4Gb/s Router Backplane Design Utilizing 5-Level Signalling for Error Detection

Francois-Fabien Ferhani, Younghwan Kim, Edward S. Li, and Michael D. Mulligan

1.0 Introduction

As demand for higher system data rates increases, techniques for accomodating such data rates must be developed in an efficient, reliable, and cost-effective manner. In this report, we discuss a system design that increases the per line data rate from 2Gb/s to 4Gb/s over an existing router backplane. The design utilizes various methods for increasing the data rate while not substantially degrading signal integrity. We have also implemented a novel 5-level signalling method, which greatly increases system robustness and performance and allows for error detection.

1.1 Overview of Signalling Approach

Our system utilizes differential, bipolar, current-mode signalling. This signalling approach was determined to be more robust to supply noise, cross-talk, and other forms of noise than the alternatives. In addition, rather than attempting to increase the symbol rate of the existing system, we chose to implement a multi-level signal approach. This method was chosen early on in the design process as it seemed that the data-dependent jitter due to cross-talk alone would make a much smaller symbol rate very difficult to obtain.

To mitigate the effects of inter-symbol interference (ISI), we implemented an efficient method for equalization at the transmitter using a 30-tap FIR filter. Similarly, we implemented active cross-talk cancellization at the transmitter to reduce the effects of adjacent aggresor signal lines. These two methods were found to be the most effective in improving system reliability.

To further increase system performance, we utilize adjustable termination resistances as well as receiver offset cancellation.

1.2 Overview of Timing Approach

Our timing system uses bundled timing with phase-alignment at the receiver. This method has the advantage of requiring only four pins and lines for every group of 16 differential signal pairs. In this system, a clock is transmitted alongside each group of signal lines. At the receiver, the phase-alignment sysem senses the phase difference between the clock and each signal and adjusts a variable delay line in a closed-loop to obtain the required 90-degree phase offset for proper data sampling. Those 16 signals are subsequently phase-aligned after being sampled properly.

1.3 Summary of Results

Our system acheives the required data rate of 4Gb/s with a bit error rate (BER) of 7.8e-17. The overall system cost is approximately $3100. This cost estimate includes 16 line cards and 2 switchcards.

2.0 Board-level Design

In this section we discuss the board level design, escape routing of the chips, and electrical model of our card line. The section concludes with a discussion of a cost model for the system.

2.1 PCB Design for Switch Card

Each switch card needs to connect a total of 576 lines from given backplane to the switching circuit. Among them, 512 of the lines are required for 256 pairs of differential signaling. The rest of the lines are for receiving (and sending) differential clock signals from (to) 16 line cards. Our goal in this section is designing a PCB transmission line for these 576 lines with good electrical characteristics and reasonable cost.

2.2 Escape Routing

The key restriction of escape routing is that we can only connect two rows of pins into each layer of the PCB stackup. The inner row suffers from additional 6-mil limitation of metal width, which is coming from the “Via Land Diameter(VL)” or “Solder Mask Diameter(M)” [1]. Using two chips for each switch card solves this problem, which enables 1:1 row to layer connections. The two-chip approach is also economical compared to implementing one-chip, with 8-layer stackup compatibility and smaller PCB board size requirement. The fact that two-chips require smaller PCB board seems counterintuitive at the first glance. Figure A.1 shows the reason why it happens. When we use two chips, the 1:1 row to layer connection makes the overlapping of two sides of chip peripheral possible.

The major drawback of twin chip is the necessity of additional connectivity between the two chips. For those connections, we require two more layers in the PCB stackup, which is described in next section. It does not harm the overall performance because the delay difference between each line card connection is dominant compared to the delay between twin chips.

2.3 Electrical model of the line

Figure A.2 shows the side-cut of the PCB board. It is based on Figure 2-13 of [2]. The special feature here is the usage of a differential structure. Lines A1 and A2 form a differential pair for signal A, while lines B1 and B2 form another differential pair for signal B. Because of the characteristics of the differential pairs, the crosstalk only occurs between Ax and Bx. The advantage of this configuration is that the influence of B1 and B2 on A1 cancels each other, which results in

smaller effective coupling capacitance and mutual inductance between Ax and Bx. All of the other subtle influences are neglected because the layers are shielded and homogeneous.

