet Rate	Magnitude of Potential Timestamp Accuracy Impairments per Transmit or Receive Port (ns)			
	Mismatched Message Timestamp Point[1]	Idle Insertion/ Removal[2,3]	AM/CWM Insertion/ Removal[3]	PCS Lane Distribution/ Merging
10M	800	400	N/A	N/A
100M	80	40	N/A	N/A
1G	8	16[4], 8[5]	N/A	0[5], N/A[4, 6,7]
2.5G	3.2	12.8	N/A	N/A[7]
5G	1.6	6.4	N/A	N/A[7]
10G	0.8	3.2	N/A	N/A[4], 0[5]
25G	0.32	1.28	10.24	N/A
40G	0.2	1.6	6.4	4.8
100G	0.08	0.64	12.8	12.16
200G	0.04	0.32	2.56	N/A[7]
400G	0.02	0.16	2.56	N/A[7]

Notes:

1. The value shown only accounts for the time between the two message timestamp point options when they are adjacent. See Annex 90A.3 for other factors that can affect some of these values.
2. The value shown corresponds to the effect of a single Idle insertion/removal.
3. The path data delay of a TimeSync message is only affected when the message coincides with an AM, CWM, or Idle insertion/removal event.
4. For 1000Base-X or 10GBase-R
5. For 1000Base-T or 10GBase-X
6. For 10GBase-T
7. For these rates, the lane distribution/merging operation belongs only to the forward error correction (FEC) function and not to the PCS function. The FEC lane distribution/merging operation is not subject to potential timestamp accuracy impairments because its path data delay determination was already clearly defined by the original specification, IEEE Std 802.3-2018, and not subject to implementation flexibilities.

90A.3 Considerations for Use of Different Message Timestamp Points

If two TimeSync Clients use different message timestamp points, a timestamp accuracy impairment equal to the time difference between the two message timestamp points will be incurred on the TimeSync link delay measurement. The magnitude of this impairment, as shown in Table 90A-1, is the time difference between the beginning of the SFD message timestamp point and the beginning of the first symbol after the SFD message timestamp point when they are adjacent to each other, which they normally are. For implementations that do not use the TX/RX num_unit_change support and Multilane support registers (see Table 45-235), an additional impairment could result if these two message timestamp point options are further separated at due to:

- Insertion of bytes between the two message timestamp points for AM or CWM functions

- Multi-PCS lane distribution delays

Implementations compliant to this version of the standard only suffer a timestamp accuracy impairment of one byte time between the two message timestamp point options because:

- The effect of AM or CWM insertion is accounted for, using the Tx_num_unit_change and Rx_num_unit_change primitives (see 90.4.3.3, 90.4.3.4, and Annex 90A.5)
- The multi-PCS lane path data delay is modelled as a constant value for all PCS lanes (see 90.7 and Annex 90A.4).

90A.4 Considerations for Multi-PCS Lane Functions

The general concept used to accommodate the delay variation of the multi-PCS lane distribution/merging operation is explained in Annex 90A.7. This concept takes advantage of the fact that the sum of the intrinsic delay variation of the transmit (Tx) multi-PCS lane distribution operation and of the intrinsic delay variation of the receive (Rx) multi-PCS lane merging operation is a predetermined constant for the given multi-PCS lane function.

This concept allows the intrinsic delay variations to be modelled as constant values, thus enabling the static TimeSync PCS transmit path data delay register and TimeSync PCS receive path data delay register to be used with high accuracy timestamping even when multi-PCS lane functions are present. As explained in 90.7, the TimeSync PCS transmit path data delay register would use the greatest PCS lane distribution delay as its constant value (which corresponds to the start of the Tx PCS lane distribution function) and the TimeSync PCS receive path data delay register would use the smallest PCS lane merging delay as its constant value (which corresponds to the start of the Rx PCS lane merging function).

Because the PCS transmit path data delay is modelled as a constant value, the minimum and maximum TimeSync PCS transmit path data delay registers in an ideal implementation have the same value due to the multi-PCS lane distribution operation. Likewise, because the PCS receive path data delay is modelled as a constant value, the minimum and maximum TimeSync PCS receive path data delay registers in an ideal implementation have the same value due to the multi-PCS lane merging operation. Having identical minimum and maximum values in these registers indicates that there is no uncertainty in the PCS path data delay.

The above consideration of the multi-PCS lane distribution/merging operation is consistent with that for the multi-FEC lane distribution/merging operation.

