White Paper on the MB810 Line Code for 10GbE

Dae Young Kim, Changoo Lee, Chun Sik Shin, Hae Won Jung, and Hyeong Ho Lee

Abstract

This paper describes a new line code called MB810 proposed for use with 10GbE being standardized by IEEE 802.3. The key feature of the code is that it preserves many good code properties of the 8B10B code, a 1GbE standard and a strong candidate even for 10GbE, and yet consumes half the bandwidth. The theoretical background, design principles, performance characteristics, and implementation issues are given.

I. Introduction

IN pursuit of line codes for 10 Giga-bit Ethernet (10GbE), binary block codes persist to be competitive, as it did with 1GbE, due to many good features inherently provided:

- Being binary, they fit nicely with optical channels suffering from nonlinearity which cannot be perfectly eliminated in practiced engineering.
- Being binary, they offer the maximum receive signal-to-noise ratio(SNR) for the same given conditions, e.g., transmit optical power and optical span, compared to other multilevel codes.
- By block coding, they can be made inherently run-length limited(RLL).
- They can be easily made to be DC-free.
- By block coding, it is easy to provide for extra control symbols.

In spite of all these advantages, the major concern playing against adopting the block coding is the fact that they usually require more channel bandwidth than other choices due to the increased line rate. When using 8B10B for 10GbE, for example, the main-lobe bandwidth, i.e., the bandwidth to the first power spectral null, is 12.5GHz, 25% more than with Scrambled NRZ.

This White Paper is to introduce a code which exactly alleviates this bandwidth dilemma of binary block coding. Our code called MB810 encodes each 8 data bits into 10 line bits as 8B10B does, yet with half the resultant line bandwidth; the main-lobe bandwidth for 10GbE is 6.25GHz.

Design of such a code has been possible by making novel use of a theorem established and published by the author [1]. The theorem establishes a condition for a digital signaling system to operate within the theoretical minimum-bandwidth(MB) dictated by Nyquist, hence usually called the Nyquist bandwidth. The Nyquist bandwidth is, by definition, half the signaling frequency. With most usual non-MB codes including NRZ, 8B10B, and PAM5, the line bandwidth is as wide as the signaling frequency.

II. Theoretical Background

Consider a digital transmission system consisting of

- a line coder
- a pulse shaper
- a receive sampler
- a line decoder.

Usually, the function of a pulse shaper is distributed among a transmit(TX) filter, the optical channel, and a receive(RX) filter. The pulse shaper represents the whole channel between the outlet of the TX line coder and the inlet of the RX sampler.

D. Y. Kim is with the InfoCom Dept., Chungnam National University, Taejon 305-764, Rep. of Korea(email: dykim@ccl.chungnam.ac.kr).

C. Lee, C. S. Shin, H. W. Jung, and H. H. Lee are all with the Router Technology Division, the Electronics and Telecommunications Research Institute, Taejon 305-606, Rep. of Korea(email: cglee@etri.re.kr).

Assume the line coder outputs a symbol every T seconds, and let y_n denote the coder output symbol at $t = nT$. Then consider a code parameter called running alternate sum(RAS):

$$RAS \equiv \sum_{n=I}^{J} (-1)^n y_n, \tag{1}$$

where I and J are integers. RAS is the sum of the coded output symbols within an arbitrary interval between $t = IT$ and $t = JT$, yet modified with alternating polarity.

Then further consider an associated code parameter called alternating sum variation(ASV) defined as

$$ASV \equiv \max_{I,J,\{y_n\}} |RAS| = \max_{I,J,\{y_n\}} \left| \sum_{n=I}^{J} (-1)^n y_n \right|. \tag{2}$$

That is, ASV is the peak-to-peak variation of RAS measured over entire coded output symbol stream.

One thing to be careful in evaluating DSV and ASV according to Eqs. 1,2 is that the coded line symbols y_n are assumed to take on levels of normalized spacing. That is, in the case of binary signaling, each y_n takes on $-1/2, 1/2$. Therefore, logical line symbol '1' and physical line symbol '1/2' will be used interchangeably in this paper. The same will be true of '0' and '-1/2'.

By use of ASV, the following theorem can be proved [1]:

Theorem 1: If ASV is finite, the code has a spectral null at the Nyquist frequency.

The *Nyquist frequency* f_N is, by usual definition, half the signaling frequency R:

$$f_N \equiv R/2 = 1/2T.$$

And a *spectral null* of a code means two things:

- There is no line spectrum (discrete power spectral component) at the associated frequency.
- There is a notch in the continuous power spectrum at the associated frequency.

A code with a spectral null at the Nyquist frequency is usually called a Nyquist-free code. Thus the above theorem is equivalent to:

Theorem 2: If ASV is finite, the code is a Nyquist-free code.

