Line coding methods for high speed serial links

Abdelaziz Goulahsen[1], Julien Saadé[2], Frédéric Pétrot[2]
([1]) STMicroelectronics, Grenoble, France
([2]) TIMA laboratory, Université Grenoble-Alpes, Grenoble, France
Email: Abdelaziz.goulahsen@st.com, Julien.saade@imag.fr , frederic.petrot@imag.fr

Abstract

A line coding for high speed serial transmission is defined by two major characteristics: the maximum guaranteed run length (RL) which is the number of consecutive identical bits, and the running disparity (RD or DC-Balance) which is the difference between the number of 'zeroes' and 'ones' in a frame. Both should be bounded to a certain limit, RL to ensure reliable clock recovery and RD to limit baseline wander. Another important parameter is the overhead predictability. This parameter may be critical for applications that need a regular synchronization but for other applications, especially if the variable transfer rate is handled by the upper layer protocol, a statistical value of this parameter is good enough. In this paper, we propose two programmable line codings which bound RL and RD with fixed or variable overhead. The resulting overhead for the line coding we propose is shown to be the lowest among the existing methods, as much as to 10 times lower than well-known encoding methods. The fixed overhead line coding is based on a generalization of the polarity bit approach and can be dynamically adapted to link quality and the environment. First we propose a line coding which bounds the RL, and then we propose another one which bounds the RD. We end up by combining both methods to build a DC-balanced and Run Length limited line coding.

Key words

Running disparity, dc-balance, baseline wander, line coding, clock data recovery, polarity bit inverted.

I. Overview on DC-Balanced coding

Line coding applied on data before transmission can be split into 2 families: variable and fixed overhead. In both cases the line coding is characterized by two main parameters: a maximum RL to guarantee frequent transitions for Clock and Data Recovery (CDR) in asynchronous links, and a bounded RD (counted: +1 for a transmitted "1" and -1 for a transmitted "0") to reduce Baseline Wander (BLW) [1]. The smaller the RL and RD bounds, the lesser the constraints are on the CDR unit and on the filters. This helps in reducing the receiver's complexity, power consumption, chip area and Bill Of Materials.

Line coding usually comes at the cost of additional bits; for example, the 8b/10b encoding [2] which is widely used, adds 2 bits for every 8 bits resulting in 2/8 = 25% overhead while ensuring a maximum RL of 5, and a RD bounded to +/- 3 at bit level. The 64b/67b encoding which is used by the Interlaken's protocol, ensures a maximum RL of 64 and a RD bounded to +/- 96 [3] at the cost of 4.68 % of overhead. Table 1 summarizes the characteristics of some existing high speed links line coding methods.

In this paper we propose two methods which bound the RD at the desired value. A first method with very low overhead but not fully predictable in term of overhead cost, and a second one with a predictable overhead but at a higher cost. Both methods exhibit a very low overhead compared to all existing solutions, we then show how, in both cases, we can control the run length to end up with a controlled RD and RL line coding.

In section II we review the methods that were proposed to bound the RD, outline their advantages and drawbacks. In section III to VI we present our DC-balanced coding methods and the simulation results. In section VII we combine the proposed methods with our RL's limited coding. Finally In section VIII we compare the eye diagrams obtained by the different methods.

II. Overview on DC-Balanced coding

Since the early days of data communication, DC-balanced codes have been used to counter the BLW effect which is generally caused by AC-coupling [1] and results in reducing the eye diagram opening. BLW can be also observed in DC-coupled devices as we showed in the eye diagrams in [4].

In 1986, Knuth proposed a method [5] to construct frames with equal number of 0's and 1's. Knuth proved that any binary sequence of a specific size can be balanced by inverting, at a specific bit position, all the rest of the sequence. The drawback of this method is that this particular bit position must be sent with the frame (and should be balanced as well) for the receiver to know how to reconstruct the original frame. This will add a relatively large number of bits for small frames. For large frames, the number of added bits is lower, but the RD could reach high values inside the frame before going back to zero.

