WHITE **PAPER**

cādence®

PCIe 6.0 Electrical Testing for High Data-Bandwidth Applications

Anoop Veliyath, Sr. Design Engineering Manager, and Vinod Kumar Khera, PhD

For nearly three decades, PCI Express® (PCIe®) technology has been the standard interconnect inside computers providing high bandwidth and low latency to meet customer demand. However, as the industry needs to evolve, so does the standard, keeping pace and driving future innovation.

PCIe 6.0 is ubiquitous and offers power-efficient performance and high bandwidth for latency-sensitive applications from handheld mobile phones to supercomputers. It is backward compatible with all prior generations to ensure seamless operation, interoperability, reusability, and easy transition.

Contents

What Is PCIe 6.0?

Technological advancements are leading to innovations such as high-performance computing (HPC), autonomous driving, AI/ML, and data centers. These data-intensive applications create a deluge of data and demand high bandwidth, high-speed interconnects, and the least latency for quick data access. Figure 1 shows that data bandwidth doubles every three years to support next-generation innovations.

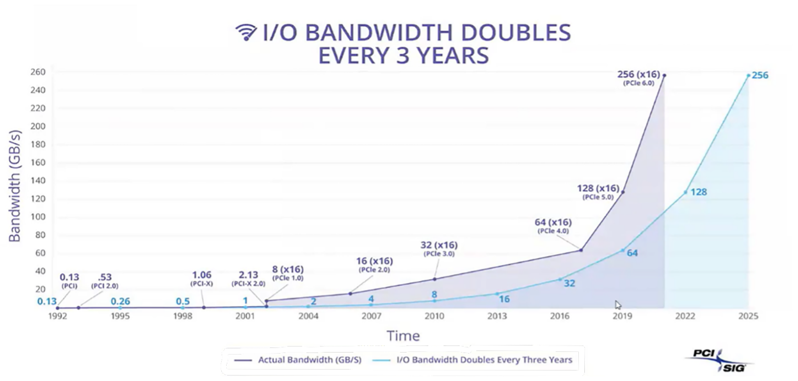


Figure 1: PCIe evolution[1]

To stay ahead of the data demand curve and serve the entire compute continuum, the Peripheral Component Interconnect Special Interest Group (PCI-SIG) introduced PCIe 6.0 with a double data rate of 64GT/sec. To keep the insertion loss under check, PCIe 6.0 uses a new physical layer and PAM4 (Pulse Amplitude Modulation with four signaling levels). Operating at a data rate of 64Gbps with a Nyquist frequency of 32GHz (NRZ mode) generates a differential insertion loss of -58db, as Figure 2 shows. It is impossible to equalize and yield a satisfactory bit error rate (BER) performance with such huge losses.

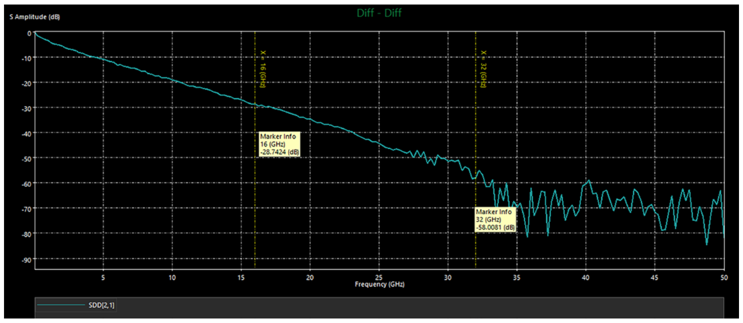


Figure 2: High insertion losses at high Nyquist frequency

PAM4 helps control insertion losses by utilizing the same Nyquist frequency as in PCIe 5.0. The major benefits of the PCI 6.0 technology are as follows:

- Increased bandwidth with low latency
- Scalability
- Backward compatibility

As PCI technology is scalable to hundreds of lanes in the platform, all at low cost, it is a natural fit for data-intensive markets such as data centers, artificial intelligence/automotive internet of things (IoT), and military applications.

PAM4: Modulation Scheme for PCIe 6.0

PAM4 is a multilevel modulation scheme with four different signal levels (and allows 00, 01, 10, and 11 as) per unit interval (UI) without increasing the transmission frequency. It ensures that channel loss is consistent with PCIe 5.0 (32GT/s), as the baud rate is 32Gb/s for both with a Nyquist frequency of 16GHz. However, PAM4 signaling is more susceptible to errors. To mitigate such issues and ensure low latency and high efficiency, PCIe 6.0 incorporates a forward error correction (FEC) scheme and a cyclic redundancy check (CRC) and retry mechanism.

Error detection and correction: FEC and CRC

The PCIe 6.0 specification features two primary mechanisms to correct errors: forward error correction (FEC) and cyclic redundancy check (CRC). FEC uses fixed-size packets FLITs to avoid any framing, and inefficient interconnects. Each FLIT used in FEC is 256 bytes and is partitioned into 242 bytes of payload protected by 8 bytes of CRC, as Figure 3 shows.

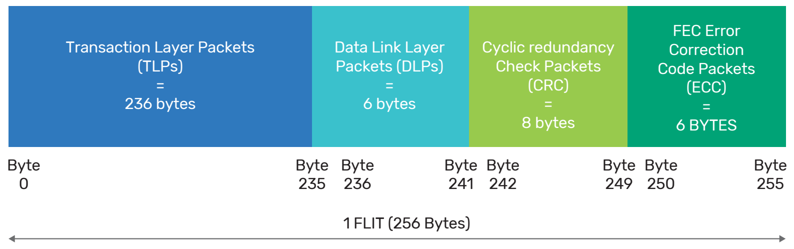

Transaction Layer Packets (TLPs) = 236 bytes
Data Link Layer Packets (DLPs) = 6 bytes
Cyclic redundancy Check Packets (CRC) = 8 bytes
FEC Error Correction Code Packets (ECC) = 6 BYTES
Byte 0
Byte 235
Byte 236
Byte 241
Byte 242
Byte 249
Byte 250
Byte 255
1 FLIT (256 Bytes)

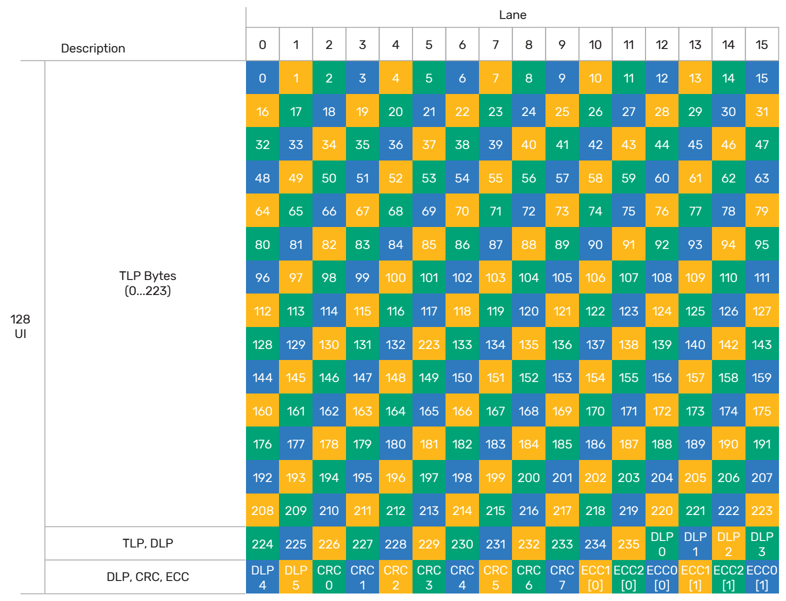
Figure 3: FLIT partitions

The 250 bytes of payload and CRC are protected by 6 bytes of FEC. Error correction code (ECC) and CRC are critical features while operating in the FLIT mode for detection and correcting bit errors in the data stream. The FEC involves adding some extra redundant bits into the data to facilitate detecting and correcting a limited number of bit errors. Here, CRC also involves adding some extra redundant bits into the data but can only detect errors and not correct them. If errors remain even after the maximum permissible attempts of FLIT replays (up to six times in 6.0), the PCIe Link moves to the "RECOVERY" state in the PCIe "Link Training and Status State Machine" (LTSSM).

