BEE: Predicting Realistic Worst Case and Stochastic Eye Diagrams by Accounting for Correlated Bitstreams and Coding Strategies

Aadithya V. Karthik[1,3], Sayak Ray[2,3], and Jaijeet Roychowdhury[1]
[1]The Department of Electrical Engineering and Computer Sciences, The University of California, Berkeley, CA, USA
[2]The Department of Electrical Engineering, Princeton University, NJ, USA
[3]Corresponding authors. Emails: aadithya@berkeley.edu, sayakr@princeton.edu

***Abstract*—Modern high-speed links and I/O subsystems often employ sophisticated coding strategies to boost error resilience and achieve multi-Gb/s throughput. The end-to-end analysis of such systems, which involves accurate prediction of worst-case and stochastic eye diagrams, is a challenging problem. Existing techniques such as Peak Distortion Analysis (PDA) typically predict *overly pessimistic* eye diagrams because they do not take into account the coding strategies employed. Monte-Carlo methods, on the other hand, often predict *overly optimistic* eye diagrams, and they are also very time-consuming. As an alternative, we present BEE, an accurate and efficient computational technique that applies dynamic programming algorithms to predict realistic worst-case and stochastic eye diagrams in modern high-speed links and I/O subsystems – with neither excessive pessimism nor undue optimism. BEE is able to fully and correctly take into account many features underlying modern communications systems, including arbitrary high-level transmit-side coding schemes and strategies, as well as various low-level non-idealities introduced by the underlying channel(s), such as inter-symbol interference (ISI) and crosstalk, asymmetric rise/fall times, jitter, parameter variability, *etc.* Furthermore, BEE accurately captures the fact that different received bits typically have widely different eye diagrams when a channel is driven by correlated bitstreams generated by coding strategies. We demonstrate BEE on links involving (7,4)-Hamming and 8b/10b SERDES encoders, featuring channels that give rise to multiple reflections, dispersion, loss, and overshoot/undershoot. BEE successfully predicts actual worst case eye openings in all these real-world systems, which can be twice as large as the eye openings predicted by overly pessimistic methods like PDA. Also, BEE can be an order of magnitude faster (and much more reliable) than Monte-Carlo based eye estimation methods.**

I. Introduction

High-speed signaling/communications links are becoming increasingly prominent in today's cutting-edge mobile chipsets, SoCs, and other high-performance hardware – including supercomputers, network switches used in data warehouses, high-frequency trading systems, and so on. Such links now play a significant role in determining key system-level performance metrics such as speed, power consumption, and reliability [1]–[4]. Over the past decade, the throughput of high-speed links has improved dramatically, from a few Mb/s to multi-Gb/s. This improvement is due to sophisticated error-resilient communications techniques that rely on transmit pre-emphasis/de-emphasis, new coding and modulation schemes, advanced equalization methods to compensate for inter-symbol interference (ISI), improved PLLs, DLLs, and clock and data recovery (CDR) solutions, advances in transceiver/link design with improved noise/jitter mitigation, *etc.* [1]–[6].

However, the additional complexity accompanying the above advances has made system design and optimization vastly more challenging, and in many situations, existing computational solutions/design tools are no longer adequate [1]–[3]. For example, a key task is to predict the *eye opening* of a high-speed link [1], [2]; this has important implications for noise margins and end-to-end BER analysis, choice of encoder/decoder architecture, design-space exploration for the receiver's front-end, *etc.*

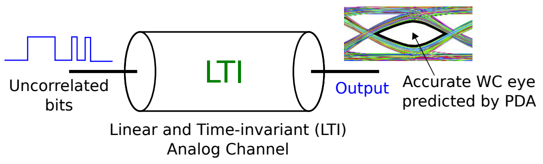
Uncorrelated bits
LTI
Output
Accurate WC eye predicted by PDA
Linear and Time-invariant (LTI) Analog Channel

Fig. 1. Worst case eye diagram computation using PDA.

As depicted in Fig. 1, worst-case eye diagrams are often estimated today using Peak Distortion Analysis (PDA) [1], [3], [7], which relies centrally on the assumption that all possible bit sequences are allowed across the channel. However, in modern high-speed links, it is often *the norm* to apply coding strategies [2] to the bits being transmitted, *e.g.*, 8b/10b encoding, error correcting codes such as the Hamming codes, *etc.* Such coding schemes are designed to improve error resilience by incorporating redundancy in the transmit bits. This significantly restricts the bit sequences allowed across the channel, and as a result, the transmitted bits become tightly correlated. This makes eye prediction by PDA overly pessimistic, which can in turn lead to link over-design, increased design/simulation/debugging costs, sub-optimal link operation, *etc.* PDA also misses subtler effects, such as the fact that different received bits can have widely different eye openings in the presence of coding strategies (see §II-C).

Due to the above limitations of PDA, high-speed link engineers and communications system architects typically resort to Monte-Carlo simulation to estimate eye openings. However, exhaustive enumeration of all relevant bit sequences is often computationally infeasible, so only a random subset of bit sequences is used for Monte-Carlo simulation. This frequently misses important bit patterns and corner cases and often predicts *overly optimistic* worst case eyes, especially if the system is being designed for very low BERs (*e.g.*, 10^{-12}). Such overoptimism can lead to serious problems downstream, *e.g.*, increased system-level design/debug time, underperformance requiring expensive post-silicon fixes, *etc.*

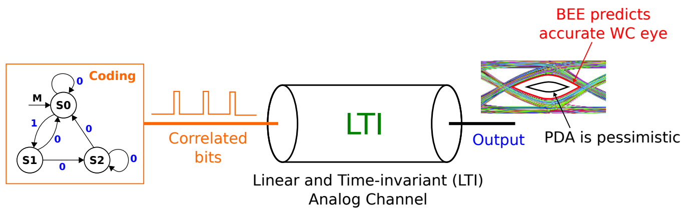
BEE predicts accurate WC eye
Coding
M
S0
S1
S2
1
0
0
0
0
0
Correlated bits
LTI
Output
PDA is pessimistic
Linear and Time-invariant (LTI) Analog Channel

Fig. 2. Worst case eye diagram computation using BEE.