Nyquist-free codes possess an extra yet very important MB property [1]:

Theorem 3: A Nyquist-free code is a minimum-bandwidth(MB) code.

An MB code is a code whose output symbol stream can be passed free of inter-symbol interference(ISI) through a bandwidth not greater than the Nyquist bandwidth. That is, with an MB code, the necessary channel bandwidth is only the Nyquist bandwidth.

By use of a system modeling introduced at the beginning of this section, an MB code can also be defined to be a code which can be pulse-shaped by a sinc pulse $sinc(t/T)$ without closing the resultant eyes in the RX eye diagram for any arbitrary source data input to the TX line coder. Pulse shaping by a sinc pulse is equivalent to filtering by an ideal (brick-wall) low-pass filter(LPF) of a Nyquist bandwidth. Also note that the vertical eye opening (eye height) will remain intact since the shaping pulse is ISI free. However, the horizontal eye opening (eye width) will be affected by the choice of the source data pattern. What this definition of the MB property states is that the eye will remain horizontally open for any and even the worst-case source data pattern.

In the case of non-MB line codes, a usual method of reducing the necessary channel bandwidth for ISI-free transmission is to pulse-shape the code symbols by a raised-cosine filter with a roll-off factor less than one. However, for random binary source data input, the horizontal eye width of such a system shrinks proportionally with the roll-off factor. With zero roll-off factor, i.e, with the Nyquist bandwidth, horizontal eye widths of such a code reduce to zero, closing the RX eyes.

What if the pulse is not ISI-free? For example, in high-speed optical digital transmission, the channel response is hardly shaped to an ideal ISI-free form. The TX pulse is usually a square pulse with 100% duty cycle. The composite of the optical path and the RX filter is approximately a second-order LPF. It can be experimentally demonstrated that with the same pulse shaping(TX filter + Channel + RX filter), MB codes render eyes considerably wider than non-MB codes. More on the performance issue will be discussed in Sec. VII.

Before stepping to the next section, one might recall a familiar code parameter related to RAS and ASV. Code designers have used a code parameter called running digital sum(RDS) defined as

$$RDS \equiv \sum_{n=I}^{J} y_n. \tag{3}$$

The peak-to-peak variation of RDS is called digital sum variation(DSV):

$$DSV \equiv \max_{I,J,\{y_n\}} |RDS| = \max_{I,J,\{y_n\}} \left| \sum_{n=I}^{J} y_n \right|. \tag{4}$$

And the following theorem holds:

Theorem 4: If DSV is finite, the code is a DC-free code.
Note the similarity between the ways how concepts of the DC-free and the Nyquist-free property are derived. Yet, the Nyquist-free property accompanies a very important extra property; MB property.

Also note that the smaller the value of DSV or ASV, the more profound the DC-free or the MB property; the smaller the DSV, the wider the spectral notch at zero frequency; the smaller the ASV, the wider the spectral notch at the Nyquist frequency and the eye width.

III. Basic Code Design Building Block: BUDA Cell

With the knowledge of ASV and DSV, assume we want to design a DC-free MB code. There might be many ways to do the job, but one strategy might be to seek for a short binary sequence for which RDS and RAS are both zero in a self-contained way. Then a code design based on a finite accumulation of this zero-sum sequence will always ensure finite DSV and ASV values and thus a DC-free and MB code.

Consider a binary sequence $'1100'$. Its RDS is zero:

$$d = (1/2) + (1/2) + (-1/2) + (-1/2) = 0. \tag{5}$$

And its RAS is also zero:

$$a = (-1)^n(1/2) + (-1)^{n+1}(1/2) + (-1)^{n+2}(-1/2) + (-1)^{n+3}(-1/2) = 0. \tag{6}$$

It is also easy to see, by use of Eqs. 4,2, that the peak-to-peak variations, i.e., DSV and ASV, are both one.

Another way of deriving the code parameters of the sequence is to draw the trace of the sequence over the RDS-RAS plane as done in Fig. 1. Assume that the RDS and RAS are zero at the start of the sequence. Then as each bit of '1100' is output, the RDS changes step-wise as '+1/2, 1, +1/2, 0', returning to the equilibrium. In the case of RAS, the value of n in Eq. 6 has some bearing. If n is even, the RAS changes as '1/2, 0, -1/2, 0', returning to the equilibrium as shown in Fig. 1(a). If n is odd, on the other hand, the RAS changes as '-1/2, 0, +1/2, 0', also returning to the equilibrium as shown in Fig. 1(b).