90A.5 Considerations for AM/CWM and Idle Functions

Timestamp accuracy impairment can occur because AM, CWM, or Idle insertion and removal events cause an instant change in the PCS path data delay. Unlike other PHY functions, these events do not generate PCS path data delay variations that can be pre-determined and the Tx path data delay variation is not mirrored by the Rx path data delay variation.

Each of these path data delay variations may be accounted for by using the Tx_num_unit_change and Rx_num_unit_change primitives (see 90.4.3.3 and 90.4.3.4). These primitives allow the TimeSync Client to compensate for the instant change in the path data delay. Because the primitives compensate for the instant path data delay changes, the TimeSync PCS transmit path data delay register and TimeSync PCS

receive path data delay register can operate as static values, even when AM, CWM, or Idle insertion/removal operations are present.

Examples that show how Tx_num_unit_change and Rx_num_unit_change may be used are given in Annex 90A.5.1 and Annex 90A.5.2, respectively.

[Note: do we need to add figures to Annex 90A.5.1 and Annex 90A.5.2 to help illustrate the examples?]

90A.5.1 Example use of Tx_num_unit_change

1. Scenario without AM, CWM, or Idle insertion/removal event:
    - Arrival time of a message timestamp point at the Tx xMII = T1
    - Tx PCS path data delay = PDD1
        - The constant value, PDD1, is programmed into the TimeSync PCS transmit path data delay registers
    - Calculated Tx departure timestamp = T1 + PDD1
2. Scenario with AM, CWM, or Idle insertion/removal in which Tx_num_unit_change is used to account for the Tx PCS path data delay variation, allowing the Tx PCS path data delay to be modelled as a constant:
    - Adjusted arrival time of the message timestamp point at the Tx xMII = T1 + Tx_num_unit_change*(nanoseconds/unit)
        - The arrival time at the Tx xMII is modified to reflect the AM, CWM, or Idle insertion/removal event (as if it happened before the Tx xMII, per 90.7)
        - The value of Tx_num_unit_change is positive when data is inserted ahead of the message timestamp point, increasing the Tx path data delay, and negative when data is removed ahead of the message timestamp point, decreasing the Tx path data delay
    - Tx PCS path data delay = PDD1
        - The constant value, PDD1, programmed into the TimeSync PCS transmit path data delay registers does not change.
    - Calculated Tx departure timestamp = T1 + (PDD1 + Tx_num_unit_change*(nanoseconds/unit))

90A.5.2 Example use of Rx_num_unit_change

1. Scenario without AM, CWM, or Idle insertion/removal event:
    - Arrival time of a message timestamp point at the Rx xMII = T2
    - Rx PCS path data delay = PDD2
        - The constant value, PDD2, is programmed into the TimeSync PCS receive path data delay registers
    - Calculated Rx arrival timestamp = T2 – PDD2

2. Scenario with AM, CWM, or Idle insertion/removal in which Rx_num_unit_change is used to account for the Rx PCS path data delay variation, allowing the Rx PCS path data delay to be modelled as a constant:

    - Adjusted arrival time of the message timestamp point at the Rx xMII = T2 + Rx_num_unit_change*(nanoseconds/unit)
        - The arrival time at the Rx xMII is modified to reflect the AM, CWM, or Idle insertion/removal event (as if it happened after the Rx xMII, per 90.7)
        - The value of Rx_num_unit_change is positive when data is inserted ahead of the message timestamp point, increasing the Rx path data delay, and negative when data is removed ahead of the message timestamp point, decreasing the Rx path data delay
    - Rx PCS path data delay = PDD2
        - The constant value, PDD2, programmed into the TimeSync PCS receive path data delay registers does not change.
    - Calculated Rx arrival timestamp = T2 – (PDD2 + Rx_num_unit_change*(nanoseconds/unit))

90A.5.3 Considerations for Implementations without Tx num_unit_change and Rx_num_unit_change

For an implementation that does not compensate for the path data delay variation resulting from AM, CWM, or Idle insertion/deletion removal events (e.g., without the Tx_num_unit_change and Rx_num_unit_change primitives), the effect of the timestamp accuracy impairments that result from these events can be evaluated to determine if they cause significant degradation in the TimeSync system's performance. Some observations that might help this evaluation are given below:

- Typically, the probability that an AM, CWM, or Idle insertion/deletion removal event affects the path data delay of a TimeSync message is small.
- A low-pass filter, which might be present in a TimeSync Client's time recovery algorithm, could attenuate the effect of the resulting impairments.
- An implementation that does not transmit TimeSync messages in the region of AM/CWM insertions or Idle insertions/removals might avoid the generation of the impairment at its Tx port. However, this does not guarantee that the corresponding remote Rx port will not generate impairments from its own Idle insertions/removals.