Three-Way Interleaving Scheme for FEC

PCIe BASE 6 spec uses a three-way interleaved FEC, as Figure 4 shows. A 256-byte wide FLIT is split into three ECC groups, ECC0, ECC1, and ECC2, shaded in blue, orange, and green, respectively. Every byte of FLIT is allocated in these three ECC groups based on the interleaving scheme (see the x16 Link configuration below). One of these three groups is 86 bytes wide, and the rest are 85 bytes wide. Each of these three ECC groups is protected by a dedicated set of two ECC code bytes (thus, we have six ECC code bytes in all for an entire FLIT) present within these groups.

- ECC0[0] and ECC0[1] => To protect the bytes in ECC group 0
- ECC1[0] and ECC0[1] => To protect the bytes in ECC group 1
- ECC2[0] and ECC0[1] => To protect the bytes in ECC group 2



		Lane															
	Description	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
128 UI	TLP Bytes (0...223)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
		32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
		48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
		64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
		80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
		96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
		112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
		128	129	130	131	132	223	133	134	135	136	137	138	139	140	142	143
		144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
		160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
		176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
		192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
		208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
	TLP, DLP	224	225	226	227	228	229	230	231	232	233	234	235	DLP 0	DLP 1	DLP 2	DLP 3
	DLP, CRC, ECC	DLP 4	DLP 5	CRC 0	CRC 1	CRC 2	CRC 3	CRC 4	CRC 5	CRC 6	CRC 7	ECC1 [0]	ECC2 [0]	ECC0 [0]	ECC1 [1]	ECC2 [1]	ECC0 [1]

Figure 4: FEC interleaving scheme[2]

PCIe BASE 6 spec defines the FEC architecture as a "lightweight" FEC. It only corrects errors in a single byte in the ECC group to reduce latency. The three-way interleaving scheme performed on FLIT bytes for FEC restricts the bit errors inside the bytes allocated into each ECC group to, at most, one byte, even in case of continuous burst errors.

How Does the FEC Scheme Defined in the BASE 6 Spec Correct Errors Inside the FLIT Bytes?

To demonstrate the effectiveness of the FEC scheme, let's consider ECC group 0 with 86 bytes. We have the first 84 bytes consisting of TLP/DLLP/CRC bytes and the last two bytes consisting of the check byte and parity byte, as Figure 5 shows. We consider the calculations on the transmitter side and receiver side to explain the error correction scheme.

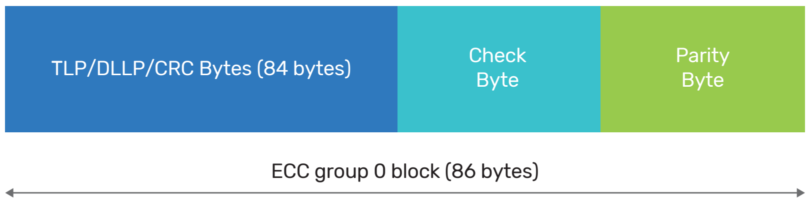


Figure 5: ECC group

Transmitter side: Here, the check byte and the parity byte, computed based on the Byte0 to Byte83 payload, are appended as Byte84 (check byte) and Byte85 (parity byte), respectively, to generate the whole 86-byte ECC group block. Figure 6 shows the basic FEC scheme; here, the parity byte is calculated as

Parity Byte = Byte0 ˆ Byte1 ˆ Byte2.... Byte82 ˆ Byte83
where "ˆ" denotes the bitwise XOR operation

Check Byte is also mathematically computed at the transmitter side encoder from Byte0 to Byte83, but the computation is more involved than the parity bit computation.

Only ECC group 0 has a payload (excluding the check and parity bytes), which is 84 bytes long. For ECC groups 1 and 2, the payload is only 83 bytes long. In this case, an extra byte (comprising of all 0 bits) is appended to generate the 84-byte payload, on to which the check and parity bytes are computed and appended precisely as in the ECC group 0 case to derive the 86-byte ECC group block even for ECC groups 1 and 2.

Receiver Side: On the receiver side, the 256-byte FLITs are again split into three-way bytes belonging to the three ECC groups, and the three ECC group blocks are fed into the respective decoders corresponding to each ECC group. So, considering the ECC group 0 decoder, it receives the 86-byte ECC block (bytes 84 and 85 being the check and parity bytes, respectively). This received block computes the parity byte and check byte from Byte0 to Byte83 in the same way it was done at the transmitter side encoder before transmitting.

To illustrate the FEC process, let's denote the received parity and check bytes at the receiver (i.e., Bytes 85 and 84, respectively, in the, received 86 byte-wide ECC group block) as Parity_Byte_Received and Check_Byte_Received, and those computed at the receive side from the actual 84 bytes received (Byte0 to Byte83) as Parity_Byte_Computed and Check_Byte_Computed.

We define two new bytes, Syndrome Parity and Syndrome Check, as follows:

- Syndrome Parity = Parity_Byte_Received ˆ Parity_Byte_Computed, where "ˆ" denotes the bitwise XOR operation
- Syndrome Check = Check_Byte_Received ˆ Check_Byte_Computed, where "ˆ" denotes the bitwise XOR operation

The syndrome parity byte indicates the bit positions where we have mismatches between the received parity byte (i.e., received Byte 85) versus the expected parity byte computed from the received Byte0 to Byte83 bytes. Similarly, the syndrome check byte indicates the bit positions where we have mismatches between the received parity byte (i.e., received Byte 85) versus the expected parity byte computed from the received Byte0 to Byte83 bytes.

- If we did not have any bit errors in symbols, the syndrome parity and syndrome check bytes would be all 0s.
- If either or both the syndrome parity and syndrome check bytes are non-zeros, we have bit errors within the 86-byte block corresponding to that ECC group.

Where there is only a single byte error within the 86-byte block corresponding to the given ECC group, based on the syndrome parity and syndrome check byte value, it is mathematically possible to identify the exact byte in error and the exact bits within that byte in error, to correct those bit errors in the byte. Here, we won't delve into the specifics involved in this operation.

- Where more than one byte is in error, the ECC correction scheme can't correct all errors, and the onus will be on the CRC check + link layer retry mechanism to escape the errors.
- Figure 6 shows the overall mechanism mentioned above in a simplified manner.

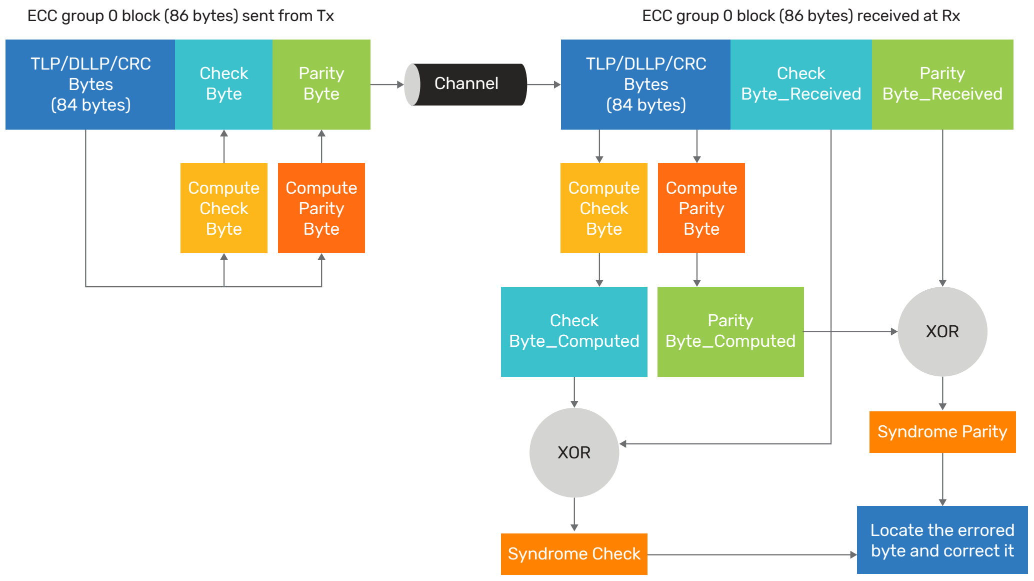


Figure 6: FEC scheme in PCIe 6.0

Gray Coding over PAM4

The adoption of PAM4 introduces three eye diagrams and reduces the vertical eye diagram height by 33%, as Figure 7 shows.