In this paper, we present BEE, a technique that can efficiently find worst-case and probabilistic eyes of high-speed links with restricted or correlated bitstreams due to coding or other mechanisms. As Fig. 2 shows, BEE takes a description of the restriction/coding mechanism in the most general form, *i.e.*, a finite state machine (FSM) [8], as well as a description of the (linear) communications channel. It then analyses them together to find the *precise worst case eye*, with neither pessimism nor optimism. Typically, this eye opening is significantly bigger than the pessimistic eye predicted by PDA. If the underlying system is probabilistic (*e.g.*, due to jitter, parameter variability, stochasticity in the coding mechanism *etc.*), BEE can also predict important properties of the *probability distribution of the eye*, *e.g.*, mean, variance, and higher-order moments.

The core idea behind BEE is to label each node in the underlying FSM with a realistic worst-case scenario, *i.e.*, a sequence of bits terminating at the given node with the highest cumulative cost due to ISI, jitter, *etc.* At each iteration, the bit sequence associated with each node is expanded by one bit, and this bit is calculated from previously computed worst case sequences via an efficient dynamic programming [9] method. The iterations are continued until the lengths of the computed bit sequences exceed the memory of the underlying channel. At this point, the computed bit sequences are provably the worst-case scenarios for the link, and the corresponding worst-case eye openings are returned. These eye openings are actual worst cases; they involve neither excessive pessimism nor undue optimism, unlike PDA or Monte-Carlo based approaches. Further, BEE's time complexity is linear in the size of the given FSM.

BEE provides link design capabilities that are far ahead of any existing technique that we are aware of. It can quickly identify the exact shape of the worse case eye, a capability that fits well with corner-based design methodologies. Also, BEE can identify regions of the eye diagram that correspond to a given probability of occurrence, depicting them graphically to provide immediate visual intuition. Further, BEE correctly captures the fact that different bits of a correlated bitstream can have widely different eye openings, a feature needed for accurate BER estimates. Also, BEE is orders of magnitude faster (and far more reliable in predicting actual worst-case scenarios/probabilities) than Monte-Carlo simulation, making it well suited for practical industrial designs. BEE correctly accounts for ISI, crosstalk, reflections/loss, overshoot/undershoot, dispersion, jitter, and parameter variability – all of which need to be taken into account to make a high-speed link robust in manufacturing practice. BEE's modelling framework is general and widely applicable: virtually any encoder or error correcting scheme that can be implemented digitally (whether using combinational or sequential logic) can be expressed as an FSM, while the linear channel can be specified using SPICE netlists, differential equations, Fourier/Laplace domain transfer functions, measured data, S-parameters, *etc.* We expect that replacing PDA's pessimistic eyes with BEE's accurate ones during cutting-edge industrial link design will have immediate wide-ranging impact; for example, the adoption of simpler, lower-power encoders with no loss of performance. We plan to release BEE under an open source license.

In this paper, we demonstrate BEE on two different types of links. The first (§III-A) employs a well-known error correcting code, the $(7,4)$-Hamming code, in the FSM; the channel is modelled as an RLGC chain (a commonly used model for an I/O link/interconnect). The second system (§III-B) employs an 8b/10b SERDES encoder – a very effective and widely used encoding method employed in several modern standards, including Gigabit Ethernet, IEEE 1394b, PCI Express, SATA and SAS, USB 3.0, *etc.* [10]. The channel here is a weighted sum of smoothed delays, representing a network of heterogenous transmission lines with reflections, or a wireless channel with multiple obstructions and reflectors. In both these cases, we demonstrate that PDA makes overly pessimistic worse-case eye predictions, whereas BEE is accurate and exact (as verified against Monte-Carlo). Indeed, the correct eye opening predicted by BEE can sometimes be twice as large as the pessimistic opening predicted by PDA, translating to an exponentially lower BER. We also demonstrate how BEE accurately takes into account clock jitter and parameter variability in the channel.

The rest of this paper is organized as follows. In §II, we first cover key concepts such as the standard PDA algorithm, the use of FSMs to represent bit correlations, *etc.*, and then we present the core algorithms and techniques underlying BEE. We demonstrate our results (on the two systems described above) in §III, and we conclude in §IV.

II. CORE TECHNIQUE: NEW EYE ESTIMATION ALGORITHMS UNDERLYING BEE

In this section, we first describe some key concepts such as how PDA works, how FSMs can be used to model correlated bits, *etc.* Then we present the core techniques and algorithms underlying BEE.

A. Peak Distortion Analysis for LTI channels driven by uncorrelated bits

As we mentioned in §I, PDA is a technique for determining WC eye diagrams at the output of a Linear and Time-invariant (LTI) channel, when the channel input is a sequence of uncorrelated bits (*i.e.*, no coding strategy is used). We now describe how PDA works.