Therefore the '1100' sequence draws a diamond-shaped cell on the RDS-RAS plane. Recall that DSV is the peak-to-peak variation of RDS by definition. Then from Fig. 1, the horizontal width of

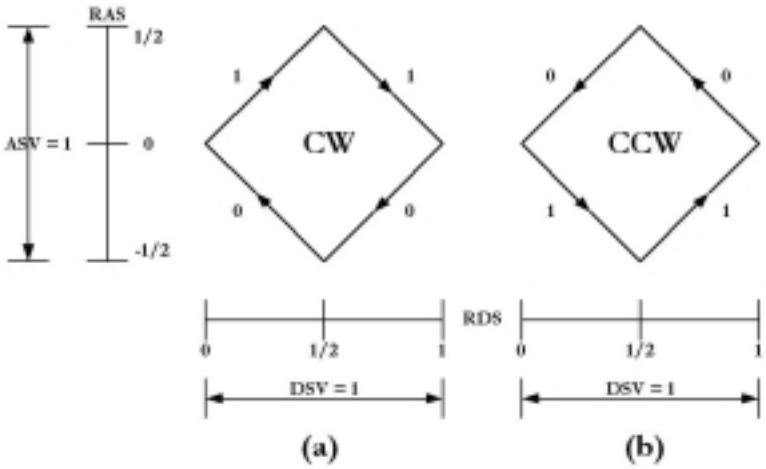

Fig. 1. BUDA cell. (a) CW BUDA (b) CCW BUDA

the cell corresponds to RDS, which is one. A similar observation is true of RAS and so the vertical width of the cell corresponds to RAS, which is also one. Thus the diamond-shaped cell represents a sequence of unit DSV and ASV, thus is named BUDA for 'binary unit DSV and ASV.'

As seen from Fig. 1, there are two kinds of BUDA's. Fig. 1(a) is a clockwise(CW) BUDA whereas Fig. 1(b) is a counterclockwise(CCW) BUDA. It is also interesting to notice that right-arrowed branches indicate logical symbol '1's(physical symbol 1/2) whereas left-arrowed branches indicate logical symbol '0's(physical symbol -1/2).

Although we described the concept of BUDA by use of the '1100' sequence, it is to be noted that the same line of logics apply to any circular shift of the sequence. That is, all of '1100', '0110', '0011', and '1001' are equivalent in the sense of BUDA.

BUDA can be utilized as a basic building block for designing binary DC-free MB code. Next few sections illustrate such usage.

IV. Simple Design Examples

Consider we want to design a (1,2) binary MB block code, i.e.; one input data bit will be coded to two output line bits. We do this by constructing a necessary state transition diagram by accumulating a minimal number of BUDA cells.

Consider a thus constructed BUDA state diagram shown in Fig. 2. Four BUDA cells are stacked by overlapped branches of the same direction of arrows. Two cells are CW BUDA's while the other two are CCW BUDA's.

Four nodes are used as states while other nodes are merely transit nodes to be visited in moving from one state to another.

To understand the coding procedure, assume the coder is at S1 at the start. Because the number of input bits is one, there needs to be two transition paths to other states. And because the number of output bits is two, each such path should consist of two-hop branches; each BUDA branch represent one line bit. With these criteria in mind, we find that there are two two-hop paths out of S1; one is to S2, and the other is to S4. The path S1-S2 represents two line bits '01'. The path S1-S4 represents '11'. Recall, in identifying the line bits along transition paths, each right-arrowed branch represents '1' while each left-arrowed one does '0' in BUDA.

Studying the cases of other three states, we find that the same holds true of them; there exists two two-hop paths out of each state; each such path terminates on another state. In all, we have a successful BUDA stack for our MB12 code.

Yet, there remains one more degree of freedom about which exit path is to be triggered by which input bit value. At S1, for example, shall input '0' trigger the '11' transition to S4 or the '01' transition

TABLE I
MB12 CODEBOOK

state	input	output	next state
S1	0	11	S4
S1	1	01	S2
S2	0	11	S3
S2	1	10	S1
S3	0	00	S2
S3	1	10	S4
S4	0	00	S1
S4	1	01	S3

to S2? Similar questions hold for other states, too. And just here lies a design freedom.

In further studying the output bits along the transit paths out of each state, we note that one of the two exit paths out of a given state represent two bits of the same value whereas the other represent two bits of different values. Therefore, one choice in mapping from an input bit to two output bits is to map a '0' to the two output bits of the same value and to map a '1' to the two bits of different values. We then construct a state transition table or a codebook as shown in Table I. At S1, an input '0' triggers an output '11' and a state transition to S4 whereas an input '1' triggers an output '01' and a state transition to S2. And the like with other states.

The coding of MB12 can also be represented by a conventional form of state transition diagram which is shown in Fig. 3.