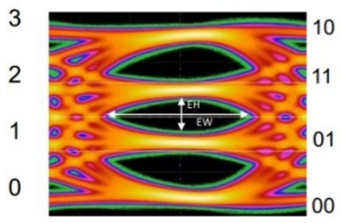


Figure 7: PAM4[3]

The reduced noise margins result in exacerbated cross-talk interferences, signal reflections, and power supply noise. PCIe 6.0 adopted Gray coding and (1/1+d) precoding scheme to minimize bit errors and reduce burst error propagation. Gray coding for PAM4 signal transmission in PCIe 6.0 ensures that symbol errors induced by voltage noise result in a single-bit error at most. For instance, Table 1 demonstrates how.

Coding Scheme Used	Original Symbol Transmitted	Erroneous Symbol Received	Number of bit Errors Between the Original Transmitted Symbol and the Erroneous Received Symbol
Binary	2'b10	2'b01	2'b10 vs. 2'b<u>01</u> = 2 bit errors
Gray Coding	2'b11	2'b01	2'b11 vs. 2'b<u>0</u>1 = 1 bit error

Table 1: Gray coding and single-bit error

Gray coding helps achieve a better BER for the same symbol error rate (SER) versus binary coding.

1/1+d Precoding

This precoding scheme helps to reduce the extent of error bursts due to decision feedback equalizer (DFE) error propagation. DFE affects the FEC efficacy in long error bursts. The DFE error events burst length for PAM4 can be reduced by precoding. Figure 8 shows the 1/1+d precoding used in PCIe 6.0.

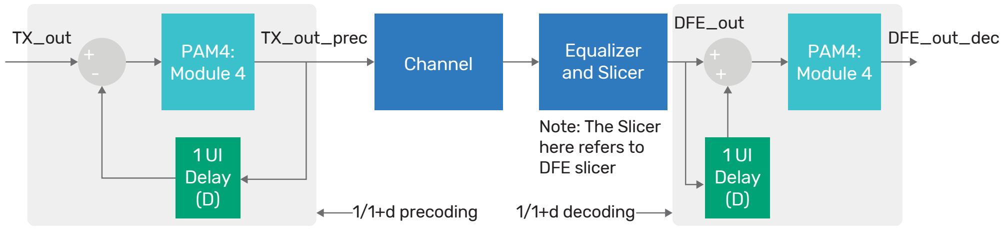


Figure 8: 1/1+d precoding scheme

Benefits of 1/1+d precoding scheme

To visualize how the 1/1+d precoding scheme helps reduce the risk of DFE error propagation-related error bursts, let us consider an ideal one-tap DFE circuit, as Figure 9 shows. Here, DFE constituent blocks, such as the summer, clocked comparator, etc., are implemented using idealized behavioral models of the same.

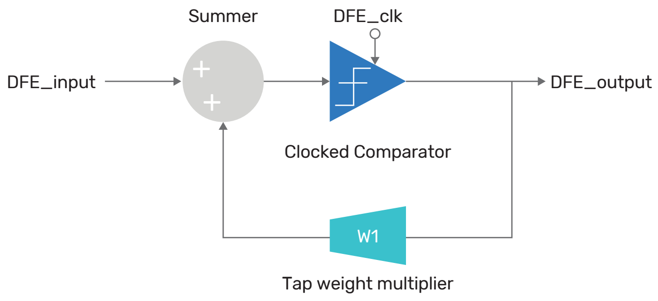


Figure 9: 1-tap DFE circuit

Let's consider a simplified and idealistic case, as Figure 10 shows, where an ideal single-tap DFE (as above) is used. Here, the DFE tap weight represented by "W1" is untrained, and fixed tap weights are applied.

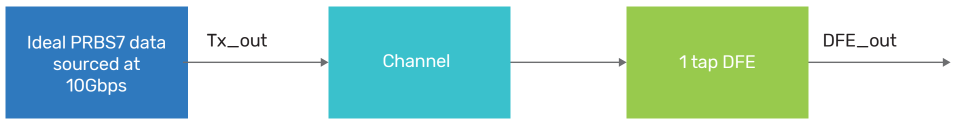
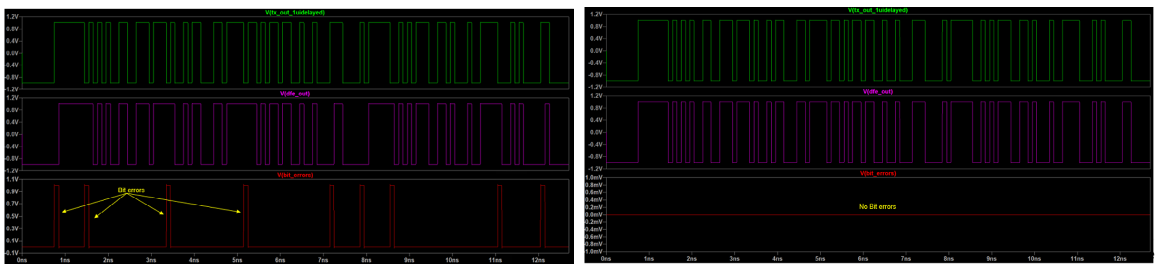


Figure 10: Simple ideal solution with one-tap DFE

Also, for simplifying the analysis, a simple RC low pass filter (R= 175 ohms, C = 1pF) is used to introduce some ISI. Figure 11 shows the waveforms (at different weights) using SPICE sims using idealized behavioral models.



W1 = 0V (No DFE correction) W1 = 0.15V

Figure 11: Response at different DFE weights

Here

- “Tx_out” is the original signal transmitted into the channel
- “DFE_out” is the output after the DFE slicer
- “Bit_Errors,” indicates bit errors resulting from the XOR operation of “Tx_out” and “DFE_out” after accounting for the DFE related 1 unit-interval (UI) delay on “DFE_out” versus “Tx_out” by adding a 1UI delay on “Tx_out” before performing the XOR operation (henceforth, this 1UI delayed version of “Tx_out” will be referred to as “Tx_out_1UIdelayed”)

As we can see, when the DFE tap weight (W1) is “0,” that is, in the absence of DFE tap corrections, we make a few errors because the ISI and the DFE are doing their jobs, offering sufficient ISI correction to eliminate any bit errors here when W1 value is increased to 0.15V. The heavier the DFE tap weight, the heavier the correction voltage applied at the summer by the DFE will be; hence, the more severe the consequence of a wrong DFE decision will be on future decisions.

The DFE slicer output is deliberately inverted for a single bit to artificially induce a false DFE decision to showcase the impact of DFE tap weight (W1) on the risk of setting off error bursts, as Figure 12 shows.

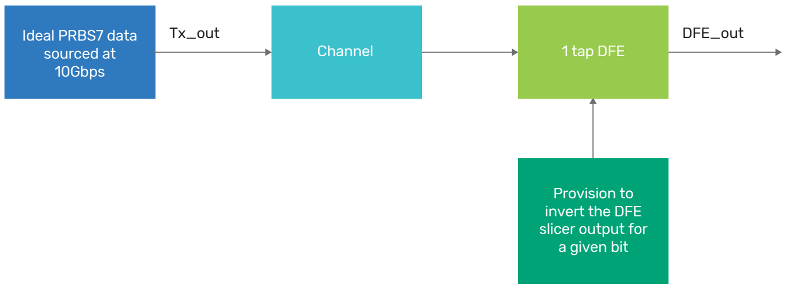
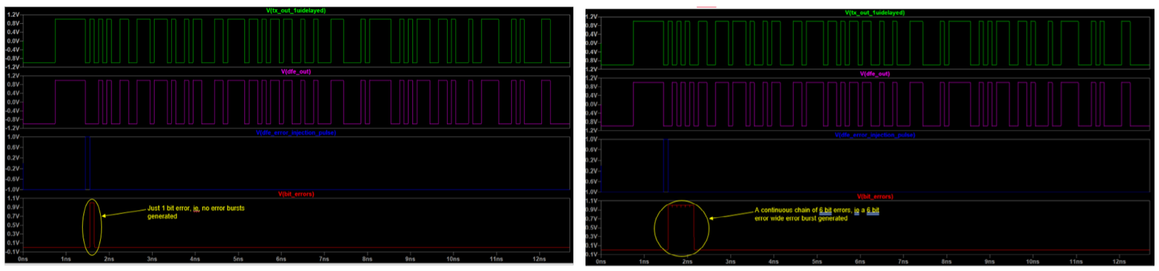

Ideal PRBS7 data sourced at 10Gbps
Tx_out
Channel
1 tap DFE
DFE_out
Provision to invert the DFE slicer output for a given bit

Figure 12: False DFE induction scheme

This bit position is strategically chosen to maximize the risk of setting off burst errors to showcase the continuous bursts, as Figure 13 shows.