Given an LTI channel, let $h(t,T)$ denote the channel's response to the pulse $u(t,T)$ (where t denotes time and T is the pulse width) given by:

$$u(t,T) = \begin{cases} 1 & \text{if } t \in (0,\ T] \\ 0 & \text{else} \end{cases}$$

Suppose we apply a sequence of bits $\{b_i\}_{i=-\infty}^{+\infty}$ to the above system as input, with each bit being active for a period T. The corresponding input waveform $x_b(t,T)$ is given by:

$$x_b(t,T) = \sum_{k=-\infty}^{+\infty} b_k\ u(t-kT,T)$$

The channel's response $y_b(t,T)$ to the above input is given by:

$$y_b(t,T) = \sum_{k=-\infty}^{+\infty} b_k\ h(t-kT,T)$$

The key idea behind PDA is that we can formulate the worst case (WC) eye computation at time Δ as a pair of optimization problems on a truncated version of the output $y_b(t,T)$:

$$\begin{aligned} \text{WC0}\ (\Delta,\ T) \quad &= \quad \max_{\{b_i\}} \sum_{k=-M}^{\lfloor \Delta/T \rfloor} b_k\ h(\Delta-kT,\ T) \\ &\text{subj. to} \quad b_0 = 0, \text{ and} \end{aligned} \tag{1}$$

$$\begin{aligned} \text{WC1}\ (\Delta,\ T) \quad &= \quad \min_{\{b_i\}} \sum_{k=-M}^{\lfloor \Delta/T \rfloor} b_k\ h(\Delta-kT,\ T) \\ &\text{subj. to} \quad b_0 = 1 \end{aligned} \tag{2}$$

Here, M is a suitably chosen large integer that exceeds the memory of the channel. The crux of PDA is that the solutions to the above optimization problems are straightforward. The intuition is that, simply depending on the sign of the $h(.)$ term, we decide whether we want the corresponding bit to be a 0 or a 1. For example, if we are trying to maximize the output (*i.e.*, the WC0 problem), and we find that $h(\Delta-kT,T)$ is positive for some k, we choose b_k to be 1, and vice versa. In this way, we obtain two optimal solutions, WC0 and WC1, for each Δ. By iterating over Δ, these solutions trace the WC0 (lower half) and WC1 (upper half) of the required worst case eye opening.

The above, of course, was possible only because the bits $\{b_k\}$ were all uncorrelated, and could be set to 0 or 1 independently of one another. On the other hand, if the bits $\{b_k\}$ are correlated (*e.g.*, due to a coding scheme), the entire method breaks down and more powerful techniques (discussed below) are needed.

B. FSMs as a way to model fully-general coding schemes

Our chief concern in this paper is the computation of eye openings in the presence of coding strategies/bit correlations. Therefore, we need a general way to specify these coding schemes/bit correlations. In this context, FSMs are powerful data structures that enable us to represent virtually any kind of bit correlation/coding scheme in a fast, convenient, and fully general way.

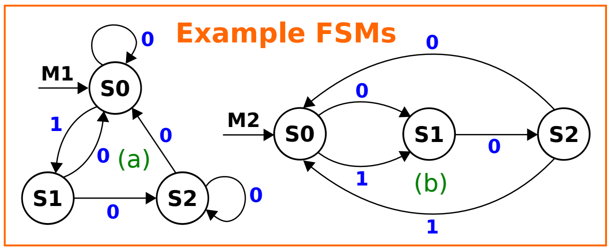

Fig. 3. Example FSMs representing correlated bits/coding strategies: (a) an FSM that transmits no two consecutive 1s, and (b) an FSM that transmits a 0 every third bit.

Briefly, an FSM is a data structure that consists of finitely many *discrete states* (*e.g.*, see Fig. 3). Each FSM represents an automaton, *i.e.*, a machine that behaves according to a pre-defined logic. At the beginning, the FSM is in a *start state*. For instance, the FSMs of Fig. 3 each have a start state labelled $S0$. FSMs also have arcs (or edges) between states annotated with *output bits*, as Fig. 3 shows. The FSM operates in discrete time (*e.g.*, at every uptick of a periodic clock signal). At each time point, the FSM *transitions* from its current state (along one of the arcs directed outward from its current state), reaching a new state (wherever the arc leads to). During this time, the machine's *output* is the bit along the arc that was just followed. For example, in Fig. 3 (a), if the FSM is in state $S0$, it could transition to either $S1$ (producing a 1 as output), or it could remain in $S0$ (producing a 0).

It is easy to see that the bit sequences produced by FSMs are *correlated*. Each bit cannot be set independently of the others. For example, the FSM of Fig. 3 (a) can never produce two consecutive 1s at the output. The FSM of Fig. 3 (b) produces a 0 every third bit.

Also, it can be shown that virtually any digitally implementable system, including complex communications protocols, encoders and decoders, error-correcting codes, *etc.*, can be represented as an FSM. This enables us to use FSMs as a general way to specify arbitrary coding schemes/bit correlations for eye diagram analysis.

C. Coding schemes: different eye openings for different bits

When a coding strategy is employed as part of a communications scheme, different received bits can have widely different eye diagrams.

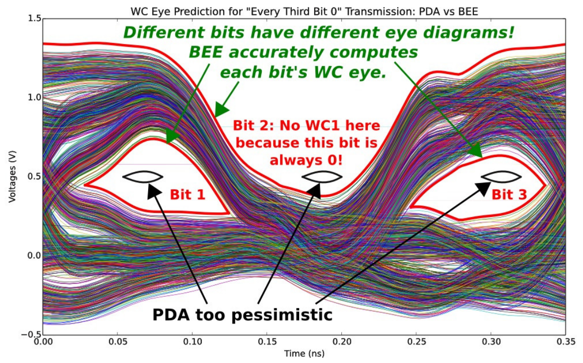

Fig. 4. An example to show that a transmit-side coding scheme can have a profound impact on receive-side eye diagrams: different received bits can have widely different eye diagrams. In this example, the encoder transmits a 0 every third bit (Bit 2, Bit 5, Bit 8, *etc.*), which significantly improves the eye diagrams for all bits at the receiver. PDA does not recognize this and is too pessimistic, whereas BEE accurately accounts for each individual bit, producing worst case eye diagrams that are consistent with Monte-Carlo simulations. This is true even for the bits that are always 0, where the question of a worst case 1 does not arise.