A low overhead method is the polarity-bit coding. It add 1 bit to a frame of a certain size to indicate whether it is inverted or not depending on the Cumulated RD (CRD) and the RD of the frame itself; *i.e.* if the CRD is positive, and the RD of the frame is positive as well, all the bits inside the frame will be inverted and the polarity bit will transmit the info to the receiver. This method is used by the 64b/67b encoding; 3 bits are added to the 64 bits of the frame: 2 bits ('10b' or '01b') to ensure a transition and indicate whether the frame is raw data or control, and 1 polarity bit to indicate if the 64 bits are inverted or not. The CRD bounds ensured by such coding is computed according the worst case scenario as following:

$$CRDbounds = +/- (FrameSize + FrameSize/2) \quad (1)$$

Which gives for the 64b/67b encoding $CRD_{bounds} = +/-96$ for a FrameSize = 64. The overhead cost for the CRD bound is 1/64 = 1.56 %. The total overhead cost is 3/64 = 4.687 %.

The 64b/66b, 128b/130b and 128b/132b encoding rely on scrambling-only for the 64 or 128 bits of the frame. As we showed in [4], scrambling creates a sort of balancing between 0's and 1's, so in average it reduces the CRD. It does however not give bound guarantees, and could reach high values as we can see in figure 1. The advantage of scrambling is that it does not add any extra bits, so it has no overhead.

Recently, a method that bounds the RD was presented in [6] at 0% overhead. But this method can ensure the bounds only when the data is constant or idle.
In the following section, we will introduce two new methods which bound the Running Disparity to the desired limit with a very low overhead down to10x lower than existing encodings in both predictable and non-predictable overhead.

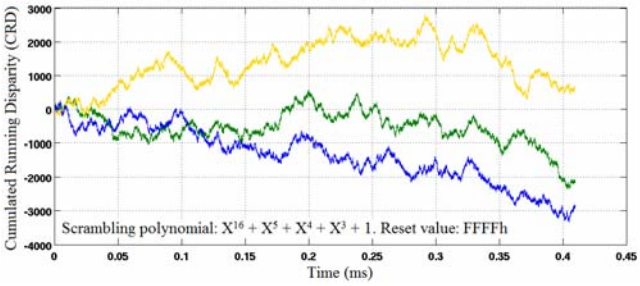

Figure 1 Three random examples of the Cumulated RD of Scrambled frames of 4 Megabits each at 10 Ghz

Table 1 Summary of some existing Line Coding

Line Coding	Standards	RL bound	RD Bound	Overhead
8b/10b	PCIe 2.0, USB 3.0 …	5	+/- 3	25 %
TMDS	HDMI (1)	+/-20	+/-20	25%
64b/66b	10G Ethernet	64	N/A	3.125 %
64b/67b	Interlaken	64	+/- 96	4.687 %
128b/130b	PCIe 3.0	128	N/A	1.562 %
128b/132b	USB 3.1	128	N/A	3.125 %

(1) Clock forwarded

III. Novel method to bound the Running Disparity using aperiodic polarity bit insertion

The proposed method computes the CRD bit-by-bit and when the CRD reaches a positive or negative Threshold T, the RD of the next packet of Size 'S' bits is checked to see if it should be inverted, or not. A bit will be added after the S bits to indicate if they were inverted or not. Only when RD(S) = 0, will there be no bit added. The algorithm is as follows:

- If CRD = +T and RD(S) > 0, the S bits are inverted and a '1' bit is added to indicate it to the receiver.
- If CRD = -T and RD(S) > 0, the S bits are not inverted and a '0' bit indicates it to the receiver.
- If CRD = +T and RD(S) < 0, the S bits are not inverted and a '0' bit indicates it to the receiver.
- If CRD = -T and RD(S) < 0, the S bits are inverted and a '1' bit is added to indicate it to the receiver.
- If CRD = +/-T and RD(S) = 0, the S bits are not inverted and **no** bit is added. The receiver knows when RD(S) = 0 that the bits were not inverted by default.

This allows the receiver to recover the data. Figure 2 illustrates the Transmitter and an example. S bits should always be buffered and RD(S) is calculated permanently. The values of T and S should be agreed on by the

transmitter and the receiver before the start of transmission and can be adjusted to the link quality and environment.