W1=0.15 W1 = 0.36V

Figure 13: DFE tap weights and the error bursts

When DFE tap weight magnitude = 0.15V, there is just a 1-bit error overall, that is, the one corresponding to the inversion of the DFE decision. It means the artificially induced false DFE decision sets off no error bursts. However, the increased DFE tap weight magnitude of 0.36V results in a continuous burst of 6-bit errors. This results from the chain reaction of errors ignited by the inverted correction voltage applied by the DFE, which is high enough to corrupt the next DFE decision, which then corrupts the next one, and so on.

Further, Figure 14 shows a zoomed view of the signals around the region of the error burst. It is easy to visualize that for the block of bits, we get an error burst, and the result is the same as having an inverting function upon the original correct bits within this block.

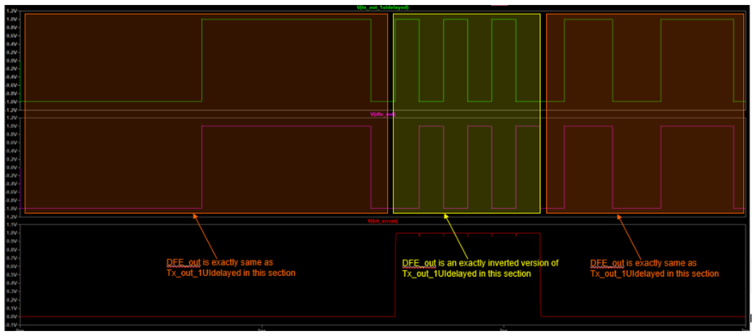


Figure 14: DFE_out is like theTX_out1 UI delayed

This property plays a significant role in the operation of 1/(1+D) precoding, as we will see next. Figure 15 shows a block diagram representing the simplified 1/(1+D) precoding scheme in NRZ mode.

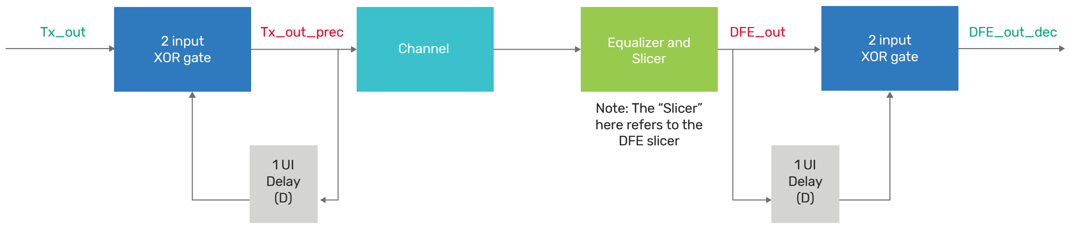
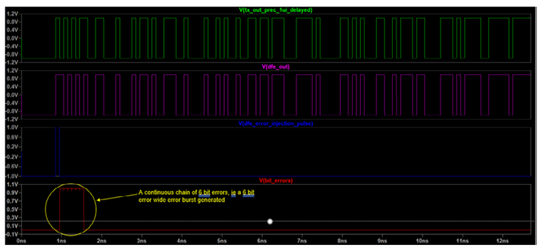
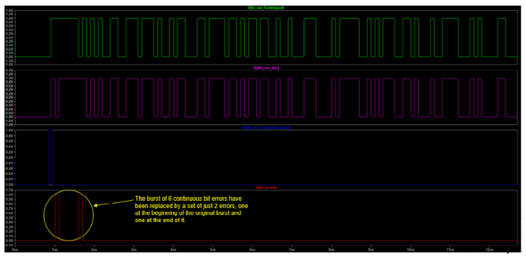


Figure 15: 1/1+D precoding scheme in NRZ mode

With a DFE tap weight magnitude of 0.36V and the DFE slicer decision inverted for a single bit to artificially induce a false DFE decision, even here, we see that we get a continuous burst of 6-bit errors as in Figure 16(a).



(a) Continuous burst of error

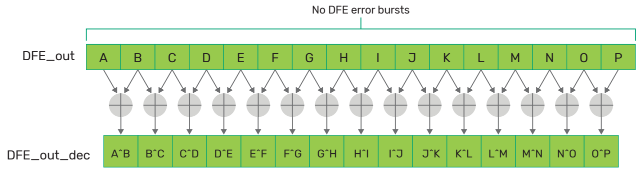


(b) Pair of errors only

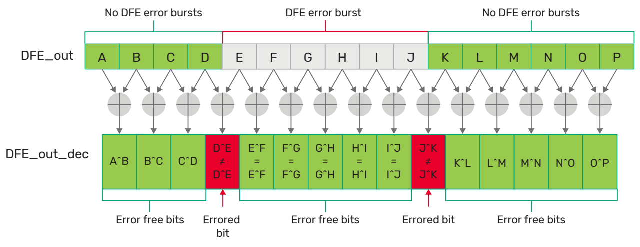
Figure 16: Impact of 1/1+d precoding scheme

Let us now examine the primary input (Tx_out) and final output (DFE_out_dec) signals, encircled in green, in Figure 16(b), where the precoding and decoding processes have been entirely performed. We now see that the burst of six continuous bit errors has been replaced by just two errors, one at the beginning of the original burst and one at the end. This holds for similar burst errors of any length.

Let us consider an example to show the impact of the 1/(1+D) decoding scheme in the context of NRZ. As in Figure 17(a), A, B, C, D, etc., represent the output of the DFE slicer for a few consecutive NRZ bits sent across the channel, that is, "DFE_out."


No DFE error bursts
DFE_out
A B C D E F G H I J K L M N O P
DFE_out_dec
A^B B^C C^D D^E E^F F^G G^H H^I I^J J^K K^L L^M M^N N^O O^P

(a) NO DFE error burst



(b) Error burst reduced to two errors

Figure 17: Precoding scheme and its impact

Now let us assume bits from E to J (both inclusive) were all flipped due to a DFE error burst that originated from bit E, a 6-bit wide error burst, as observed on “DFE_out.” However, on “DFE_out_dec,” (i.e., the final output after the 1/(1+D) decoding), we see 2-bit errors (as indicated by the red shaded cells in Figure 17(b)), one at the point where the original error burst on “DFE_out” begins, and the other where the error burst on “DFE_out” ends. This is because for any pair of bits X and Y, X ^ Y = $\bar{X}$ ^ $\bar{Y}$, for a two-input XOR operation, the outcome is exactly the same when either both inputs are inverted or when both are not inverted.

To extend these observations in the PAM4 context, let us revisit the PAM4 case. The critical difference is that in PAM4, we apply a Modulo-4 operation on Tx_out – z-1*Txout_prec rather than a Modulo-2 operation as in NRZ. In the context of PAM4 symbols, the DFE burst errors typically manifest in an alternating “+1,-1,+1,-1…” error pattern added on top of the original PAM4 symbols. Let us consider a case where the DFE made an error for some reason, resulting in a symbol X being interpreted as X+1. Then, depending on the peculiarity of the symbol pattern following X, if an error burst triggers due to the wrong decision on X, the next symbol, say Y, becomes Y-1, and that, in turn, leads to the symbol that follows, say Z being interpreted as Z+1, and so on. The following table shows an example.

Tx_out_prec	0	2	0	2	2	1	1	3	2	0
DFE_out	0	1	1	1	3	0	2	2	3	0
Symbol error	0	-1	+1	-1	+1	-1	+1	-1	+1	0

Figure 18 depicts one such case, with a DFE error propagation-related error burst (six PAM4 symbols wide) as observed on “DFE_out” and what we obtain after performing 1/1(1+D) decoding on it, as observed on “DFE_out_dec.” In Figure 18 (a), A, B, C, D, etc., represent the output of the DFE slicer for a few consecutive PAM4 symbols sent across the channel, that is, “DFE_out.”

