8B9B Encoding for Crosstalk Reduction in a High-Speed Parallel Bus

Sunil Sudhakaran, Russell Newcomb
NVIDIA Corporation
Santa Clara, California 95050
ssudhakaran@nvidia.com

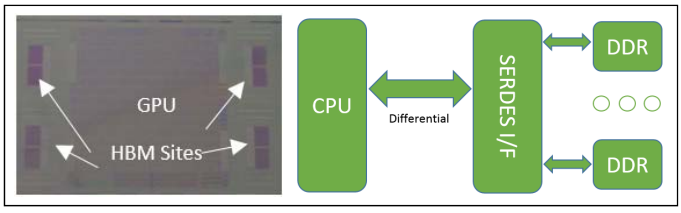
***Abstract*— This paper presents a new encoding and corresponding decoding scheme to reduce crosstalk on a high-speed parallel bus. The scheme is based on a modified Fibonacci sequence and is introduced along with potential benefits in some upcoming memory interfaces. The scheme provides appreciable eye opening for interfaces dominated by crosstalk such as existing memory interfaces.**

Keywords—Signal Integrity, Parallel bus, Memory interface, DDR, GDDR, Single-Ended, Crosstalk, Fibonacci Sequence, ISI

I. Introduction

High-speed I/O Interfaces often require a minimum signal-to-noise ratio (SNR) in order to ensure valid transmission of data. One of the dominant noise sources in high-speed links is crosstalk between signals. The majority of high-speed serial links in industry today use differential signaling including PCIe-Express, USB 3.1, and HDMI 2.0. In stark contrast, the majority of memory interfaces employ single-ended signaling. Single-ended interfaces are especially prone to common-mode noise sources such as power supply noise and crosstalk as they do not benefit from some inherent cancellation [1]. As a result, crosstalk becomes one of the dominant sources of noise in single-ended memory interfaces which eventually limits the maximum attainable data rate due to compromised signal-noise ratios.

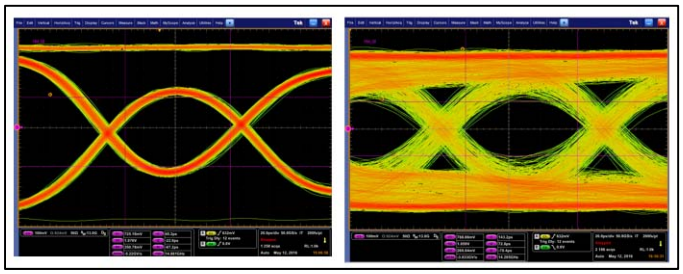
Memory bandwidth requirements for systems continue to grow which has created multiple directions for future memory. One direction currently pursued is going "wide" with an integrated memory using through silicon via's (TSV) known as high-bandwidth memory (HBM) as shown in Fig. 1(a) [2]. Current pricing precludes HBM from being adopted in the mainstream markets, so another direction being pursued is the adoption of a differential serial link for the base memory die as shown in Fig. 1(b). There are some issues with this approach, namely the incompatibility with current I/O design schemes prevalent in both mobile and graphics memory systems.



(a) (b)

Fig. 1: Future Memory Candidates (a) HBM (b) SERDES Front-end

The last commonly used option to extend bandwidth is to increase per-pin data rates. GDDR5x recently has done this, pushing per pin speeds of up to 10Gb/s [3]. To further increase data rates, the link budget and architecture needs to be studied in detail. Fig. 3(a) shows a measured GDDR5x eye diagram while 3(b) shows the same eye under ISI and crosstalk stress. The crosstalk impact is quite significant and will limit speeds if this last method of bandwidth extension is desirable.



3(a) 3(b)

Fig 3. Measured GDDR5x Eye Diagrams (a) Clock Pattern (b) Random Data with Crosstalk

Crosstalk can be managed and mitigated in several ways. The first is simply improving the channel, but this can add excessive cost to the system. Another option is active transmitter based cancellation as found in some links such as Gigabit Ethernet [4]. Arguably, the I/O design complexity could be prohibitive with respect to area, cost, and power. The final option, which is considered in this paper, is to investigate encoding schemes targeted at crosstalk reduction.

II. Current Encoding Schemes in Memory Interfaces

Encoding schemes change raw data patterns to improve the interface performance with respect to signal integrity and link performance. Fig. 4 shows one taxonomy of encoding schemes present in high-speed links. Temporal encoding is commonly found in serial links and encodes data in time. For example 8b10b encoding is quite ubiquitous and forces data transitions to improve clock & data recovery (CDR) performance [5]. Accordingly, 8b10b is sometimes referred to as a minimum transition density code.