90A.6 Considerations for Tx Skew

For a multi-lane PHY, the receiver accounts for the skew of the medium by timestamping with respect to the lane with the smallest deskew buffer delay (see 90.7). Thus, the medium delay used by the time synchronization protocol is that of the lane with the greatest delay.

For these multi-lane PHYs, the presence of skew at the transmit Medium Dependent Interface (MDI) is difficult to compensate for because this skew is entwined with but independent from the skew of the medium. As shown in the examples of Figure 90A-1, the transmit skew in series with the medium skew can either be additive or subtractive. Example 1 and Example 2 in Figure 90A-1 have the same transmit skew and the same medium skew but, because these skews are associated differently, the total skew

seen at the receiver is different. By obscuring the latency of the medium, transmit skew can contribute to time synchronization error.

The per-lane transmit skew values might be compensated at the receiver, where the total skew of each lane can be observed at its Rx deskew buffers. By using the observed per-lane total skew values at the receiver and the per-lane transmit skew values, the actual skew of each lane of medium could be determined. To negate the need for this type of processing, it is recommended that multi-lane transmitter implementations try to minimize the lane skew at their MDI.

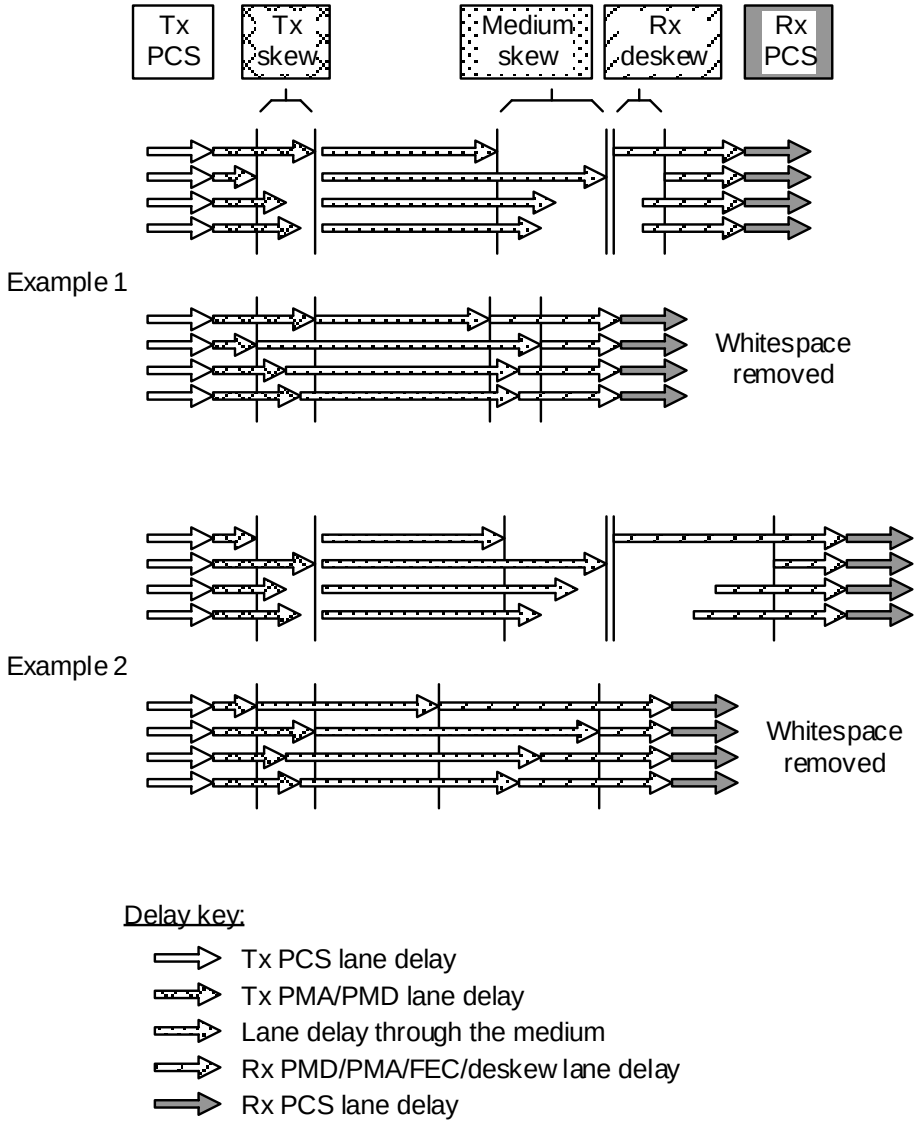

Figure 90A-1 – Transmit PMA/PMD Skew in Series with Medium Skew

90A.7 General Method for Dealing with Repeating Delay Variation Patterns

Many PHY functions have varying intrinsic delays with the following characteristics:

- The Tx and the Rx intrinsic delay variations follow a known repeating pattern.

- The Tx intrinsic delay variation pattern is a mirror of the Rx intrinsic delay variation pattern and the sum of the two intrinsic delays is a known constant value. This is true because the data stream before the Tx function and the Rx stream after the Rx function are identical.

It is possible to take advantage of the above characteristics to simplify the path data delay modeling. For example, if a PHY has multiple functions with these delay characteristics, as shown in Figure 90A-2, its aggregated path data delay may be modelled as a constant value instead of the dynamically varying sum of multiple varying delays.

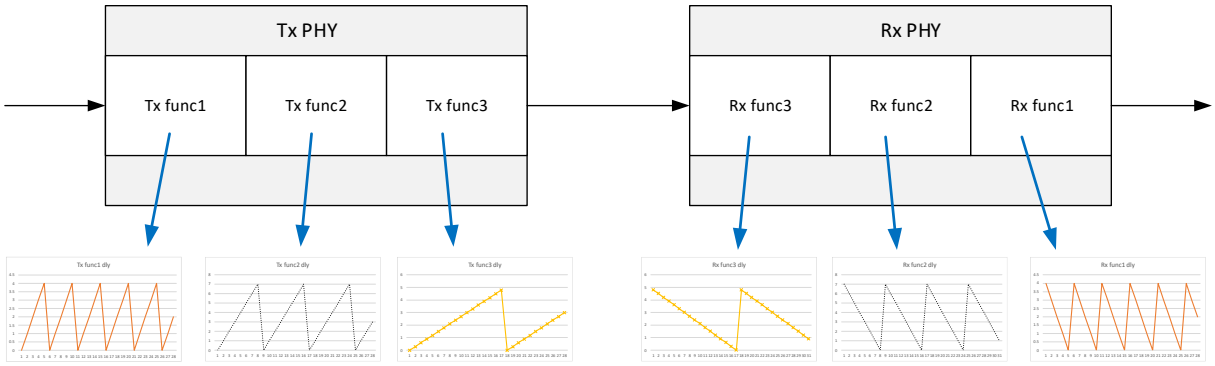

Figure 90A-2 – PHY with Cascaded Functions with Varying Delays

For the example shown in Figure 90A-2, the delay of each Tx function and the sum of them are shown in Figure 90A-3 and the delay of each Rx function and the sum of them are shown in Figure 90A-4. These sums have no easily discernable pattern and might require an implementation to determine the instantaneous path data delay for any chosen bit that corresponds to the message timestamp point of a TimeSync message in the Ethernet data steam.