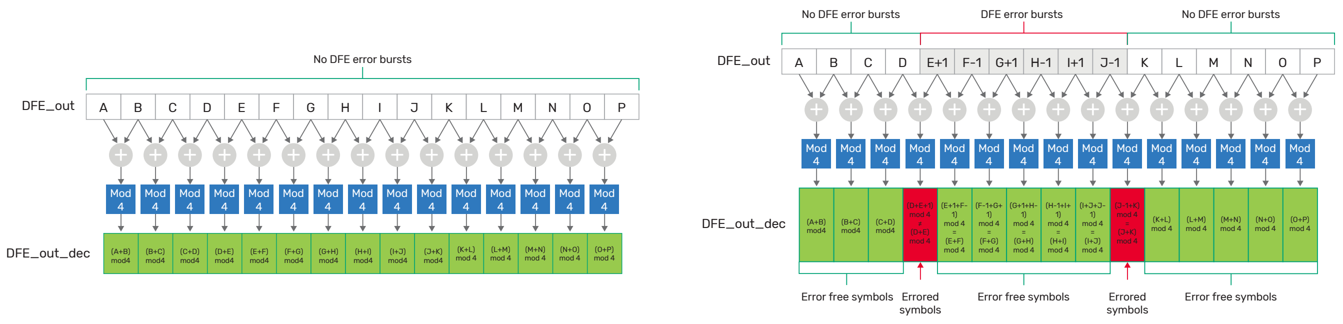


Figure 18: Continuous error burst to a pair of errors after precoding

It shows that any continuous DFE error propagation-related symbol burst error (of any length) gets converted to just a pair of symbol errors, one at the beginning of the original burst and the other at the end of the burst, in the case of PAM4 1/(1+D) precoding and decoding.

Low Power Consumption

PCIe 6.0 introduces a low-power state, L0p, in red in Figure 19, to run the traffic on fewer lanes and save power. L0p is a substate of the active link state L0 and is optional for link width resizing, and is used in FLIT mode only.

L0p enables power consumption proportionate to bandwidth usage without interrupting traffic flow. To ensure uninterrupted traffic flow, It maintains at least one active lane. L0p has the following features:

- L0p state provides a power savings mode wherein the link stays active. Still, power savings are obtained by shutting off some lanes (by maintaining them in electrical idle) in the link but ensuring that at least one lane stays active and exchanging traffic.
- It is supported for ALL PCIe rates, as long as the link is in FLIT mode of operation at the given PCIe rate.
- L0p mode support is optional for all kinds of ports except retimers. For retimers, L0p mode support is mandatory.
- L0p allows for scaling down and backing up the link widths while staying in L0 mode at all times, rather than going through the “configuration” LTSSM state, etc., for such link width changes.

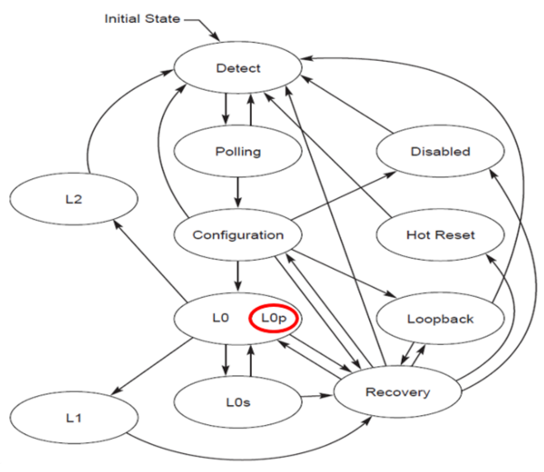


Figure 19: L0p state for low-power consumption

Training Sequence TS0

PCIe 6.0 introduces a training sequence (TS0) to communicate equalization information in specific symbols and is always 1b/1b encoded. It consists of 16 symbols of 1 byte each and is utilized only during the link equalization process of the link training and status state machine (LTSSM) in PCIe 6.0. It contains information associated with the link equalization process, such as the transmitter preset value or the C-2, C-1, C0, and C+1 transmitter coefficient values requested from the far-end partner. One important property that makes it different from TS1/TS2 in the context of PCIe 6.0 PAM4 operation is that in TS0, for each bit pair that forms a single PAM4 symbol, the bit in the even bit position is set to be identical to the bit in the odd bit position. This implies that 2'b00 or 2'b11 are only the PAM4 symbols possible inside a TS0; therefore, the TS0 behaves like an NRZ waveform rather than a PAM4 waveform, as only two levels are available.

This NRZ nature of the TS0 sequence enables the receiver to obtain the initial CDR lock and makes it easier to resolve the bits more accurately when the link has just entered 6.0 mode, but the link equalization process has yet to happen. Here, the scrambling process is called "half scrambling," as both bits are identical.

Measurements in PCIe 6.0

What's new in Tx jitter parameter measurements for PCIe 6.0?

A lower value of jitter and the increased slew rate of 0/1 transitions in PAM4 demands an advanced algorithm to measure the jitter in PAM4. The following parameters are defined to be measured with a "48 edge" jitter measurement for PCIe 6.0, unlike PCIe 5.0.

- Tx uncorrelated total jitter
- Tx uncorrelated total jitter when testing for the IR clock mode with SSC
- Tx uncorrelated deterministic jitter

PCIe 6.0 defines random jitter, uncorrelated deterministic pulse width jitter, and total pulse width jitter with a clock-like pattern. The BASE 6 spec recommends the jitter measurements in PCIe 6.0 to be made with a fourth-order Bessel-Thomson filter with a -3dB cut-off frequency at 33GHz applied on the oscilloscope to minimize the impact of high-frequency oscilloscope noise on the measurements. This will be applied over and above a minimum oscilloscope bandwidth of 50GHz. Next, let us examine the "48 edges" jitter measurement methodology in the context of PCIe 6.0 Tx testing.

48 edge jitter measurement methodology

As we have four unique PAM4 symbols (0, 1, 2, and 3), we have 12 unique transitions possible (i.e., transitions from each PAM4 symbol to the three other PAM4 symbols), as follows:

Transition No	Transition
1	0→1
2	0→2
3	0→3
4	1→0
5	1→2
6	1→3
7	2→0
8	2→1
9	2→3
10	3→0
11	3→1
12	3→2

Table 2: Symbols transition table

The PCIe 6.0 spec defines a 12UI base pattern that includes all 12 of these transitions when the base pattern is repeated, as follows: { 0, 1, 0, 3, 0, 2, 1, 2, 3, 1, 3, 2 }. This pattern is DC-balanced and optimized for using the minimum number of symbols to involve all 12 transitions. However, its limitation is that it's not a prime number, which may result in not all 12 transitions being exercised for all combinations of the different transmit blocks. The PCIe 6.0 spec slightly modifies the 12UI base pattern block to make four segments. It avoids the reverse case scenario during actual operation during jitter testing. Each consists of a 13UI pattern, resulting in a 4*13 = 52UI long pattern. This approach maintains overall DC balance across the 52UI pattern and minimizes DC imbalances between the 13UI wide segments.

As per the PCIe 6.0 spec, for the three jitter parameters Tx uncorrelated total jitter, Tx uncorrelated total jitter when testing for the IR clock mode with SSC and Tx uncorrelated deterministic jitter, we are supposed to measure the jitter on each of these 48 edges separately and then average these to arrive at the final respective jitter parameter results.

SNDR measurements

SNDR is the variation between the ideal and measured signals for a specified number of measurements. The signal-to-noise-and-distortion ratio (SNDR) measurements of the transmitter measure the voltage noise and waveform shape distortion present in the transmitter waveform. As per the PCIe BASE 6 spec, the expression for Tx SNDR is as follows:

$$SNDR = 10 \times \log_{10}\left(\frac{p_{max}^2}{\sigma_e^2 + \sigma_n^2}\right) \tag{1}$$

Here, pmax is the maximum value of p(k), and σe is the standard deviation of e(k). σn is an average of four measurements of RMS deviation of the PAM4 voltage levels. It uses linear fit pulse response p(k) and linear fit error e(k) (the difference between the actual transmitter output signal and the ideal signal). To find the SNDR, we need to understand the "linear fit pulse response" waveform as the terms pmax and σe are derived from it. The following are the steps involved in the extraction of the linear fit pulse response waveform (also see Figure 20):

1. Source the compliance pattern defined in the spec sheet from the transmitter and capture the waveform on the oscilloscope for at least 250 repetitions of this pattern. While driving this waveform, no equalization must be applied from the transmitter, and a minimum oscilloscope bandwidth of 50GHz is required for the capture. Additionally, to limit the impact of the inherent oscilloscope noise on these measurements, the measurements are to be made with a fourth-order Bessel-Thomson filter with a -3dB cutoff frequency at 33GHz.

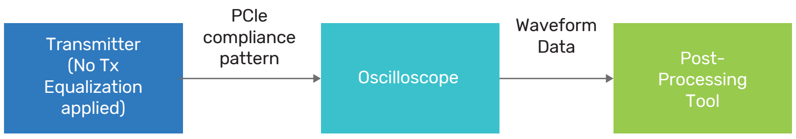


Figure 20: Linear fit pulse response extraction

The PCIe BASE 6 spec requires the oscilloscope to capture at least 32 samples per UI. At high speeds, such as 64Gbps, etc., the oscilloscope sampling rate required to achieve these many samples per UI may be prohibitive, so the spec allows interpolation methods, such as Sinc x Interpolation, etc., to achieve such sampling granularity.