For example, consider the coding scheme represented by the FSM of Fig. 3 (b). This scheme produces a 0 at every third bit. Suppose this FSM feeds into an LTI channel (*e.g.*, an RLGC chain). The eye openings for the received bits at the end of the channel will clearly be very different from one another. For example, every third bit (constrained to be a 0), will have only a one-sided eye opening (as there is no WC1 to consider). Similarly, the bit immediately preceding the constrained-0 bit will have a completely different eye opening from the bit immediately following the constrained-0 bit. Fig. 4 depicts the eye diagrams for these bits, along with the eye openings predicted by PDA (black) and BEE (red).

PDA only accounts for the channel, not the coding scheme. Therefore, it predicts the same pessimistic eye for all bits, as Fig. 4 depicts. BEE, on the other hand, fully accounts for the underlying coding strategy as well as the channel, and therefore accurately predicts the worst-case eye opening for each received bit (including the one-sided "eye" for the constrained-0 bit, as shown in Fig. 4).

D. BEE: Worst case eye prediction for correlated bits/coding schemes

Following §II-B, we model the underlying coding scheme as an FSM, which produces a correlated bitstream that feeds into an LTI channel (Fig. 2). We now describe BEE's algorithm for computing WC eye openings for such systems.

The problem statement is almost identical to the PDA optimization problems (1) and (2). The crucial difference, however, is that unlike PDA, the bits $\{b_k\}$ cannot be independently chosen as 0 or 1; the bits have to correspond to valid output sequences that can be produced by the underlying FSM.

As mentioned in §I, the core idea behind BEE is to label each node in the FSM with a realistic worst-case scenario, *i.e.*, a sequence of bits terminating at the given node with the highest cumulative cost. This is best illustrated by an example. Let us imagine that the FSM in question is the one shown in Fig. 3 (a). Further, let us assume that we are solving the WC0 optimization problem, and that we would like to maximize $5b_0 + 3b_1 + 2b_2$, where bit b_i is the output of the FSM at time i (assume that we begin at time 0 at state S0). If we were executing PDA, the solution would be simple; since all the multiplying coefficients are positive, we choose all the 3 bits (b_0, b_1, and b_2) to be 1. However, this solution is overly pessimistic, because we know from §II-B that our FSM can never produce 2 consecutive 1s.

As §I mentions, BEE solves the above problem via *dynamic programming*. In the above example, BEE starts with a simpler problem: what is the maximum value of $5b_0$ that leaves us in each state (S0, S1, and S2) at time 1? Clearly, the answer is 0 for state S0 (corresponding to the bit sequence 0), 5 for state S1 (corresponding to the sequence 1), and we cannot ever be in state $S2$ at time 1. At this stage, therefore, BEE tags the states S0, S1, and S2 with the optimal bit sequences 0, 1, and `UNDEFINED`, and the costs 0, 5, and `UNDEFINED` respectively.

Now BEE solves a slightly harder problem: what is the maximum value of $5b_0 + 3b_1$ terminating in each state at time 2? Clearly, to arrive at S0 at time 2, we must either reach S0 at time 1 and stay there, or reach S1 at time 1 and then transfer to S0. The first option yields a maximum cumulative cost of 0, while the second yields a cost of 5 (a cost of 5 to reach S1 at time 1, known from the previously calculated optimal solution at S1, plus a cost of 0 going from S1 to S0). So BEE chooses the second option, and tags S0 with the sequence 10 and the cost 5. Note how BEE used the solution to the previous problem to build up a solution to the current one. This is the crux of BEE's dynamic programming algorithm: reusing solutions to previously solved simpler problems to gradually compute the entire WC eye. Thus, in this example, at the end of the second iteration, the states S0, S1, and S2 are tagged with the sequences 10, 01, and 10, and the costs 5, 3, and 5 respectively.

More generally, at each iteration, the bit sequence associated with each node is expanded by one bit (or set to `UNDEFINED`), and this bit is calculated from previously computed worst case costs. The iterations are continued until the required optimization problem is solved, which occurs when the lengths of the associated bit sequences exceed the memory of the underlying channel. At this point, the computed bit sequences are provably the worst-case scenarios for the link, and the corresponding worst-case eye openings are returned.

For instance, in the above example, at the end of the final iteration, the states S0, S1, and S2 are tagged with the sequences 100, 101, and 100, and the costs 5, 7, and 5 respectively. Picking out the maximum, the worst case cost is 7, corresponding to the worst case bit sequence 101. By contrast, if we had used PDA, the worst case cost returned would have been a pessimistic 10, corresponding to the (impossible) bit sequence 111.

Thus, not only is BEE's WC prediction *provably exact*, but it can also generate a certificate specifying the exact sequence of correlated bits corresponding to the WC outcome. This is a very useful property; in addition to providing an independently verifiable mathematical guarantee, it also serves to benchmark other commonly used heuristics/algorithms for WC eye estimation.

We note that our implementation of BEE includes mechanisms for handling asymmetric rise/fall times, jitter, stochasticity and parameter variability, *etc.* Each of these is accounted for by adding extra functionality to the basic algorithm described above. However, due to space constraints, we are unable to describe these added functions here. However, our results below illustrate these additional capabilities of BEE using real-world examples and test-cases.

III. Results

We now apply the algorithms and techniques discussed above to perform worst case and stochastic analysis of eye diagrams in systems that arise in Signal Integrity, I/O and high-speed communications applications.