Layers 3 and 4 are used for auxiliary connections between chips. For the twin-chip structure, they need to be connected to two layers, which makes their metal width 6-mil.

2.4 Price model

We found that the electrical characteristics of the backplane were the dominant bottleneck of system performance. Our basic approach was to choose low cost parts unless their effect on signal degeneration was comparable to that of the backplane. The overall cost model contains the price of each line and switch card, as well as the price of each chip.

8-row-Teradyne-VHDM-HSD connectors require 19.2cm width for 512+64 pins. The height of card can be estimated from 40cm (given by spec) + 1.9cm (half chip width). With reasonable engineering margin, we take 46cm*20.3cm as the switch card size, which is exactly one third of industrial-standard-panel size (460*610mm). This is another advantage of the two-chip approach, because with a single-chip design we can only make two PCB cards from one standard panel. With 8-layer stackup and on-chip crystal oscillator, panel price for switch card and line card are $93.40 and $20.00 each.

Pin count and power consumption should be considered for chip-cost estimation. Chips for switch cards are utilizing all of their pins (256 for data signals, 36 for clock signals, 256 for chip-to-chip communication on the switch card, and 220 for power and other functions), whereas chips for line cards need only 256 pins approximately. Switch card chips also consume more power. For the worst case, each differential pair consumes 2*45.4mA. Considering that only half of the pins are driven by the transmitter on the chip and the other half is simply for receiving signals, switch card and line card chips consume 33W and 4W, respectively. A simple calculation shows the switch card chip and line card chip costs $378.40 and $62.00 each.

We have total 16 line cards and 2 switch cards yielding a total cost estimate of is $3015.

3.0 Signalling

3.1 5-level Signalling

Our signalling method uses a novel 5-level signalling approach outlined in [3]. Throughout this section, we will assume voltage-mode, single-ended signalling for simplicty. We will also assume a swing of 2V peak-to-peak, centered about 0.

For comparison we begin by examining the case of 4-level signalling (4-PAM). Figure 1 provides an illustration of 4-PAM signalling. Each line transmits two bits per symbol. Neglecting bounded noise sources, the net margin is half the level between signals: V_{net_margin} =0.66/2=0.33

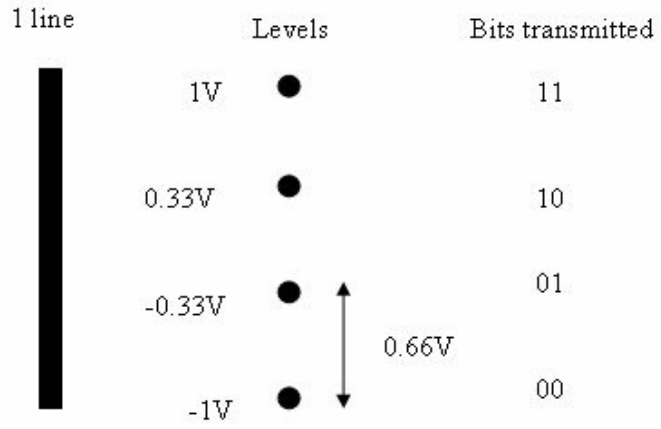


FIGURE 1. 4-PAM signalling

For 5-level signalling, we view 4 signal lines at a time for a total of $5^4 = 625$ possible combinations of levels Figure 2 shows an illustration of the possible level combinations for the 4 bundled signals.