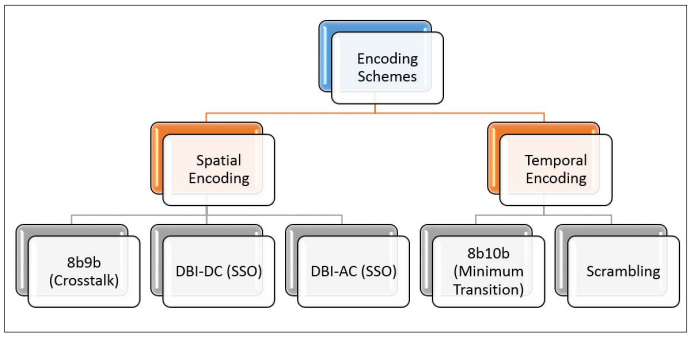
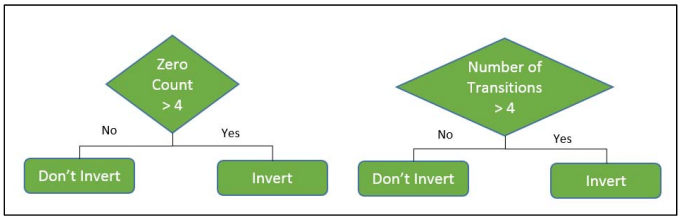


Fig. 4 Encoding Scheme Classification

The other group of encoding schemes can be referred to as spatial encoding as the encoding is done across data lanes instead of in time. Sometimes the cost of having this control for transmitted data is an extra pin as with recent memory interfaces with their use of a special DBI pin. DBI-DC is present in GDDR5/GDDR5x interfaces and works as shown in Fig. 5(a). The encoding is done on the data word being transmitted which eases the encoding complexity. DBI-AC works by analyzing the transition vector which involves a bit more complicated logic since the previous data word needs to be traced as shown in Fig. 5(b).



5(a) 5(b)

Fig. 5 DBI-DC & DBI-AC Algorithm

DBI-AC can be represented with by a transition vector corresponding to which bits switch at given time. This can be accomplished with an XOR as shown in (1)

$$V_i = X_i \; XOR \; X_{i-1}$$
$$X_i: \text{8x1 data vector at bit time } i$$
$$V_i: 8x1 \text{ transition vector to at bit time } i \quad (1)$$

Crosstalk encoding seeks to target crosstalk aggressors and avoid simultaneous toggling to improve eye margin. Fig. 6 shows the crosstalk response in a HBM silicon interposer channel environment where the nearest neighbors are the dominant aggressors. For such a channel, it would be advantageous to avoid simultaneous toggling of neighboring bits.

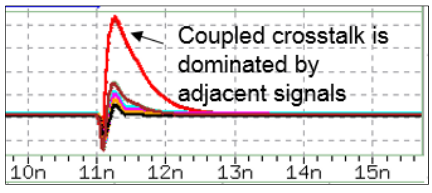


Fig. 6 Crosstalk Time Domain Responses in HBM Channel.

III. 8b9b Crosstalk Encoding Scheme

Similar to DBI-AC, the 8b9b encoding scheme works by analyzing the transition vector as opposed to the data word. Consider a simple example of a 3-bit transition vector. To avoid all 3 bits toggling, the transition vector b111 is forbidden as shown in Fig. 7.

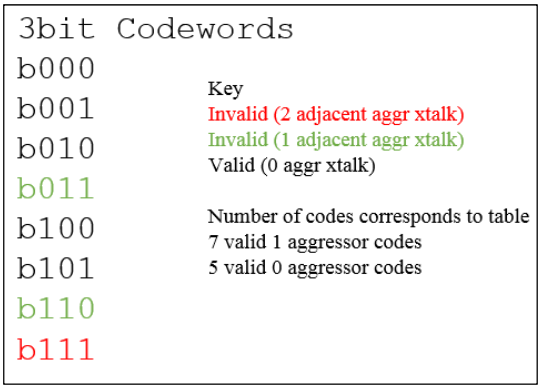


Fig. 7: Codewords Subject to No 3-Aggressor Switching in 3-bit Data Word.

This approach was extended to a data word of 8 lanes. The goal in such a scheme would be to find a code which produces 2^8=256 valid code words with 9 pins. Fig. 8 reveals that there are 274 such code words which satisfy 1-adjacent toggling requirement in a code word of 9 lanes which meets the target of 256 valid code words.

Number of Bits	No Adjacent XTALK Code Words	1 Adjacent XTALK Code Words
1	2	2
2	3	4
3	5	7
4	8	13
5	13	24
6	21	44
7	34	81
8	55	149
9	89	274

Fig. 8 Codewords Subject to No 2 & 3-Aggressor Switching

From Fig. 8, it follows that the number of "No Adjacent" crosstalk code words for a given number of bits (or lanes), F_m, is formulated by the Fibonacci squeence

$$F_m = F_{m-1} + F_{m-2} \quad (2)$$

A close examination of the "1 Adjacent" code words shows that the number of such valid words, G_m, follows a modified Fibonacci scheme

$$G_m = G_{m-1} + G_{m-2} + G_{m-3} \quad (3)$$

This follows as the number of valid code words can be conditioned recursively based on the number of valid code words with less bits. As an example, consider the case of the number of valid 4-bit code words, N_4,which can be conditioned as in (4)

$$N_4 = \{b0,\_\_\_\} + \{b1,\_\_\_\} = N_3 + \{b10\_\_\} + \{b110\ \_\} = N_3 + N_2 + N_1 \quad (4)$$