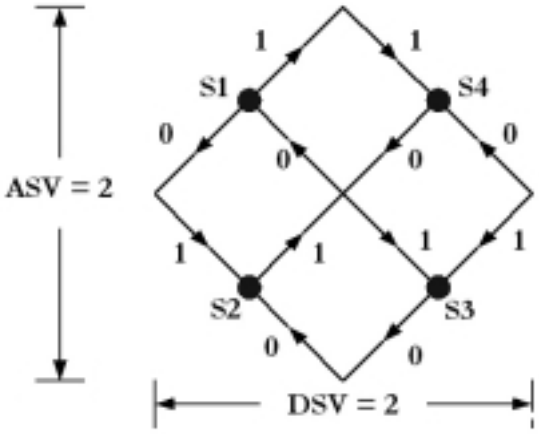

Fig. 2. BUDA cell stack of MB12

MB decoding is simple due to a novel mapping rule adopted in coding. That is, Exclusive OR of the two receive bits is the decoded data bit; '00' and '11' maps back to '0' while '01' and '10' do to '1'.

Now what about DSV and ASV of MB12? It is trivially easy to evaluate the parameters. See Fig. 2 and observe that both DSV and ASV of MB12 are 2. Since both DSV and ASV are finite, MB12 is a DC- and Nyquist-free code, i.e., a DC-free MB code. It should also be noted that MB12 is also strictly RLL; the run-length limit is three. In all, MB12 is a binary DC-free RLL MB code.

One can exercise the coding principles established in Sec. III and exemplified in this section to design one's own binary MB codes, e.g., MB24 and MB34. As noted above, there are a fair degree of freedom in exercising the coding principles to arrive at different design results.

Our design choices of MB24 and MB34 are shown in Figs. 4,5. MB24 neatly resulted in one single state. It consumes the same number of BUDA cells as MB12 does and so has the same DSV and ASV values of 2. The difference is that the central node is the one and only state. Decoding is

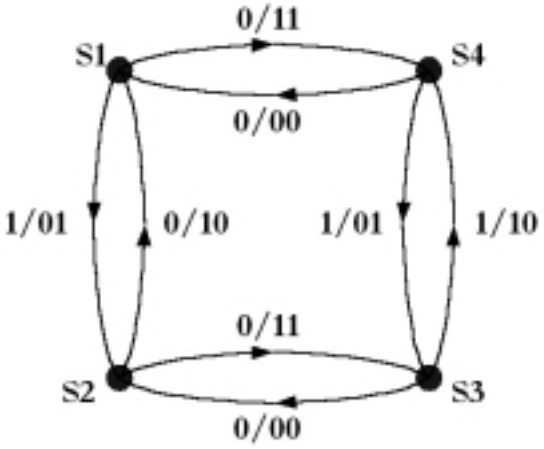

Fig. 3. State diagram of MB12

straightforward. Two middle bits in each codeword correspond to data bits. MB code is also known as WACX [2].

MB34 has four states, implying a yet fairly low implementation complexity. As seen from the figure, both DSV and ASV are four. At each state, there are exactly nine ($= 2^3$) transition paths. Four of them are self-returning. Each of the two neighboring states terminate two paths. The farthest state terminates the last path. It is left to the reader to construct a codebook and the associated state diagram.

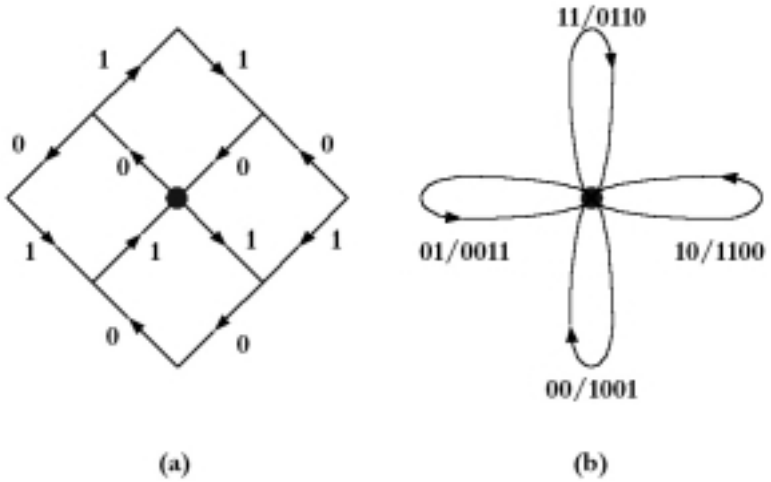

Fig. 4. MB24. (a)BUDA stack (b)State diagram

One notable aspect of all the codes introduced so far, MB12, MB24, and MB34, is that all the possible transition paths are fully exploited to construct a code. They are, in a sense, complete codes.