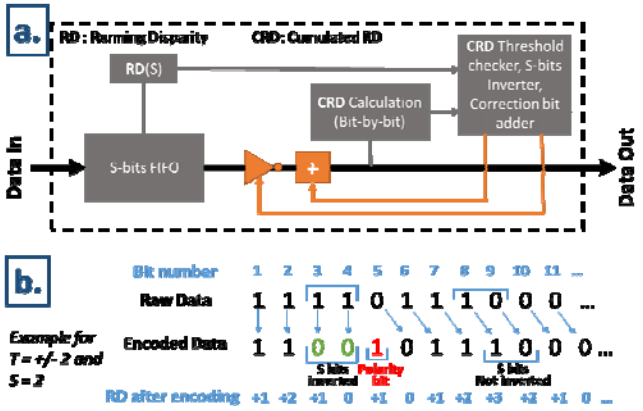

Figure 2 a. Proposed DC-Balancer's block diagram b. Encoding example

The CRD bounds ensured by our proposal are defined by:

$$CRDbounds = +/- (T + S/2) \quad (2)$$

Two conditions should be respected to ensure the bounds mentioned in equation (2):

- T > S/2 and
- S is even

IV. Simulation result for aperiodic polarity bit insertion

In figure 6, we plot in red the CRD of a random frame of 200 Kbits scrambled, and then we apply our proposed balanced encoding for T = 64 and S = 64. The CRD for the balanced frame is shown in green. We can see that the CRD never exceeds +/- 96. The encoding we propose can be applied without scrambling, but scrambling is recommended to reduce the CRD of the initial frame, limit the CRD excursion and thereby reduce the overhead (the added aperiodic polarity bits).

The following figure shows an example of scrambled data's CRD before and after applying the proposed DC balancer with T = 64 and S = 64 (CRD bound = +/-96)

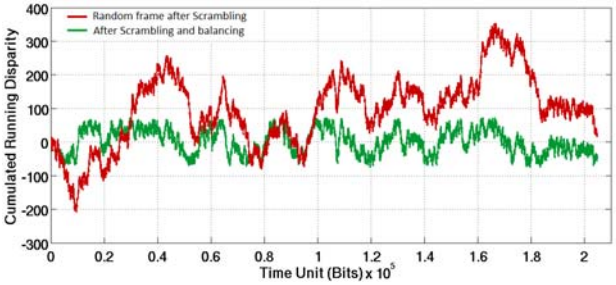

Figure 3 Scrambled data's CRD before and after applying the proposed DC balance

A. Overhead estimation of the proposal

$$OH = added\_bits / Raw\_bits \quad (3)$$

To estimate the overhead (OH) resulting from our coding, we generate random frames on Matlab, scramble them with the scrambling polynomial mentioned in Figure 1, and then apply our line coding. The overhead is shown in Figure 4 and Table 2, and it is calculated according to equation (3) based on the simulation of 200 frames of 400 Kbits each. Averaging is then done.

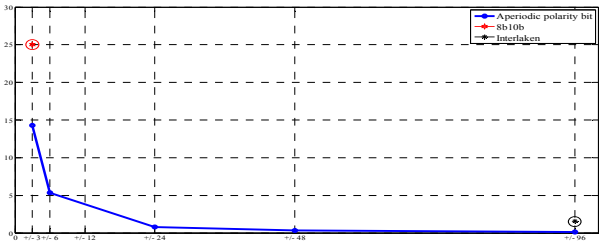

Figure 4 Proposal's overhead compared to 8b/10b encoding and Interlaken's protocol

Table 2 Overhead examples of the proposed method

T	S	CRD bounds	Overhead
2	2	+/- 3	14.27 %
3	2	+/- 4	9.05 %
4	2	+/- 5	6.6 %
5	2	+/- 6	5.32 %
9	6	+/- 12	2.05 %
16	16	+/- 24	0.8 %
32	32	+/- 48	0.31 %
64	64	+/- 96	0.11 %

As we can see, the overhead due to our proposal is very advantageous compared to other encodings. For a CRD bounded to +/- 3, we have more than 10% overhead reduction compared to 8b10b. If we release the constraints of the RD, we can bound the CRD to low values with only a few percent of overhead or even less than 1%. Compared to the Interlaken's protocol which adds 1.56% to bound the CRD to +/- 96, we can obtain the same bound with only 0.11% overhead which is more than 10x lower.