A. *Worst case eye analysis of a* (7,4)*-Hamming encoded system*

We now apply BEE to carry out WC eye diagram analysis of a (7,4)-Hamming encoded communications scheme (a commonly used parity-based error correcting code), over an LTI channel composed of a chain of RLGC units (a lumped transmission line model) that captures inter-symbol interference, crosstalk, overshoot/undershoot, dispersion, *etc.*

The (7,4)-Hamming encoder accepts 4 data bits ($d_1 - d_4$) in parallel, and outputs a serial stream of 7 Hamming-encoded bits (p_1 through p_7). The relationship between the output bits $\vec{p}$ and the data bits $\vec{d}$ is given by:

$$\underbrace{\begin{pmatrix} p_1\\p_2\\p_3\\p_4\\p_5\\p_6\\p_7 \end{pmatrix}}_{\vec{p}} = \underbrace{\begin{pmatrix} 1&1&0&1\\1&0&1&1\\1&0&0&0\\0&1&1&1\\0&1&0&0\\0&0&1&0\\0&0&0&1 \end{pmatrix}}_{\mathbf{G}} \underbrace{\begin{pmatrix} d_1\\d_2\\d_3\\d_4 \end{pmatrix}}_{\vec{d}} \quad \text{(modulo 2)}$$

We now construct an FSM for the (7,4)-Hamming coding scheme above. To do so, we observe that not all 7-bit combinations are valid Hamming codewords. Indeed, although there are 128 possible combinations of 7 bits, only 16 of these represent valid codewords. These can be represented using a *prefix tree* like FSM structure, as shown in Fig. 5.

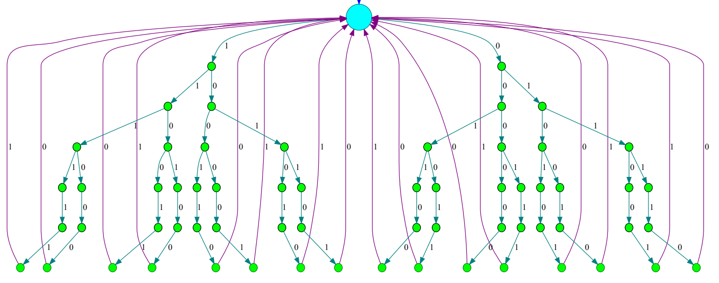

Fig. 5. The (7,4)-Hamming encoder FSM based on the 16 codeword prefix tree.

The root of the prefix tree denotes the start state of our FSM. Every proper prefix of each valid Hamming codeword corresponds to an internal node in the tree (or a state in the FSM). The 16 leaves of the tree (representing valid Hamming codewords) are looped back to the start state of the FSM, which resets the encoder every 7 bits, making it ready to produce the next 7-bit codeword.

The channel is an analog LTI system that consists of a chain of 30 RLGC units, as shown in Fig. 6 below.

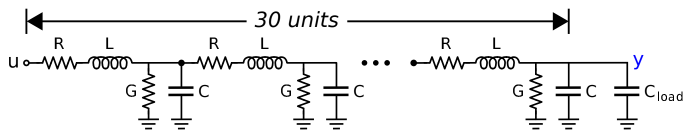

Fig. 6. A 30-unit RLGC chain used to model the analog channel following the (7,4)-Hamming encoder.

The pulse response of the above channel is shown in Fig. 7.

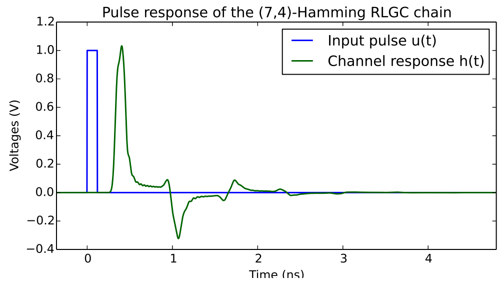

Fig. 7. Pulse response of the 30 unit RLGC chain analog channel following the (7,4)-Hamming encoder.

For the above "(7,4)-Hamming encoder + channel" system, we now determine the WC eyes at the receiver using BEE. We also validate these by carrying out a large number of Monte Carlo simulations. We also apply PDA to the above system (which pessimistically assumes that the bits are all uncorrelated), and compare its predictions against BEE's.

The results are shown in Fig. 8. Note that each of the 7 bits produced by the encoder (per codeword) gives rise to a differently shaped eye opening, as discussed in §II-C. This is a direct consequence of the correlated nature of the bits – which PDA is unable to account for. On the other hand, as seen from Fig. 8, BEE accurately reproduces the true shape of the WC eye for each received bit.

Further, in the experiment above, BEE was an order of magnitude faster than Monte-Carlo. Also, even though we simulated thousands of bits in our Monte Carlo runs, we were unable to generate the worst case. But once the worst case sequences were discovered by BEE, we were able to include them in our Monte Carlo runs and confirm that these sequences indeed led to the worst case outcomes predicted by BEE. This is true for many real-world systems: the number of Monte Carlo runs needed to reliably generate worst case outcomes is often completely impractical from a computational standpoint. In such situations, BEE can offer valuable guidance in directing the search for the worst case.

B. *Worst case eye analysis of an* 8*b*/10*b-SERDES system*

We now apply BEE to an 8*b*/10*b* SERDES encoder, followed by a behavioral channel macromodel (a cascade of tanh(.) smoothened delays).

In this system, the encoder accepts as input 8 data bits in parallel, and produces as output 10 serialized bits. The encoded bits have some very desirable correlation properties that often result in significantly improved WC eye diagrams at the receiver. For example, it is guaranteed that no more than 5 consecutive 0s or 1s will occur at any time in the bitstream. This "normalization" of bits has a very positive impact on worst case eye diagrams, a fact that is completely ignored by PDA.