2. This waveform is passed on to the post-processing tool for performing the next steps for the linear fit pulse response computation.

As mentioned before, The PCIe BASE 6 spec requires the oscilloscope to capture at least 32 samples per UI. Let this UI oversampling ratio (i.e., number of samples captured per UI) be M. Figure 21 illustrates this, where a zoomed portion of a 1010 clock-like section of the waveform for an NRZ pattern for simplicity (the same idea can be extended to PAM4 too). M = 4 is considered here for simplicity, although the spec demands M>=32 to adequately capture the artifacts of the waveform, including the transition regions, etc.

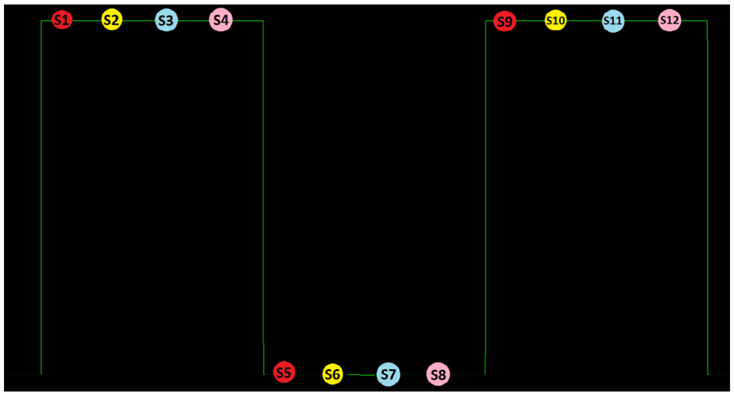


Figure 21: UI sampling

In Figure 21, the dot “S1” represents the first sample, the dot “S2” the second one, and so on. Therefore, we have M = 4 samples per UI. We denote every Mth sample with a dot of the same color. The core operation behind the linear fit pulse response computation is to perform a set of cross-correlation operations between specific groups of waveform samples and an ideal compliance pattern waveform (repeated the same number of times as was done for the Tx compliance pattern waveform capture). There are M separate cross-correlation operations involved, as follows (in our example, note that M =4):

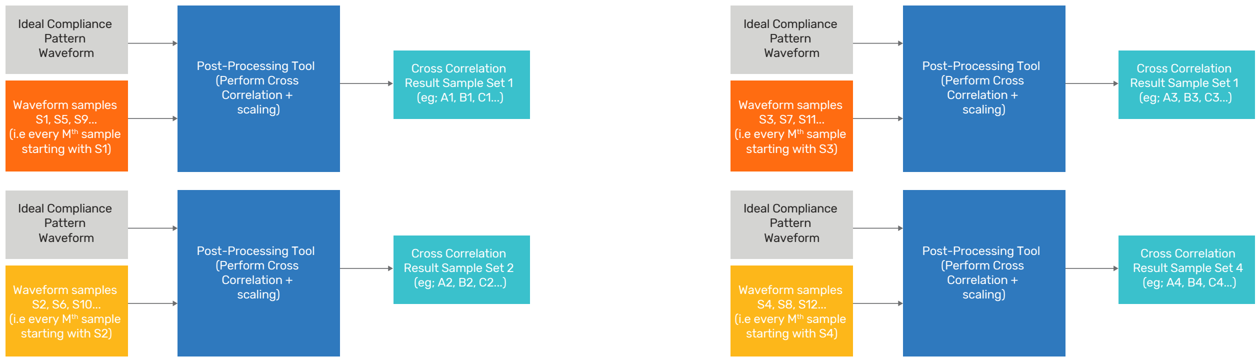

Ideal Compliance Pattern Waveform
Waveform samples S1, S5, S9... (i.e every Mth sample starting with S1)
Post-Processing Tool (Perform Cross Correlation + scaling)
Cross Correlation Result Sample Set 1 (eg: A1, B1, C1...)
Ideal Compliance Pattern Waveform
Waveform samples S3, S7, S11... (i.e every Mth sample starting with S3)
Post-Processing Tool (Perform Cross Correlation + scaling)
Cross Correlation Result Sample Set 1 (eg: A3, B3, C3...)
Ideal Compliance Pattern Waveform
Waveform samples S2, S6, S10... (i.e every Mth sample starting with S2)
Post-Processing Tool (Perform Cross Correlation + scaling)
Cross Correlation Result Sample Set 2 (eg: A2, B2, C2...)
Ideal Compliance Pattern Waveform
Waveform samples S4, S8, S12... (i.e every Mth sample starting with S4)
Post-Processing Tool (Perform Cross Correlation + scaling)
Cross Correlation Result Sample Set 4 (eg: A4, B4, C4...)

Figure 22: Cross-correlation

Figure 22 shows that we get M sets of cross-correlation result sample sets (4 sets here). A simple scaling operation, dividing the cross-correlation result points by the total number of UIs captured, is also required to scale the results appropriately for the next steps. The values in these cross-correlation result sample sets are then interleaved in the following fashion to obtain the linear fit pulse response waveform.

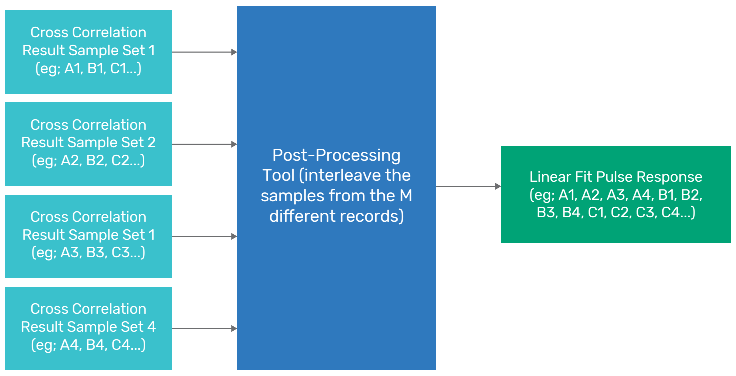

Cross Correlation Result Sample Set 1 (eg; A1, B1, C1...)
Cross Correlation Result Sample Set 2 (eg; A2, B2, C2...)
Cross Correlation Result Sample Set 1 (eg; A3, B3, C3...)
Cross Correlation Result Sample Set 4 (eg; A4, B4, C4...)
Post-Processing Tool (interleave the samples from the M different records)
Linear Fit Pulse Response (eg; A1, A2, A3, A4, B1, B2, B3, B4, C1, C2, C3, C4...)

Figure 23: Interleaving operation

Now that we've seen how the linear fit pulse response is computed, we can look more intuitively into what's happening here with these computations and how the SNDR is computed from the linear fit pulse response. A fundamental concept to be recalled in this context is that any ideal digital bit pattern can be constructed by linear superposition of scaled and time-shifted versions of a single pulse.

The following example shows an ideal 4-bit digital pattern 4'b1101 (UI width assumed is 100ps for representative purposes). Figure 24 illustrates the creation of patterns using linear superposition of scaled and time-shifted versions of a single pulse p(n) as => 1*p(n) + 1*p(n-1) + 0*p(n-2) + 1*p(n-3).