This method also bounds the Run Length to $2*CRD_{bounds}$. *i.e.* if we bound the CRD to +/-3, the RL will be automatically bounded to 6 because the worst case RL will be then going from a CRD of -3 to a CRD of +3.

We note that based on the frame's 1's and 0's distribution, the overhead estimations for our proposal can be increased if scrambling is not applied, that's why scrambling is recommended to minimizes the overhead.

V. Novel method to bound the Running Disparity using periodic polarity bit insertion

As defined in the overview, a periodic or aperiodic polarity bit insertion method relies on scrambled data. As also shown in Figure 1, the CRD of a scrambled data is not bounded at all, it is only statistically reduced over a large time window.

In order to overcome this issue, we use the probabilistic representation of the random logic. This representation has been used in [7] to obtain a target probability on the output of a system given the probability of the input.

Scrambled data is defined as a system where the output probability of having a "0" (P(0)) should be around 0.5 ($0 \leq P(0) \leq 1$) whatever the input's probability value.

We first define the output probability for the simple logic gate by the following equation:

Inverter: P(Z=1) = 1-P(A=1) (4)

And: P(Z=1) = P(A=1)*P(B=1) (5)

Or: P(Z=1) = P(A=1)+P(B=1)- P(A=1)*P(B=1) (6)

P is the probability, A and B are inputs, and Z is the output.

This equation should be interpreted as follows: for the inverter: the probability of having a "1" on the output is equal to 1 minus the probability of having a "1" on the input.

Using the equations (4) to (6) we can write the XOR equation.

XOR: P(Z=1) = P(A=1) * (1-P(B=1))+(1-P(A=1)) * P(B=1) - (P(A=1) * (1-P(B=1)) * (1-P(A=1)) *P(B=1)) (7)

Using (4) + (7) we can deduce the equation for the NXOR as follow:

NXOR: P(Z=1) = 1 - P(A=1) * (1-P(B=1))+(1-P(A=1)) * P(B=1) - (P(A=1) * (1-P(B=1)) * (1-P(A=1)) *P(B=1)) (8)

The following figure show the 3D representation of the equation (7) and (8)

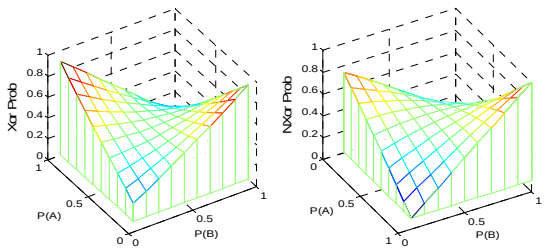

Figure 5 XOR and NXOR output probability in function of the input probability.

We can notice on both curves that if one of the inputs is around 0.5 the output probability will be around 0.5 whatever the probability of the second input. Using a simplified "fix point iteration" or fixing one of the inputs to the output (delayed) we get the following curve.

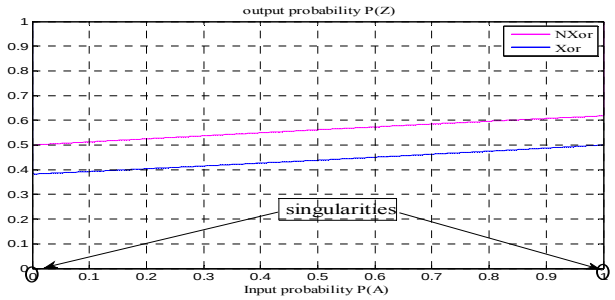

Figure 6 XOR and NXOR output probability as a function of the input probability with one input connected to the delayed output.