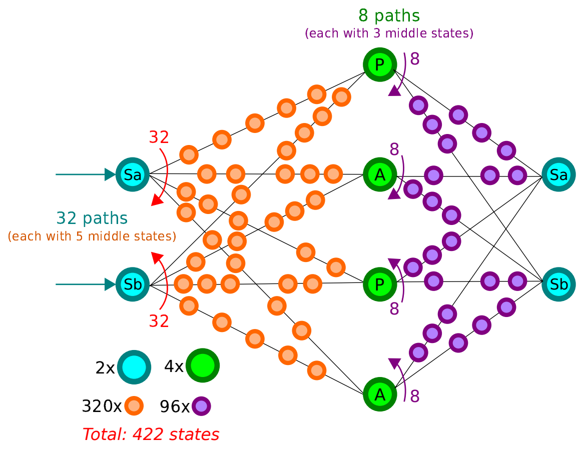

Fig. 9. An 8*b*/10*b* encoder FSM based on 5*b*/6*b* and 3*b*/4*b* codes.

The 8*b*/10*b* encoding is carried out in two stages, (i) a 5*b*/6*b* stage that encodes the first five bits into a 6-bit word, and (ii) a 3*b*/4*b* stage that encodes the last three bits into a 4-bit word. To avoid a streak of more than

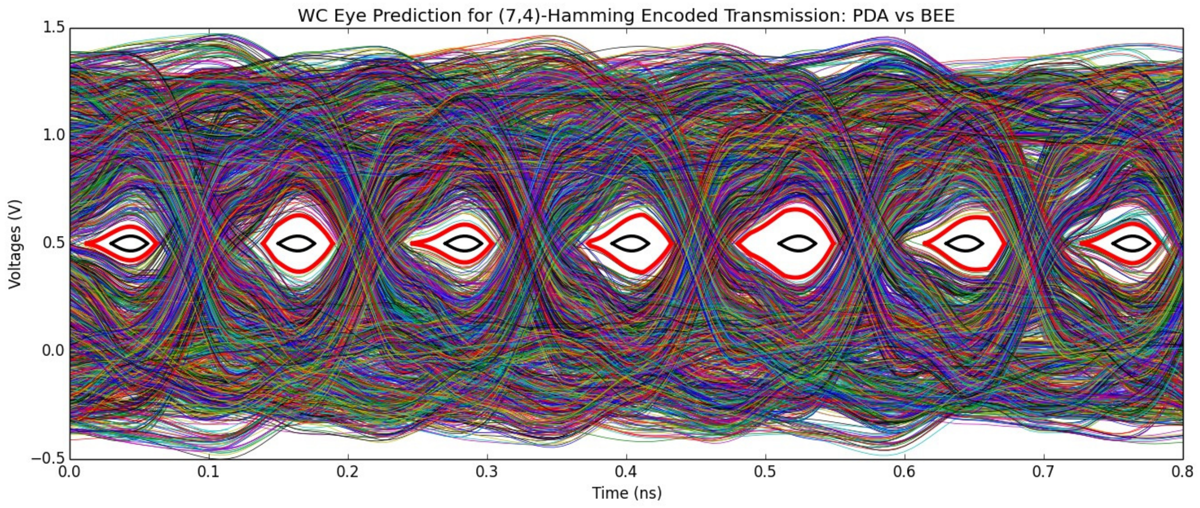

Fig. 8. Eye diagrams predicted by Monte Carlo simulation (various colors), by PDA (black), and by BEE (red) for the $(7,4)$-Hamming encoded communications scheme of §III-A. It is clearly seen that BEE is able to produce exact and accurate worst case eye diagrams for the given system, unlike the overly pessimistic eye diagrams predicted by PDA.

5 consecutive 0s or 1s, the encoder keeps track of a quantity called the *Running Disparity (RD)*, defined as the number of 1s minus the number of 0s in the bitstream produced thus far (for details, see [10]). The FSM model for the $8b/10b$ SERDES encoder is shown in Fig. 9. The FSM has two start states (labelled *Sa* and *Sb* in the figure), corresponding to RD being -1 and $+1$ respectively. As seen from the figure, the $5b/6b$ encoding stage is implemented using 64 FSM paths (32 emanating from *Sa* and 32 from *Sb*), each with 5 intermediate FSM states and a terminal state that is one of the 4 green states shown in the figure. This is followed by the $3b/4b$ encoding stage, which consists of 32 additional FSM paths (8 paths emanating from each of the 4 green FSM states), that eventually loop back to *Sa* or *Sb* after traversing 3 intermediate states each.

The pulse response $h(t)$ of the channel following the FSM is given by:

$$h(t) = \sum_{i=0}^{N-1} \alpha_i \left(\frac{1+\tanh(k(t-t_i^{(1)}))}{2} \right) \left(\frac{1-\tanh(k(t-t_i^{(2)}))}{2} \right)$$

Each term in the above summation represents a smooth "dead delay", with amplitude α_i active during the time interval $[t_i^{(1)}, t_i^{(2)}]$. The pulse response of the channel is shown in Fig. 10.

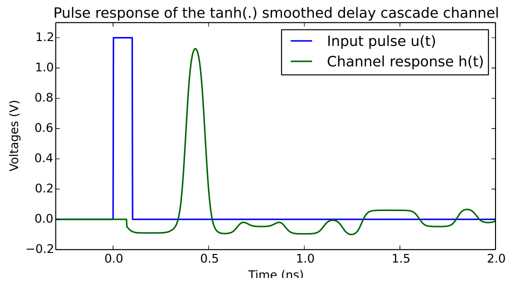

Fig. 10. Pulse response of the tanh(.) smoothed cascaded delay channel following the $8b/10b$ SERDES encoder.