For some other codes, however, there may be some valid transition paths which are not utilized in code mappings. These extra transition paths can be utilized in securing control symbols often required in practical physical layer designs.

The power spectral densities of MB12, MB24, and MB34 can be sketched like in Fig. 6. MB12 and MB 24 are in fact equivalent codes in all practical purposes. They have the same power spectral envelopes, the same DSV and ASV values, etc. In comparison with NRZ, they have the same main lobe width but offers extra DC-free and RLL property. MB34 is a further improvement over the two in terms of the main lobe bandwidth with but a small added codec complexity.

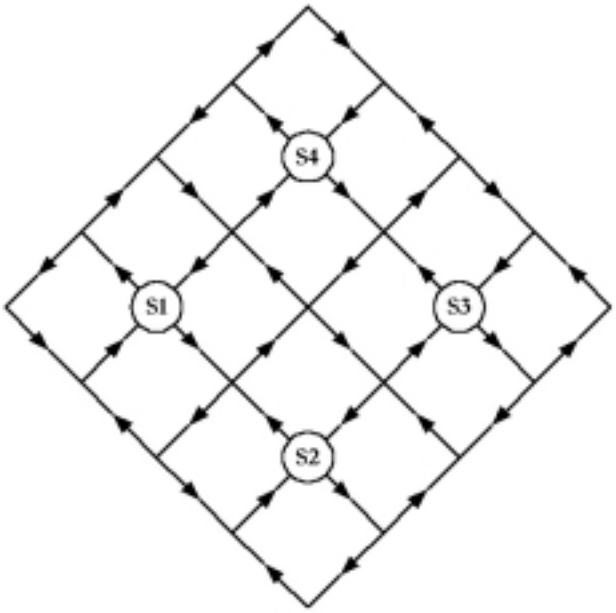

Fig. 5. MB34 BUDA stack

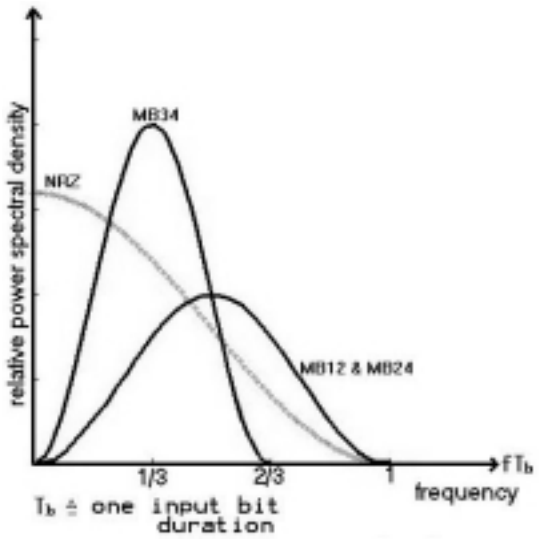

Fig. 6. Power spectral sketches for some simple binary codes

V. Code Design Procedural Steps

The design steps of DC-free binary MB line codes can be summarized as follows:

1. Select the number of input bits m and the number of output bits n for an (m, n) block code. It turns out that n should be an even number; a binary MB code of an odd value of n is found to be impossible. m preferably is $n-1$ for a minimal redundancy. It may turn out that code design with such a redundancy is not possible. In that case, m equal to $n-2$ will be taken.
2. Accumulate an enough number of BUDA cells to form a BUDA stack for derivation of the state diagram. Note that stacking cells horizontally would increase the DSV value. Doing so vertically would increase the ASV value. Therefore, where to add additionally needed cells should depend on the design preference on the two spectral properties.
3. Pick one node as a state. And try to secure at least 2^m exit paths, each being n-hop long. Pin down the terminating node of each path as another state.
4. Start with a new state and do the same as in the previous step. Try to terminate paths on already

existing state nodes and try not to generate new states as far as possible.
5. If it turns out that the stack needs to be expanded to complete the state diagram, add more cells to the stack either horizontally or vertically as appropriate.
6. Try to end up with as small a number of states as possible with all transition paths terminating on one of the arranged states. If this trial succeeds, then the stack design is complete.
7. At each state, arrange the mappings from input m-bit combinations to output n-bit combinations. The result is the state transition table or codebook. Try to use the same mapping as far as possible across many or all states to reduce implementation complexity. This will also tend to simplify the decoding rule.
8. If necessary, plot the power spectra of the coded sequence with varying input Mark probability, and choose the mapping that would generate most desirable power spectral properties. Some of the criteria for desirable power spectral properties might be the smoothness of the spectral envelope with little spectral spikes and the symmetry of the spectral envelope across the frequency band of interest.
9. If all these steps are done successfully, then the code design itself is done with success.