In this curve we have excluded the 2 singularities which will be treated in the simulation section.

We see that if one input is connected to the output whatever the second input probability value is, we will have an output probability in the range of 0.4 and 0.6 depending whether we use a XOR or NXOR output. We have put in place the required tool to define the polarity bit implementation. Using P(A) as the message probability from 0 to 1 as not predictable, we have an output probability in the range of 0.4 to 0.6 using either the XOR or NXOR output. The choice should be made in order to always minimize the CRD. This choice (polarity bit) will be always added to the message. The disparity bound will be defined by the size of this message and can be computed by the equation (1)

The following figure shows a simplified implementation example.

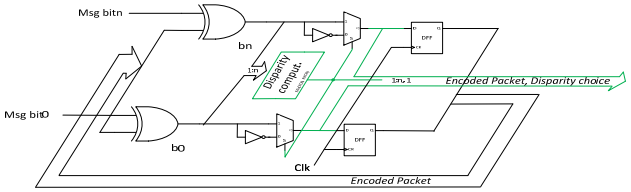

Figure 7 Programmable periodic polarity bit insertion implementation.

VI. Simulation result for periodic polarity bit

Unlike the aperiodic polarity bit insertion this method gives a fixed overhead versus the data processed. The overhead is given by the equation (3) with Raw_bits=message size and added_bits=1.

Thanks to this particular scrambling the average CRD is much lower than the maximum bound. Table 3 gives the theoretical bound and simulated values, over 50e6 bits.

Table 3 Overhead for periodic polarity bit

Msg (bit)	CRD bounds	Simulated CRD (5e6Bits)	Overhead
16	+/- 24	+/-19	6.2 %
32	+/- 48	+/-24	3.1 %
64	+/- 96	+/-33	1.5 %
128	+/-192	+/-47	0.8 %

Using a Markov chain model for this particular scrambling method, we can estimate the time occurrence when the maximum bound is reached. The Figure 8 gives this estimation.

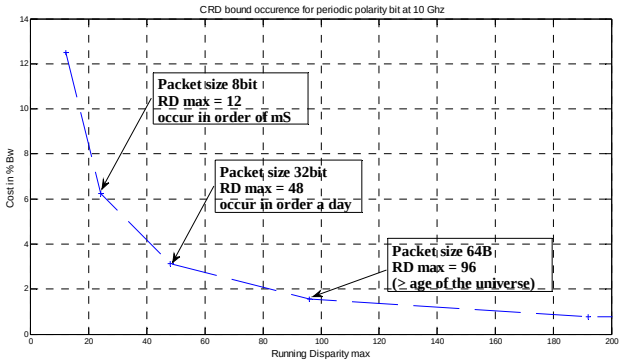

Figure 8 CRD occurrences for periodic polarity bit in time for a 10 Ghz link.

We have mentioned that 2 singularities exist. This occurs when the message is constant. This repetitive message will radiate some peaks of fixed frequency during the transfer. In this case the method still works but EMI and coexistence issue will occur. This case is shown in Figure 9 where the same data is encoded using 8b10 (8b10b) encoding, USB3 scrambling (PN) and periodic polarity bit (PPB).

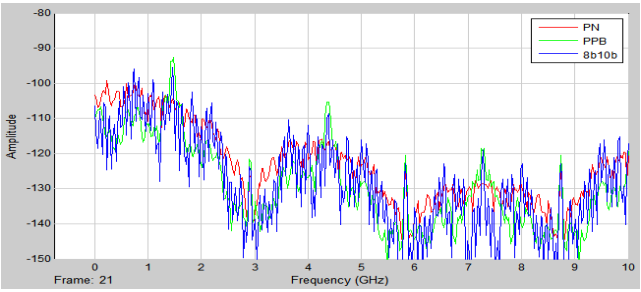

Figure 9 Power Spectral Density of a constant data

To overcome this issue the input message should be combined with a variable and predictable data. Variable for whitening the signal and predictable to have the receiver (Rx) and the transmitter (Tx) synchronized. A very simple way to do it is to use a shift register and xor the message before the processing on the Rx and Tx side. The following figure shows the EMI mitigation using a simple shift register with an unbalanced RD in it.