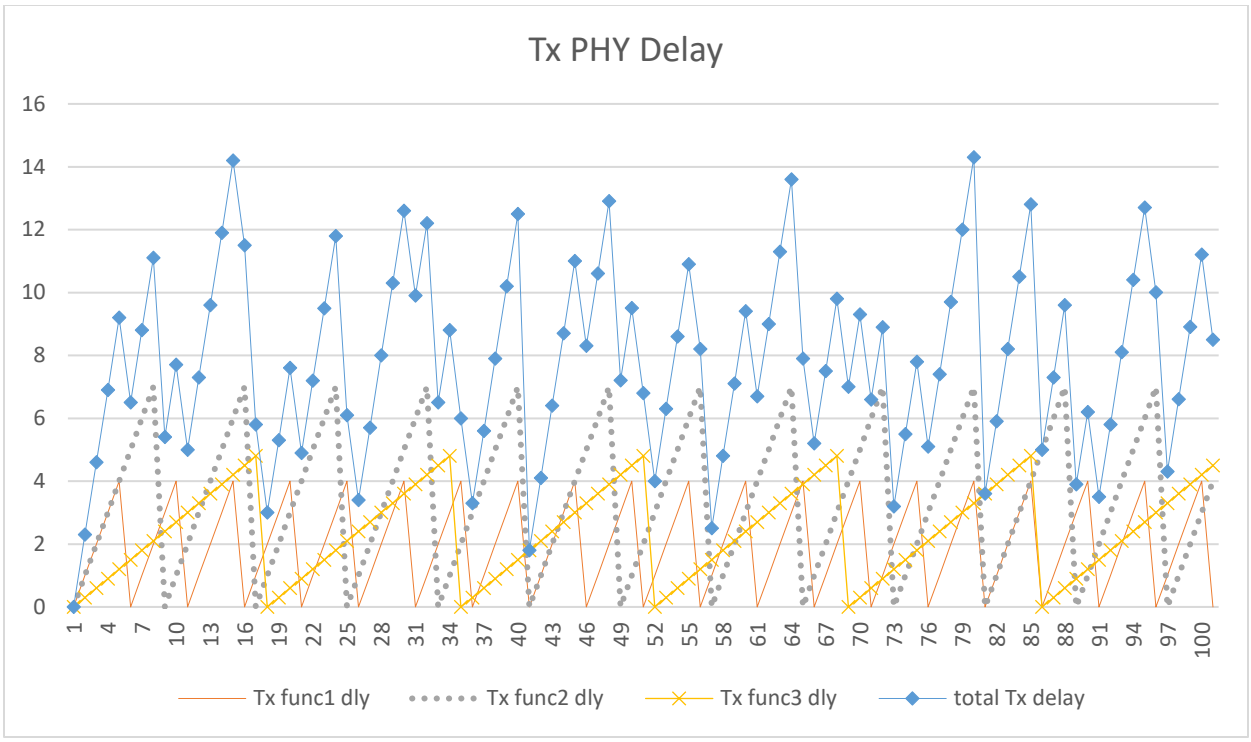

Figure 90A-3 –Total Delay of Tx PHY with Cascaded Varying Delays

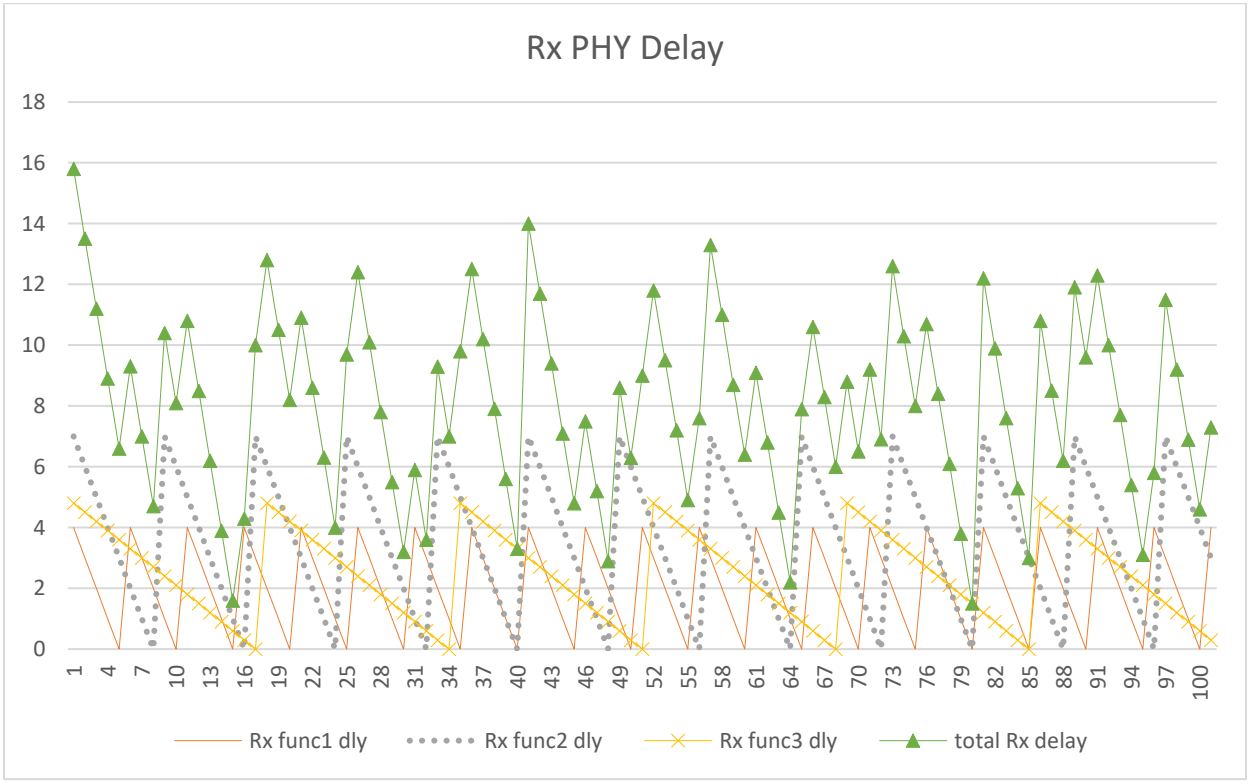

Figure 90A-4 – Total Delay of Rx PHY with Cascaded Varying Delays

Because the intrinsic varying delay in the Rx PHY is a mirror of the intrinsic varying delay in the Tx PHY, the total intrinsic delay through both PHYs is a constant, as illustrated in Figure 90A-5. This eliminates the need to track the varying delay of the message timestamp point of a TimeSync message through the Tx PHY and the Rx PHY. Instead, it is possible to divide the aggregate constant delay into Tx and Rx portions and allocate