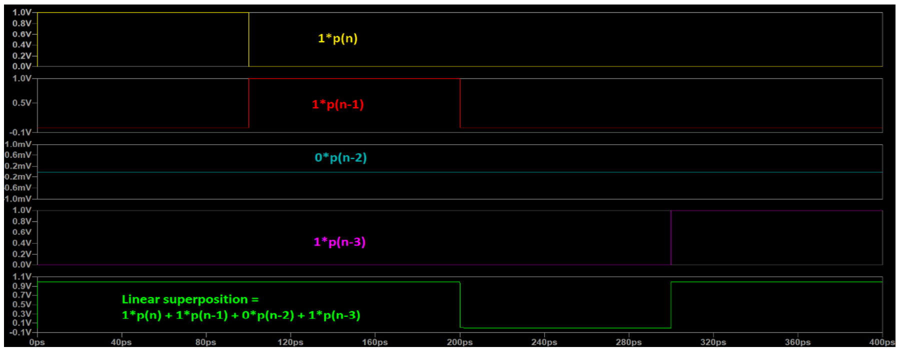


Figure 24: Pattern creation

The interleaving operation of the cross-correlation result sample sets is equivalent to performing a similar scaled and time-scaled linear superposition of the cross-correlation result sample sets samples. The linear fit pulse response will be a single ideal pulse for an ideal transmitter waveform. As shown in Figure 25 below, for representative purposes, using Octave from an ideal PRBS9 pattern using the linear fit pulse response computation approach mentioned before, but for M = 8)

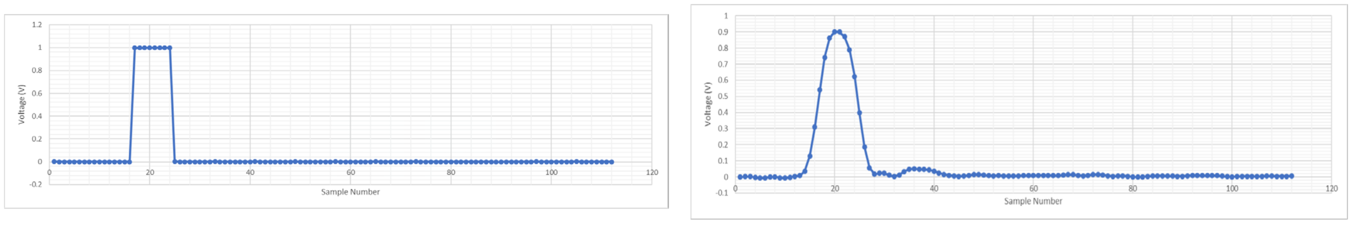


Figure 25: Interleaving and linear fit response

Here, we show the linear fit pulse response computed from a more realistic transmitter sourcing the PRBS9 pattern, again computed using Octave for M = 8). The parameter pmax in the equation 1, is the maximum value of this computed linear fit pulse response.

To compute the σe term, we first construct a waveform by linear superposition of scaled and time-shifted versions of this linear fit pulse response. We then subtract the waveform generated using the linear fit pulse response from the actual waveform captured from the transmitter on a sample-by-sample basis. Let's call the delta between the two waveforms at each sample point E (n), where n represents the sample number. This represents the amount of "distortion" present. The deviations in the waveform shape originate from non-linear effects (such as reflections, Tx nonlinearity, etc.). The parameter σe term is the RMS value of E(n), as in Figure 26.

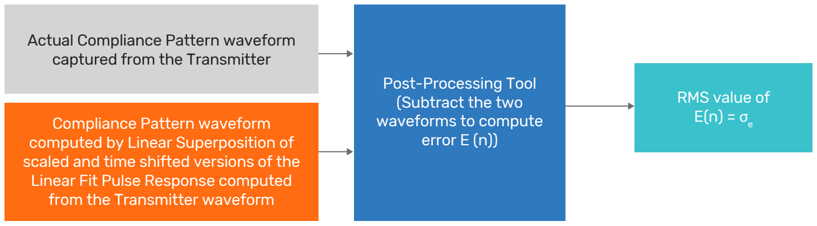

Actual Compliance Pattern waveform captured from the Transmitter
Compliance Pattern waveform computed by Linear Superposition of scaled and time shifted versions of the Linear Fit Pulse Response computed from the Transmitter waveform
Post-Processing Tool (Subtract the two waveforms to compute error E (n))
RMS value of E(n) = σ_e

Figure 26: Calculating σe

To compute the parameter σn, first, we compute the RMS voltage noise at eight equidistant sampling instants from a single UI for each PAM4 level, as indicated by the yellow dots in the image below, for "Nk" repetitions of the compliance pattern. Let them be σ0_1_rms, σ0_2_rms…. σ0_7_rms for PAM4 level 0, σ1_1_rms, σ1_2_rms…. σ1_7_rms for PAM4 level 1, and so on.

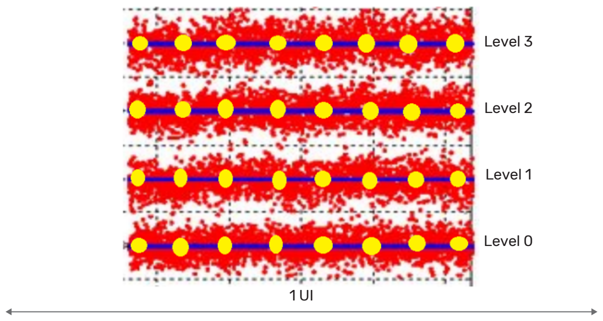


Figure 27: Levels of PAM4

This UI is chosen as the 61st UI of the 64 UI-long "DC" section corresponding to each PAM4 level. This ensures that the noise measurements associated with σn calculations are done on well-settled regions of the waveform.

Then, for each PAM4 level, we compute the square root of the RMS summation of these eight RMS voltage noise values to arrive at a single final effective value. Let it be σ0_rms, σ1_rms, σ2_rms, and σ3_rms respectively, corresponding to the four PAM4 levels 0, 1, 2, and 3.

Then σn is simply the average of these four σLrms values, that is, σn = (1/4)* (σ0_rms + σ1_rms + σ2_rms + σ3_rms).

Transmitter ratio of level mismatch (RLM-TX) measurements

RLM-Tx indicates the linearity of the transmitter output waveform in PAM4, as in Figure 28. It checks the uniform distribution in the four levels of PAM4, the different levels, and their calculation to ensure linearity.

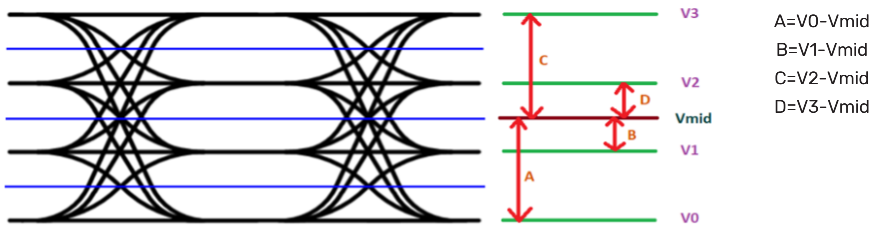


Figure 28: RLM-TX measurements

The BASE 6 PCIe spec defines

Vmid as (V0+V3)/2 and the RLM-TX is expressed as

RLM-TX = MIN((3 × ES1), (3 × ES2), (2 − 3 × ES1), (2 − 3 × ES2))

where, ES1 = B/A, and for an ideal PAM4 eye, B/A will be 1/3.

ES2 = D/C, and for an ideal PAM4 eye, B/A will be 1/3

As per PCIe 6.0 specs, RLM-Tx is expected to be greater than 0.95. We have considered three different cases to show how ES1 and ES2 affect RLM-TX

- Case 1: For, an ideal case, with ES1=ES2 = 1/3, RLM-TX = 1, which passes the specification
- Case 2: ES1 and ES2 are both > 1/3, say 0.4. Now RLM-TX = 0.8, which fails the specification.
- Case 3: ES1 and ES2 are both < 1/3, say 0.3. Now RLM-TX = 0.9, which also fails the specification.

How the Measurement Method for Tx Preset in the BASE 6 spec Is Different from the BASE 5 Spec

The Tx preset measurements were made on settled waveform regions and conducted in a more low-frequency waveform portion. In the context of these measurements, some important differences exist in the PCIe BASE 6 version of the spec. The older method relied on calculating Tx presets based on measurements made on settled regions of the waveform, that is, a more “low-frequency waveform portion” based approach.