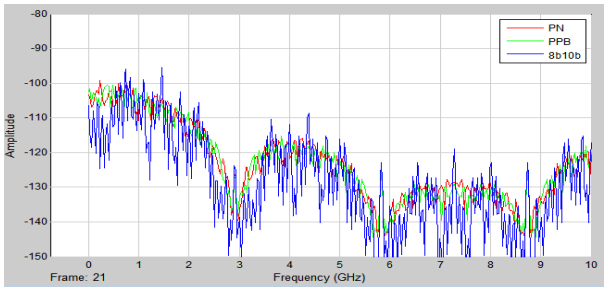

Figure 10 Power Spectral Density of a constant data with EMI mitigation

VII. Low Overhead RL limited Line Coding.

Method to bound the Run Length

In a previous article [4] we made a proposal based on scrambling followed bit stuffing, plus a method to reduce the EMI (Electro-Magnetic Interferences). Bit Stuffing (BS) adds an inverted bit after N consecutive identical bits; i.e. if N = 5, whenever a run of 5 consecutive 1's is detected, the transmitter will add a '0' bit after this run whatever comes next, creating a transition after a minimum of 5 bits. The receiver will benefit from this transition to recover the clock. It will then delete the added bit. The drawback of BS, when applied on raw data, is that the overhead can be significant. But in [4], we showed that if scrambling is applied prior to BS, the BS overhead decreases. The following table gives the overhead for different RL when scrambling is applied.

Table 4 Bit stuffing overhead for different values of RL

RL bound	3	4	5	6	7	8	9	10
OH (%)	16.65	7.13	3.33	1.61	0.79	0.39	0.19	0.09

Combining controlled RD and RL.

We now need to combine both methods: the one that limits the RD (based on periodic or aperiodic polarity bit insertion), the other (BS) that bounds the RL. Analysis shows that if the two methods are put together, one will disrupt the other; i.e. if the balancing is applied after BS, the BS will be disrupted due to bit inversion and the result will be more consecutive bits than the BS has ensured. If BS is applied after balancing, balancing will be disrupted by the BS. For example, if the BS adds more 1's than 0's, the RD will end up diverging.

One way to make both methods work together is to process the running disparity balancing by one of the previous methods described, then apply a ***modified*** bit stuffing (MBS).

Instead of adding a '0' bit after N consecutive 1's or a '1' bit after N consecutive 0's, we will add '01' after N consecutive 1's and '10' after N consecutive 0's. The

balancing will not be disrupted because the added RD is zero.

If we consider a pattern that has been balanced to a bound of +/-12, when we apply the modified bit stuffing for N = 5, the CRD bound won't change, not even by 1 unit. To exceed the limit of the CRD to +13 for example, we should have a CRD to +12 and then do the MBS by adding '10' so that the CRD can go to +13 when we add the '1' bit. However, to have a CRD to +12 and MBS by '10', it should happen with 5 consecutive 0's. In this case, it is impossible to have a CRD at +12. Thereby, the MBS will not change the CRD bounds of the balancing.

The Total Overhead (TO) of the RL limited and DC-Balanced Line Coding is given by the following equation:

$$TO = BO + MBSO \quad (9)$$

Where *BO = Balancing's Overhead and MBSO = Modified Bit Stuffing's Overhead*

VIII. Eye Diagrams Simulation.

To compare the performance of the proposed line coding and to make a comparison with 8b/10b encoding and scrambling, we plot in Figure 11 using Simulink the eye diagrams using the S-parameters of a DC-coupled PCB (Printed Circuit Board) long channel. At 10 GHz, we can see that Scrambling's eye is the most closed, and when we bound the RD with our proposal to +/-3 and RL bound to 5, we clearly see a better opening. It is very similar to 8b10b encoding because of the same bounds.

For a 10 GHz frequency, the real throughput using 8b/10b encoding is 8 Gbit/s (10-10*25% = 8). To have an equivalent throughput, the frequency of the link using our encoding is 9.4 GHz (8+8*17.4% = 9.4) according to the overhead given in Table 5. The eye diagram is shown in Figure 11 and we can see that it has a much better opening than with 8b/10b encoding at the equivalent throughput.