VI. MB810

We designed a MB810 code following the steps summarized in the previous section. The resultant BUDA stack diagram is shown in Fig. 7. We ended up with 12 states. Both DSV and ASV are 7. There are at least $256(= 2^8)$ exit paths out of each state. Each such path is 10 hops long and terminates on one of the other states. In studying the BUDA stack, it is to be recalled that each right-arrowed branch represents an output logic symbol '1'(physical symbol +1/2) while each left-arrowed one represents an output logic symbol '0'physical symbol -1/2.

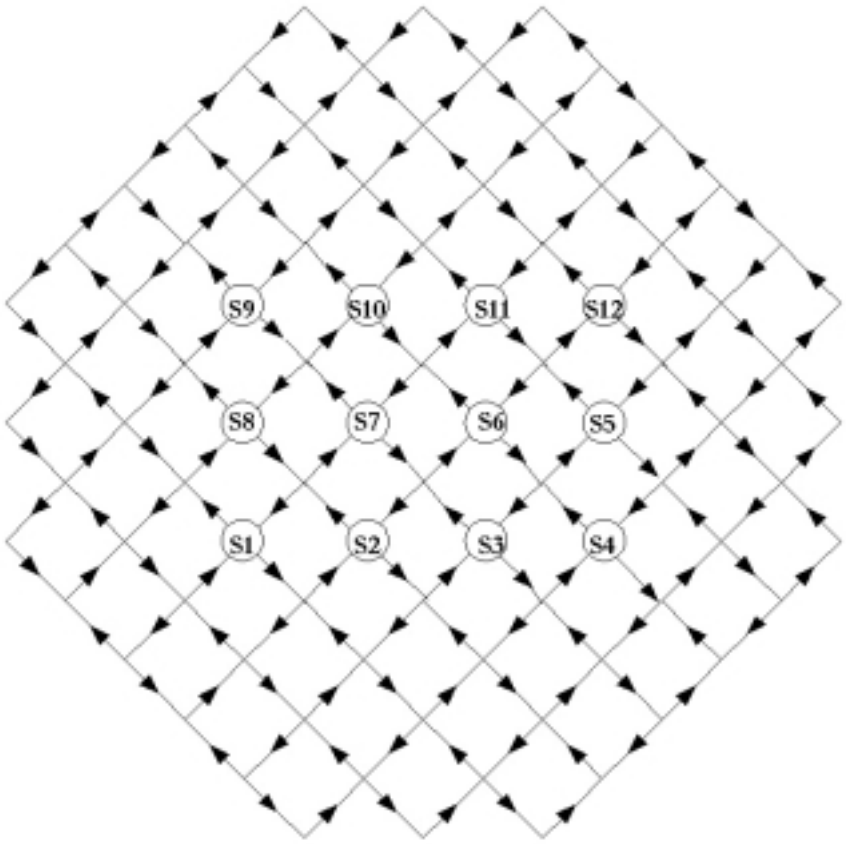

Fig. 7. BUDA stack of MB810

The state diagram is too complex to draw, but a collapsed version is depicted in Fig. 8. State

transitions are mostly limited to adjacent neighboring states. This is but our design choice. By doing so, more mapping combinations may apply in the same way to more states, resulting in simpler coder and decoder logics. It is also to be noted that not all transitions are among neighboring states. Although not shown in the simplified state diagram provided, a few of the transition paths are among states farther apart.

There are more than 256 paths out of each state. Therefore, it is easy to provide for control symbols by use of these extra paths.

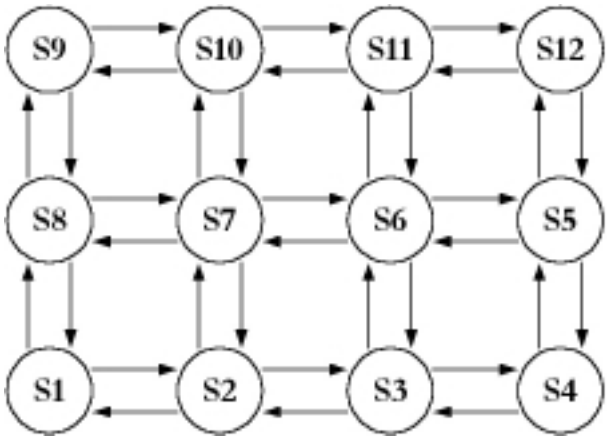

Fig. 8. Condensed state diagram of MB810

VII. Performance

The measured power spectrum of MB810 is shown in Fig. 9. Fig. 9(a) is the spectrum with square pulse shaping. The line rate was set to 10Gbps, the input data rate then being 8Gbps[1]. A spectral null at 5GHz, the Nyquist frequency, is apparent. There exists also a null at 0Hz, which is not shown in the figure. The spike at 10GHz is due to imperfect square pulse shaping, i.e., asymmetric duty cycle.