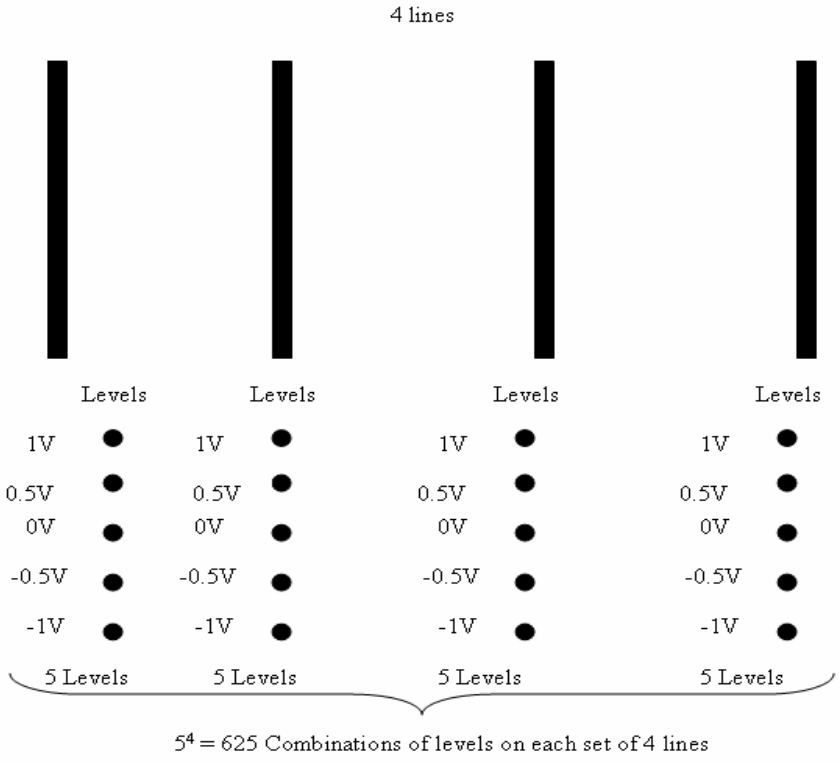


FIGURE 2. Illustration of 4-line 5-level signalling

4-PAM allows for a total of $4^4 = 256$ level combinations with 4 grouped lines. This is adequate to transmit one byte. But in the case of 5-level signalling we have more level combinations than we need. Since 625 > 2*256, we can afford to use only every other combination of levels to encode our bytes. This table represents the levels on each line:

TABLE 1. Comparison of signal levels for 4-PAM and 5-level signalling

Byte	Level Combination (4-PAM)	Level Combination (5-Level)
00 00 00 00	-1V -1V -1V -1V	-1V -1V -1V -1V
00 00 00 01	-1V -1V -1V -0.33V	-1V -1V -1V 0V
00 00 00 10	-1V -1V -1V 0.33V	-1V -1V -1V 1V

In the 4-PAM case the minimal distance between two level combinations is the minimal distance between adjacent levels (0.66V). However, in the 5-level case, since we only use every other combinations of levels, the minimal distance between two level combinations is twice the minimal distance between levels, or 1V, which is better than the 0.66V of 4 levels. 5-level signalling also has the ability to detect the most common errors. Since only 256 of the 625 level combinations will be valid data bytes, most of the other combinations will be detected as errors and the data will then be re-transmitted.

If bounded noise sources are neglected, 5-level signalling will always outperform 4-PAM signalling. If bounded noise sources are accounted for and are not negligible, depending on the severity of the noise, 4-PAM may be better suited for obtaining higher signal integrity. Though our backplane model is quite lossy and fraught with severe cross-talk disturbances, we chose still to implement 5-level signalling because the BER we obtained was sufficient to make its other features more attractive.

3.1.1 Bit Error Rate (BER)

Counter to what may be implied by the above discussion, we cannot simply calculate our BER assuming a new net margin of 0.5V. For an accurate analysis of BER, we need to look at the two cases where the transmitter will decode an undetectable error: 1) 2 or more of the 4 lines are spurious by 1 level or more 2) 1 of the 4 lines is spurious by 2 levels or more.

Case 1: 2 or more spurious lines.

There are six combinations of choosing 2 lines out of the 4. We compute the probability of case(1) as follows:

$$P_1 = C\big|_4^2 \cdot BER^2(0.25V_{swing}) \cdot \left[1 - BER^2(0.25V_{swing})\right] + C\big|_4^1 \cdot BER^3(0.25V_{swing}) \cdot [1 - BER(0.25V_{swing})] + C\big|_4^0 \cdot BER(0.25V_{swing})$$

Where BER(V) is the bit error rate for a net margin of V, C^x_y denotes that x lines are correctly received out of y total, and V_{swing} is 1/2 of the total swing per line. For typical BERs, the first term is dominant.

Case 2: 1 line is spurious by 2 or more levels.