IX. Conclusion.

In this paper, we presented two novel line codings that bound the RD with a very low overhead and show how a modified bit stuffing can be used to bound the RL. These line codings are targeting two types of application

-Variable data rate transfer: the variation is directly proportional to the parameter used and at most in order of few percent of the total bandwidth.

-Fixed rate data transfer: where the overhead is a little bit bigger than the "variable rate" propositions but still remains in the order of a few percent.

The choice could be made based on the final application. Typically for video where the blanking position is important, the periodic polarity bit insertion fits better whereas for servers, the aperiodic polarity bit insertion is preferable.

Our low overhead line codings have many advantages; either to have better throughput efficiency for a specific link frequency, or to have lower frequency at a fixed target throughput. Lower frequency means better eye opening, lower noise and lower power consumption.

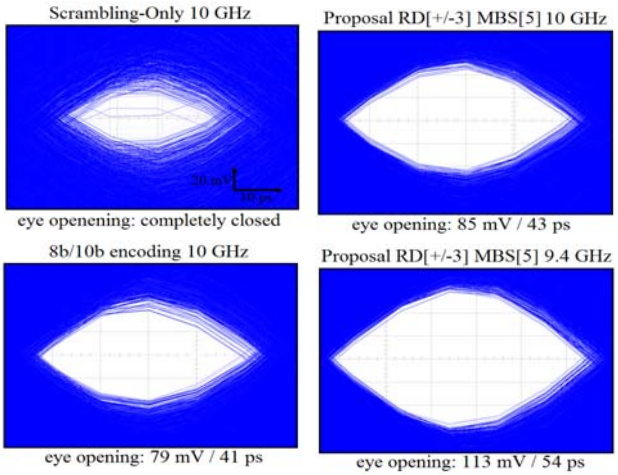

Figure 11 Eye diagrams comparison (DC-coupled channel, 100 Kbits, Tx swing 800mV, No equalization)

Table 5 Overhead breakdown for aperiodic polarity bit insertion

Balancing			B O	RL Bound	MBSO	TO (Total Overhead)
T	S	RD				
2	2	+/- 3	14.27 %	5	3.13 %	17.4 % *
3	2	+/-4	9.05	6	1.65 %	10.7 %
5	2	+/- 6	5.32 %	5	5.43 %	10.75 %
7	6	+/- 10	2.66 %	10	0.11 %	2.77 %
15	10	+/-20	1.03 %	8	0.71 %	1.75 %
64	64	+/- 96	0.11 %	7	1.56 %	1.67 %

References

[1] N. Na, D.M. Dreps and J.A. Hejase, *"DC Wander Effect of DC Blocking Capacitors on PCIe Gen3 Signal Integrity"*, IEEE 63rd Electronic Components and Technology Conference (ECTC), 2013

[2] P.A. Franszek and A.X. Widmer, *"A DC-Balanced, Paritioned-Block, 8B/10B Transmission Code"*, IBM Journal of research and development, Volume 27, Number 5, September 1983

[3] Interlaken Protocol Definition, Revision 1.2, October 7, 2008

[4] J. Saadé, A. Goulahsen, A. Picco, J. Huloux, F. Pétrot, *"A scalable low overhead line coding for asynchronous high speed serial transmission*", IEEE 18th Workshop on Signal and Power Integrity (SPI), 2014

[5] D.E. Knuth, *"Efficient Balanced Codes"*, IEEE transactions on Information Theory, vol it-32, no.1, January 1986

[6] W.M. Barrett, *"Disparity Reduction for High Speed Serial Links"*, U.S. Patent 2014/0064357 A1, March 6, 2014

[7] Weikang Qian, Marc D. Riedel, Hongchao Zhou, and Jehoshua Bruck *"Transforming Probabilities with Combinational Logic"*, IEEE Transaction on computer-aided design of integrated circuits and systems, vol. 30, No. 9, September 2011