A transformation basis vector, w, is defined as

$$w = [149, 81, 44, 24, 13, 7, 4, 2, 1] \quad (5)$$

Codes can be generated by mapping the 8-bit data word to the 9-bit code word using this basis vector. The mathematical proof of this basis is beyond the scope of this publication and can be shared in a separate submission. Fig. 9 shows code which verifies the basis function does avoid all 3-neighbor simultaneous switching cases.

```
%check encoding
weights = [149 81 44 24 13 7 4 2 1];
codes = [];
for dword=1:256
d = [ 0 0 0 0 0 0 0  0 0 0 ];
i=9;
for k=[weights]
    if dword > k
        d(i) = 1;
        dword = dword - k;
    else
        d(i) = 0;
    end
    i = i-1;
end
%big endian
d = fliplr(d);
codes = [codes;d];
end
```

Fig. 9 Matlab Code for Encoding Scheme

IV. Implementation of 8b9b Encoding Scheme

The high-level description provided in Fig. 9 needs to be translated into hardware to discern if such a scheme is implementable. One such implementation is shown below in Fig. 10 where the encoder employs brute force successive division with CLA's. Another possible approach would be to use a look-up table (LUT). The design implementation depends on the tradeoffs for the application with respect to area, latency, and design complexity. The decoder implementation could also be implemented by a LUT with the decision again depending on area and latency tradeoffs.

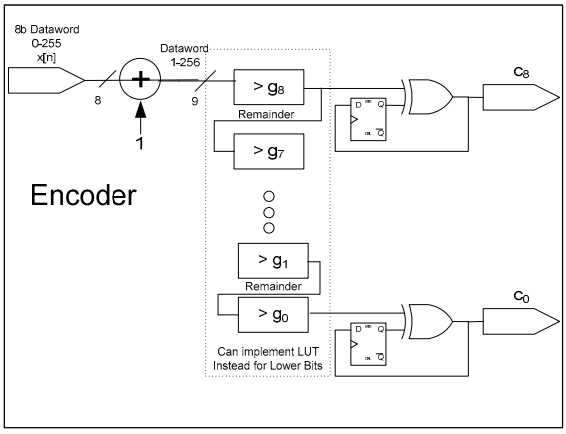
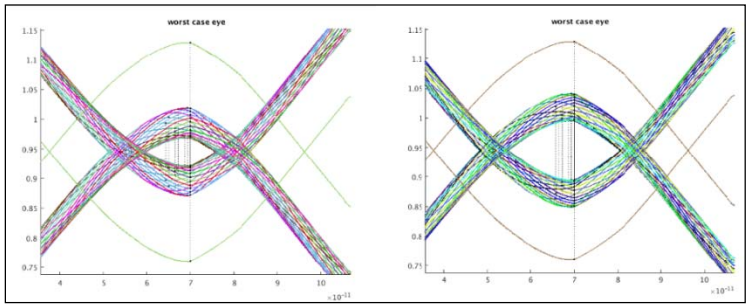


Fig. 10 Possible Encoder Implementation

V. Application of Proposed Algorithms

A. Simulation Results of 8b9b Encoding Scheme in GDDR5x Channel & Silicon Interposer Channel

The 8b9b encoding scheme was applied in simulation to a GDDR5x channel. Fig. 11(a) shows the worst case eye diagram including ISI+Xtalk across various crosstalk phase assumptions all overlaid since the phase of the aggressors is unknown apriori. Fig. 11(b) shows the improvement after applying the encoding scheme. The worst case ISI only case (outer eye) doesn't show any change as expected, but in the presence of crosstalk, the 8b9b encoding scheme improves both eye height and jitter considerably.



(a) (b)

Fig. 11 8b9b Encoding Eye Margin Improvement in GDDR5x Channel

The 8b9b algorithm was applied to a future silicon interposer memory link. The crosstalk response for this link is shown in Fig. 6 and since the proposed link is source-synchronous, the phase alignment is fixed for the analyses. The encoding scheme increased eye height significantly since most of the crosstalk noise occurs at the center of the eye.

B. Summary of Simulation Results Across Interafces

Fig. 12 shows the simulated eye improvement for both GDDR5x and the future silicon interposer link (without equalization). The GDDR5x results are chosen for one phase alignment combination. As shown in Fig 11, there are numerous phase alignment combinations but the scheme does well across all combinations.

	Eye Height (mV)		Jitter (ps)	
	No Encoding	8b9b	No Encoding	8b9b
Silicon Interposer	95	171	35	33
GDDR5x	24	72	64	49

Fig. 12 Summary of Eye Improvement with 8b9b Encoding Scheme

References

[1] Stephen H. Hall and Howard L. Heck, *Advanced Signal Integrity for High-Speed Deisgns*. 2008.

[2] *NVIDIA Tesla P100 Whitepaper*, NVIDIA Coporation, 2016.

[3] *GeForce GTX1080 Whitepaper*, NVIDIA Corporation, 2016.

[4] https://www.10gea.org/whitepapers/gigabit-ethernet/

[5] Abhijit Athavale and Carl Christensen, *High-Speed Serial I/O Made Simple*. 2005.