Similar to our analysis of the $(7,4)$-Hamming encoder §III-A, we now use BEE to determine the WC eye diagram for the $8b/10b$ SERDES system above. And just like the previous case, we see from Fig. 11 that BEE is able to accurately reproduce the true shapes of the WC eye diagrams for the given system (for all the 10 bits), whereas the predictions made by PDA are overly pessimistic.

C. Jitter analysis and worst case eye diagrams

We now apply BEE to carry out jitter analysis of the systems above. Our model for jitter is a random time shift of the channel output. Thus, if we denote the output of the LTI channel by $y(t)$, then the jittered output is given by $y(t-J(t))$, where $J(t)$ is the (time-varying) jitter in the system. If $J(t)$ is *bounded*, *i.e.*, $J(t)$ always lies in a range (say, between $J_{\min}$ and $J_{\max}$), then BEE's worst case dynamic programming algorithm can be modified to compute the worst case eye in the presence of jitter. Thus, by setting different bounds on the jitter, we can use BEE to compute the *jitter tolerance* of each bit in the system, *i.e.*, the gracefulness of degradation of the eye associated with each bit as jitter is gradually increased. This is shown in Fig. 12, where we apply BEE to predict the worst case eye of each bit in our $(7,4)$-Hamming system, as jitter increases from 0 to $\pm 10\%$ of the bit period T in gradual steps. At jitter being 0, the worst case prediction simply reduces to the worst case prediction without jitter (the black contours in the figure). However, as jitter increases, the eye of each bit starts collapsing – into smaller and smaller concentric eyes as shown in the figure. We believe that this kind of detailed information regarding the tolerance of each bit to jitter can help designers come up with optimal pre-emphasis/de-emphasis strategies, placement of active buffers and boosters, more accurate BER analysis, *etc.*

D. Stochastic analysis of eye diagrams with parameter variability

We now introduce a source of randomness into our analysis, namely parameter variability.

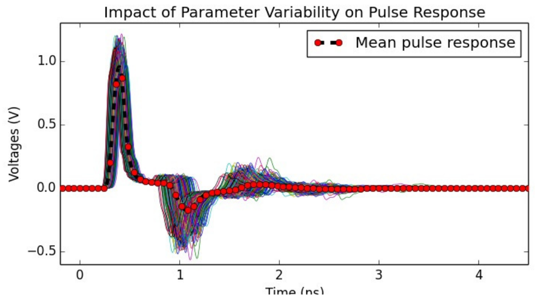

Fig. 13. Time-varying distribution of pulse response as a result of parameter variability.

To model parameter variability, we sampled the parameters R, L, G, and C of the $(7,4)$-Hamming system from independent Gaussian distributions with suitable mean and variance. For a large number of such samples, we computed the pulse responses $h(.)$ as functions of time, and plotted them on top of one another, yielding Fig. 13. From such data, we generated a statistical model for $h(.)$ and then applied BEE to carry out combined stochastic analysis of this model with a probabilistic FSM model.

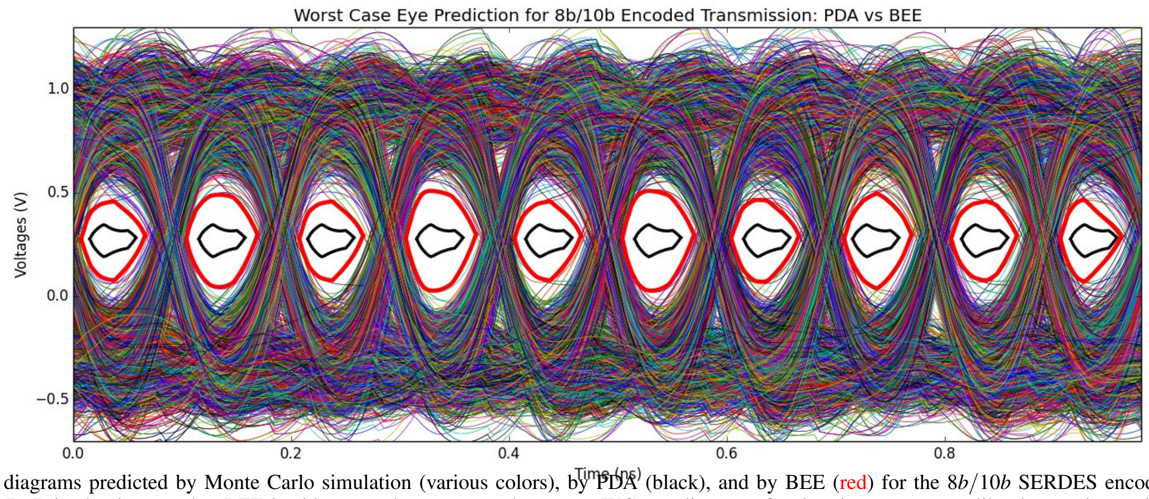

Fig. 11. Eye diagrams predicted by Monte Carlo simulation (various colors), by PDA (black), and by BEE (red) for the $8b/10b$ SERDES encoded communications scheme of §III-B. It is clearly seen that BEE is able to produce exact and accurate WC eye diagrams for the given system, unlike the overly pessimistic eye diagrams predicted by PDA.