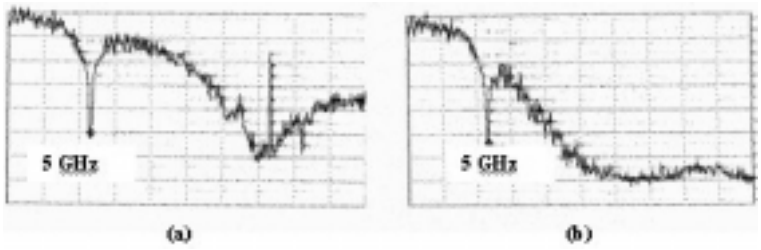

Fig. 9. MB810 power spectrum (a) Unfiltered (b) Filtered by a 5-th order Chebyshev LPF of a cutoff at the Nyquist frequency.

One might be curious to know how MB codes will behave in terms of performance if they are filtered severely above the Nyquist frequency. If they should operate successfully under this condition, it would mean not only spectral saving but also reduction of many undesirable anomalies due to spectral interference encountered in very high-speed DWDM optical transmission.

We passed the MB810 coded stream through a 5-th order Chebyshev LPF with a cutoff at 5GHz, the result of which is shown in Fig. 9(b). A better way to check the consequence of such a band limiting may be to observe the resultant eye diagram shown in Fig. 10. The left one is the eye pattern at the outlet of the transmitter whereas the right one is that observed after filtering. It is seen that the eye pattern of the filtered stream is fairly good, suggesting a stable and near error-free reception.

[1]The power spectrum analyzer available to us could measure only up to 12GHz. With a 10Gbps input data rate and so a line rate of 12.5GHz, a wider spectral capability would be required

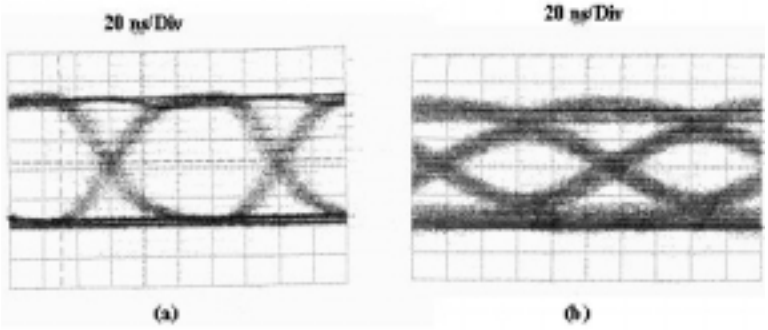

Fig. 10. Eye pattern of MB810-coded signal (a) Transmit output. (b) Filtered by 5GHz LPF.

We did not perform extensive BER test in this experimental configuration yet, which would necessarily require a large amount of time, but a similar experiment performed with MB34 code as reported in our presentation at the July 99 meeting verifies that BER as low as 10E-12 can be achieved from SMF transmission over a typical long-haul optical span of 250km.

VIII. Implementation

A straightforward way of implementing a block code is to do table look up, which would require memory access. However, a simple trick also enables implementation with only some mixture of combinatorial and sequential logics.

More difficulty in implementing a MB810 codec for 10GbE is the logic speed. It is practically impossible to implement the logic directly in the 10GHz range. A novel way of implementing the codec is to do the basic logic at a hundred times lower speed and then to execute double 10:1 multiplexing.

Fig. 11 depicts the idea we have adopted in implementing a MB810 coder. Ten coders work in parallel. Each coder is fed a byte. Let us denote each byte in sequence by a, b, ..., j and each output 10-bit word by A, B, ..., J. In order to work in parallel, the state transition information propagates from one coder to another in a sequential way and that at the earliest possible timing. The input byte clock is 125MHz. Hence the aggregate input bit rate is 10Gbps(= 125M x 8) and the aggregate output rate is 12.5Gbps(= 125M x 10).

Outputs of ten coders are bit-wise multiplexed by ten parallel MUX's; the first MUX collects the first bit of each codeword, the second MUX collects the second bit of each codeword, and so forth. In other words, the first MUX outputs A0, B0, ..., J0 in series, the second MUX does A1, B1, C1, ..., J1, etc. The output clock rate of these first-stage MUX's is 1.25GHz, ten time faster than the basic coders.

Then the second-stage MUX bit-wise multiplexes outputs of the ten first-stage MUX's. The resultant output bit sequence runs as A0, A1, ..., A9, B0, B1, ..., B9, ..., J0, J1, ..., J9 and so forth. Interpreting the output by codeword, this corresponds to ABCDEFGHIJ..., which is the correct order of the output codewords as implied by the front-end basic coders configured in Fig 11.