The BASE6 spec defines a 4-tap Tx FFE for the transmitter in PCIe 6.0 mode with 2 pre-cursors, one main cursor, and one post cursor, as follows:

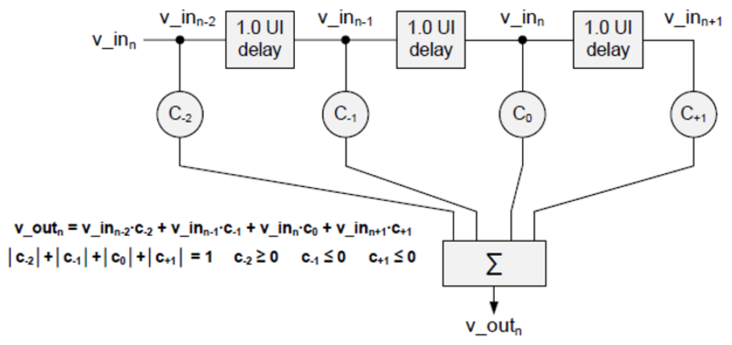

v_in_n
v_in_{n-2}
1.0 UI delay
v_in_{n-1}
1.0 UI delay
v_in_n
1.0 UI delay
v_in_{n+1}
C_{-2}
C_{-1}
C_0
C_{+1}
$v\_out_n = v\_in_{n-2} \cdot c_{-2} + v\_in_{n-1} \cdot c_{-1} + v\_in_n \cdot c_0 + v\_in_{n+1} \cdot c_{+1}$
$|c_{-2}| + |c_{-1}| + |c_0| + |c_{+1}| = 1 \quad c_{-2} \geq 0 \quad c_{-1} \leq 0 \quad c_{+1} \leq 0$
Σ
v_out_n

Figure 29: Tap Tx FFE representation for PCIe 6.0 rate

The Tx preset settings table is defined as follows for the PCIe 6.0 rate, having presets Q0 to Q10:

Preset #	Preshoot 2 (dB)	Preshoot 1 (dB)	De-emphasis (dB)	C_{-2}	C_{-1}	C_{+1}	Va/Vd	Vb/Vd	Vc1/vd	Vc2/Vd
Q0	0.0±0.5dB	0.0±0.5dB	0.0±0.5dB	0.000	0.000	0.000	1.000	1.000	1.000	1.000
Q1	0.0±0.5dB	1.6±0.5dB	0.0±0.5dB	0.000	-0.083	0.000	0.834	0.834	1.000	0.834
Q2	0.0±0.5dB	3.5±0.5dB	0.0±0.5dB	0.000	-0.167	0.000	0.666	0.666	1.000	0.666
Q3	0.0±0.5dB	0.0±0.5dB	-1.6±0.5dB	0.000	0.000	-0.083	1.000	0.834	0.834	0.834
Q4	0.0±0.5dB	0.0±0.5dB	-3.5±0.5dB	0.000	0.000	-0.167	1.000	0.666	0.666	0.666
Q5	-1.3±0.5dB	4.7±1.0dB	0.0±0.5dB	0.042	-0.208	0.000	0.584	0.584	1.000	0.500
Q6	-1.6±0.5dB	3.5±0.5dB	-3.5±0.5dB	0.042	-0.125	-0.125	0.750	0.500	1.750	0.416
Q7	-2.9±0.5dB	4.7±1.0dB	0.0±0.5dB	0.083	-0.208	0.000	0.584	0.584	1.000	0.418
Q8	-3.5±0.5dB	6.0±1.0dB	0.0±0.5dB	0.083	-0.250	0.000	0.500	0.500	1.000	0.334
Q9	-4.4±1.0dB	6.9±1.0dB	-1.6±0.5dB	0.083	-0.250	-0.042	0.500	0.416	0.916	0.250
Q10	0.0±0.5dB	0.0±0.5dB	Note 2	0.000	0.000	Note 2	1.000	Note 2	Note 2	Note 2

Table 3: Tx preset table for PCIe 6.0 rate

In the context of Tx preset measurements, there are some important differences in the PCIe BASE 6 version of the spec versus the PCIe BASE 5 version for all rates where Tx presets are used (i.e., PCIe 3.0, 4.0, 5.0, and 6.0). The older method relied on calculating Tx presets based on measurements made on settled regions of the waveform for the specific Tx preset under test as well as such measurements made on settled regions of the waveform for a suitably chosen different preset value, and so on, that is, a more "low-frequency waveform portion"-based approach.

The significant steps in the new method for measuring Tx presets are as follows:

Source the PCIe compliance pattern with the Tx preset to test, apply, and then extract the step response from this waveform (let's call it "Step_Response_WithTxPreset_Actual") as follows:

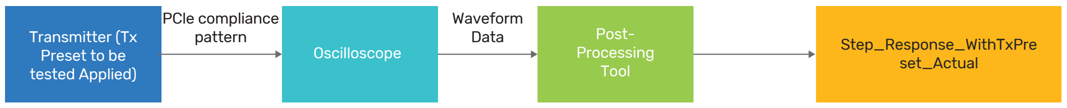

Transmitter (Tx Preset to be tested Applied)
PCIe compliance pattern
Oscilloscope
Waveform Data
Post-Processing Tool
Step_Response_WithTxPreset_Actual

Figure 30: Measuring Tx presets

Source the PCIe compliance pattern with NO Tx equalization applied (Tx preset P4 for PCIe 3.0, 4.0, and 5.0 and Tx preset Q0 for PCIe 6.0) and then extract the step response from this waveform ("Step_Response_NoTxEQ"), as in Figure 31

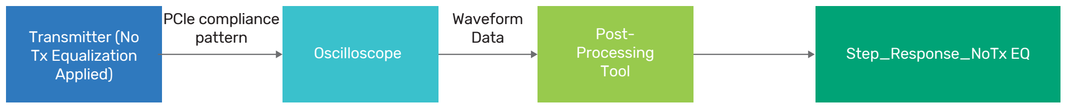


Figure 31: Step_Response_NoTxEQ

Use the post-processing tool to operate upon "Step_Response_NoTxEQ" by iterating through the Tx EQ coefficient settings ([C-1, C0, C+1] for PCIe 3.0, 4.0, and 5.0 and [C-2, C-1, C0, C+1] for PCIe 6.0). This results in "Step_Response_NoTxEQ_processed" until it finds the Tx EQ coefficient settings with which we have the least mean square error (LMS error) between the "Step_Response_NoTxEQ_processed" and "Step_Response_WithTxPreset_Actual" step response waveforms. The following diagram shows this.

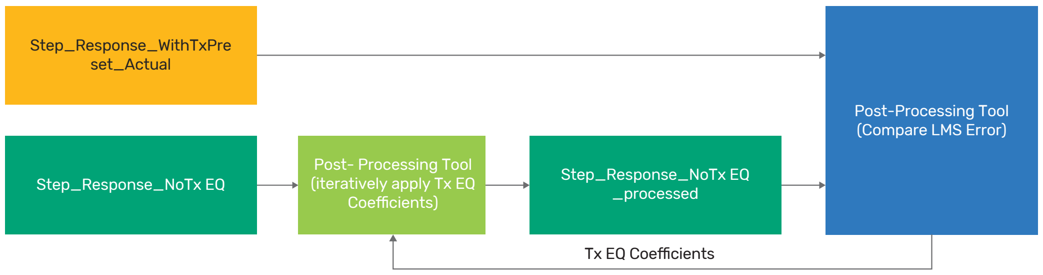

Step_Response_WithTxPreset_Actual
Step_Response_NoTx EQ
Post- Processing Tool (iteratively apply Tx EQ Coefficients)
Step_Response_NoTx EQ _processed
Post-Processing Tool (Compare LMS Error)
Tx EQ Coefficients

Figure 32: Post processing tool

The post-processing tool finds and reports the Tx EQ coefficient settings with which we have the LMS error between the "Step_Response_NoTxEQ_processed" and "Step_Response_WithTxPreset_Actual" step response waveforms, as discussed. These specific Tx EQ coefficient settings found above are deemed effective and realized by the Tx in response to the desired Tx preset setting under test.

Conclusion

It is now just a matter of computing the de-emphasis and pre-shoot values in dB from these Tx coefficient values using the standard formulae and comparing the results with the specification limits corresponding to the specific Tx preset under test for these parameters to ascertain PASS/FAILs. In the ideal case, it's easy to see that the "Step_Response_NoTxEQ_processed" versus "Step_Response_WithTxPreset_Actual" LMS error will be minimal when the exact ideal expected Tx EQ coefficient values corresponding to the Tx preset under test are used to derive "Step_Response_NoTxEQ_processed" from "Step_Response_NoTxEQ."

This process is repeated for each Tx preset to be tested to ascertain PASS/FAILs for each preset setting.

Further Information

1. Flaherty, Nick, "PCIe 6.0 specification approaches release," eeNews Europe, October 7, 2021: https://www.eenewseurope.com/en/pcie-6-0-specification-approaches-release/.
2. Singh, Sejal, "PCIe-All Generations One-Stop Point Log [Ultimate Guide]," Logic Fruit Technologies, February 28, 2022: https://www.logic-fruit.com/pcie/pcie-the-ultimate-guide/.
3. Das Sharma, Debendra, "PCIe®6.0 Specification: The Interconnect for I/O Needs of the Future," PCI-SIG, June 4, 2020: https://pcisig.com/sites/default/files/files/PCIe%206.0%20Webinar_Final_.pdf