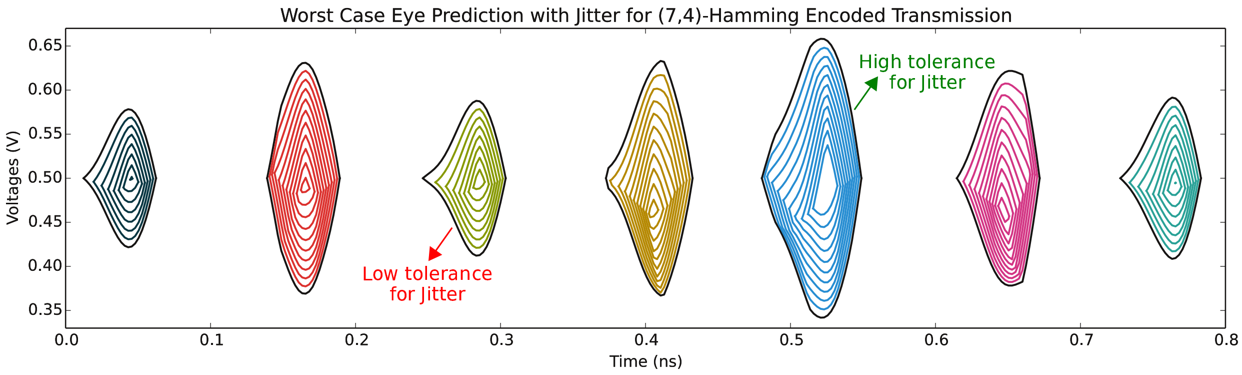

Fig. 12. Using BEE to predict the jitter tolerance/jitter margin of each bit in the (7,4)-Hamming system. The outermost eye diagram for each bit is the worst case eye in the absence of jitter. As jitter is gradually increased (from 0 to 10% of the period T), the eyes start shrinking until the opening completely vanishes.

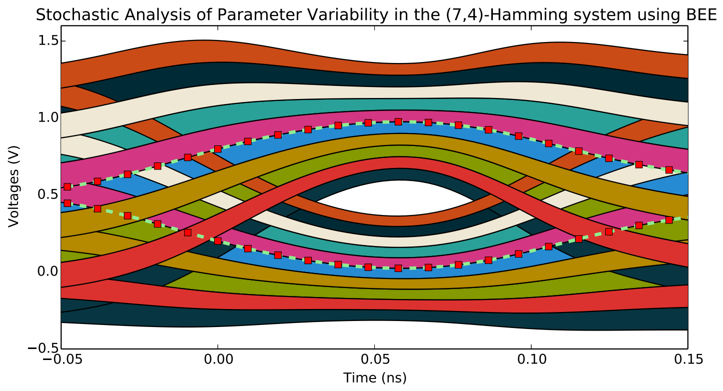

Fig. 14. Statistical characterization of a (7,4)-Hamming eye opening in the presence of parameter variability in R, L, G, and C of the underlying RLGC channel. The dashed line with the red markers is the time-varying mean of the channel output. The colored bands each have a thickness of 0.3σ, where σ is the time-varying standard deviation (square root of the variance) of the channel output.

The resulting statistical eye characterization is shown in Fig. 14 (please see the figure caption for more details).

IV. Summary, Conclusions, and Future Work

To summarise, we have developed and demonstrated BEE, a new efficient technique for accurately computing worst case and statistical eye diagrams in high-speed links and I/O subsystems, in the presence of arbitrary coding strategies, ISI, crosstalk, jitter, parameter variability, *etc.* To the best of our knowledge, no such comprehensive analysis technique existed prior to this work.

We have demonstrated BEE on some real-world designs (involving (7,4)-Hamming and $8b/10b$ SERDES coding schemes), showing that BEE is able to quickly and accurately predict actual worst case/stochastic eye openings for each bit in these systems, unlike PDA which often predicts overly pessimistic eye openings. In addition, BEE was an order of magnitude faster (and much more reliable in obtaining actual worst case eyes) than Monte-Carlo simulation.

In future, we would like to explore algorithmic refinements that will enable BEE to directly analyze error-correcting encoders, communications protocols, and other Boolean systems (*e.g.*, circuits expressed in Verilog, or as And Inverter Graphs or Binary Decision Diagrams), without having to first convert such systems into FSM form. Also, we would like to release BEE to the community as open source software.

References

[1] S. H. Hall and H. L. Heck. *Advanced signal integrity for high-speed digital designs*. John Wiley & Sons, 2011.

[2] G. Balamurugan, B. K. Casper, J. E. Jaussi, M. Mansuri, F. O'Mahony, and J. Kennedy. Modelling and analysis of high-speed I/O links. 32(2):237–247, 2009.

[3] B. K. Casper, M. Haycock, and R. Mooney. An accurate and efficient analysis method for multi-Gb/s chip-to-chip signaling schemes. pages 54–57, 2002.

[4] J. E. Jaussi, G. Balamurugan, D. R. Johnson, B. K. Casper, A. Martin, J. Kennedy, N. Shanbhag, and R. Mooney. 8-Gb/s source-synchronous I/O link with adaptive receiver equalization, offset cancellation, and clock de-skew. 40(1):80–88, 2005.

[5] P. K. Hanumolu, G. Y. Wei, and U. K. Moon. Equalizers for high-speed serial links. 15(2):429–458, 2005.

[6] J. A. Davis and J. D. Meindl. *Interconnect technology and design for gigascale integration*. Springer, Netherlands, 2003.

[7] http://download.intel.com/education/highered/signal/ELCT865/Class2_15_16_Peak_Distortion_Analysis.ppt.

[8] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to automata theory, languages, and computation, 2ed.* Addison-Wesley, 2001.

[9] C. E. Leiserson, R. L. Rivest, C. Stein, and T. H. Cormen. *Introduction to algorithms*. The MIT press, 2001.

[10] http://en.wikipedia.org/wiki/8b/10b_encoding.