The output clock rate of the second MUX is 12.5GHz, resulting in the 12.5Gbps line rate as designed.

IX. Conclusion

This White Paper presented facts about MB810, from theoretical background to implementation. Experimental results, though not thoroughly extensive and complete yet, strongly support the theoretical assertions, i.e, significant improvement in bandwidth saving over other non-MB line codes.

The real message of this White Paper is that, in achieving the right speed of 10GbE, we don't have to abandon binary block coding which was successfully used in GbE. Binary block coding has many advantages over others; better SNR, self-clocking due to RLL, error monitoring capability, availability of extra control symbols, etc. Persistent use of binary block coding is possible by MB block coding as exemplified by the proposed MB810.

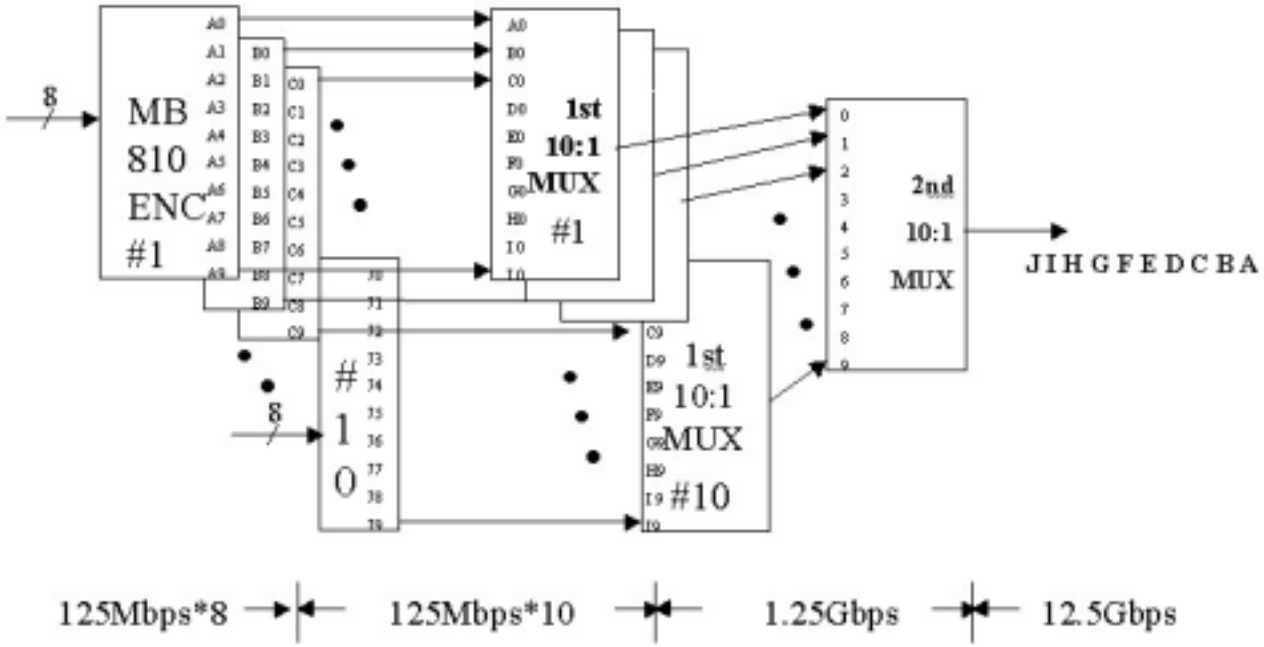

Fig. 11. Implementation schematic of the MB810 coder

You don't have to switch to NRZ. MB810 is even more bandwidth efficient than NRZ. And that, NRZ has many disadvantages. It is not DC-free. It is not RLL; scrambling never gives a tight upper bound to RLL and is just a statistical improvement, merely a naive resort to probabilistic luck.

You don't have to run the risk of multilevel signaling, like PAM5, over fairly nonlinear optical transmission systems. The bandwidth efficiency of MB810 is even comparable to PAM5. PAM5 has to suffer many shortcomings. It suffers from reduced SNR. It has to painfully combat against the inherent nonlinearity of optical devices. Successful multilevel optical signaling with sufficient reception SNR, and that as many as five levels at the Gbps speed, has never been reported. This is too much a risk at the current technology.

Stay with binary block coding and do away with the bandwidth concern by adopting a MB code like MB810.

References

[1] D. Y. Kim and Jae kyoon Kim. A condition for stable minimum-bandwidth line codes. *IEEE Trans. Commun.*, COM-33(2):152–157, February 1985.

[2] M. Y. Levy and S. R. Surie. A new coding technique for digital subscriber lines: Wacx. *Rec. ISSLS*, pages 247–251, 1984.