them to the individual PHY instances. The allocated portions of the constant total intrinsic delay value are then added to the implementation-specific delays, which are also constant values, of the corresponding Tx and Rx PHYs to compensate these intrinsic varying delays into timestamps.

It is recommended to use this method to model all varying PHY delays of this nature.

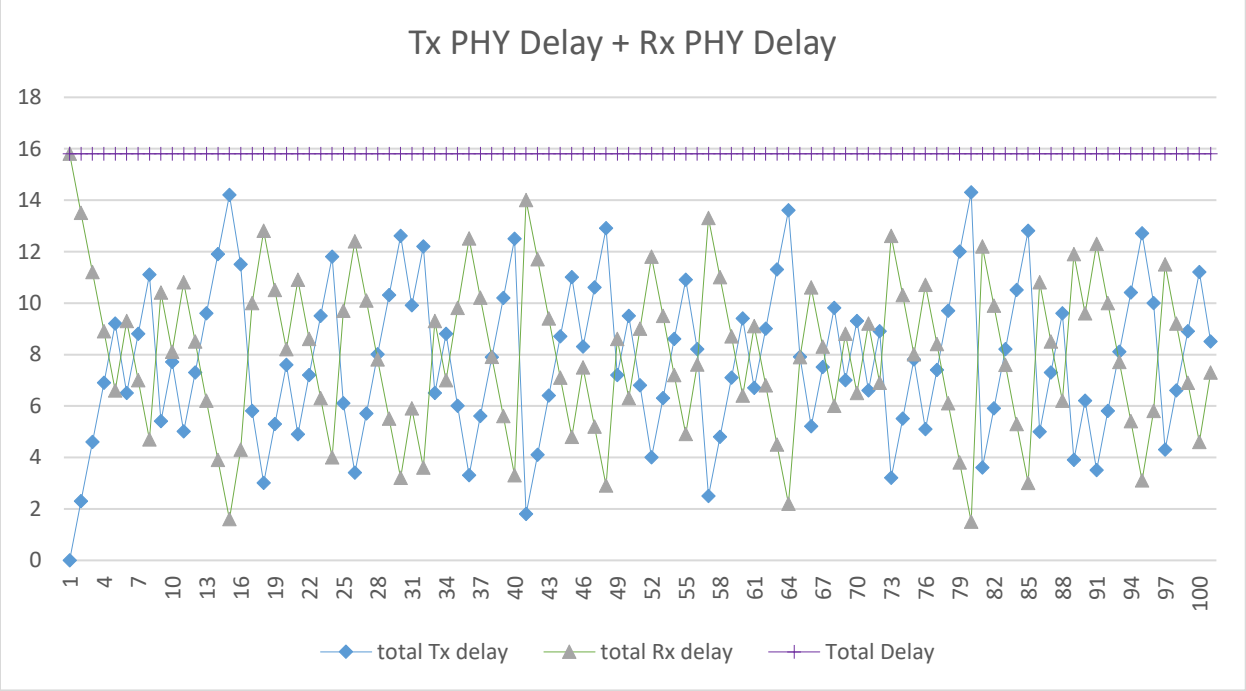

Figure 90A-5 – Tx PHY Delay, Rx PHY Delay, and Total Delay

[Note: Should we give examples of how this method can be used for existing basic functions such as 64B/66B encoding/decoding, 2x32B to 66B encoding/decoding, 256B/257B transcoding?]