The probability of one line being spurious by two or more levels is

$$P_2 = C|_4^1 \cdot BER(0.75V_{swing}) \cdot [1 - BER(0.75V_{swing})]^3$$

The total BER is the sum of P_1 and P_2 multiplied by 8 because an entire byte is in error. Though there exist two undetectable error types, case (1) is the dominant error mechanism since the effective swing for case 2 is large enough the make P_2 relatively small. Thus, our effective BER is

$$BER_{eff} = 48BER^2(0.25V_{swing})$$

3.1.2 Implementation

The coding/decoding of the data bytes can be performed by a ROM look-up table. The table needs only 256 entries for data. As we have 312 total level combinations that can be reliably used with this method, we have 56 extra levels we can use for system commands. One of these extra level combinations will be used to tell the system to retransmit data when a detectable error is decoded. The remaining commands can be used to tell the system to calibrate termination impedances, receiver offset, or may even be made available to the end user to specify in a programmable ROM (one-time) or RAM.

The minimal distance of 2 levels provides us the ability to detect an error at the receiver, not to correct it. Any decoded combination not in the look-up table is interpreted as an error. The erroneous data is then retransmitted and normal operation ensues. We ignore the incoming bits until a "corrected data" signal is decoded (this can be one of the extra 56 level combinations). For the worst-case line, we will need to ignore approximately 42 bytes during this time (1 to for the "resend" signal, 20 to go through the line towards the other card, 1 for the "corrected data" signal, and 20 to go through the line again = 42 cycles). This means that we need a buffer to store those 42 bytes and that, technically, our system requires a symbol rate of slightly lower than 500ps to run at 4Gb/s. We simulated our system with a symbol time of 490ps[1] to account for this subtlety.

Another interesting feature is its utility in harsh environments. 5-level signalling enables us to detect if all our bits are random. If for example in a space probe a solar eruption sent such a quantity of heavy ions that all our bits were random, roughly half of them would fail, the chip can easily see that half of its received bits are wrong and require a resend for all bits.

3.2 Equalization and Cross-talk Control

This section discusses our methods for suppressing the effects of ISI and cross-talk [2]. We found these to be the most valuable tools for improving signal integrity.

1. However, throughout this report we assume a symbol time of 500ps so as not to be cumbersome.

3.2.1 Equalization

In order to control ISI, a finite impulse response (FIR) equalization filter at the transmitter was designed. The purpose of this filter was to shape the impulse response of the channel. Pulse shaping was done such that when the impulse response was sampled at multiples of the bit time, all sample points were zero except for the current bit time. Said another way, the desired channel after sampling was a delta function. In the frequency domain, this amounted to flattening the spectrum over the transmission bandwith.

The technique used to design the equalization filter was the minimum mean squared error, linear estimation (MMSE-LE) method. In general, this method produces the filter coefficients that minimize the squared-error between the equalized pulse response and the desired delta function. Over the space of all linear filtering techniques, this method has been shown to be optimal for a fixed number of filter coefficients.

It should be noted that while there is always an optimal solution for a given number of filter coefficients, typically having more coefficients produces better results. The reason for this is with more filter coefficients, the time instants close to the sample times will also be nearly zero. This is important if the sampling time is slightly jittered since the impulse response values close to the sample times will now contribute to ISI.

Furthermore, it is rare to achieve zero error since this case often requires an infinite number of filter coefficients. In essence, an infinite impulse response filter is needed to achieve this theoretical best case solution. All of this implies that obtaining the largest tolerable number of filter coefficients is highly desirable.

The inputs to the process were the unequalized channel response to a lone pulse sampled at multiples of the bit time and the number of desired filter coefficients. In this project, the number of equalization filter coefficients was chosen to be 30. The reason why this number was chosen was it presented a good tradeoff between residual error at and around the sample times and system complexity. Appendix B contains our Matlab, PERL, and HSpice code used for simulations.

3.2.2 Cross-talk Control

Cross talk noise was handled in a similar manner as ISI except that instead of cancelling noise from other symbols on the line, noise coupled from neighboring differential lines was cancelled. Unlike equalization, the desired effect of such cancellation was to have zero signal level due to cross talk. Restated, the cross talk was forced to be zero at all bit sample times.

The technique used to design the FIR filter for cross talk cancellation was the least squares (LS) approximation method. Much like the equalization design, this method produced the optimal coefficients for a linear filter. The difference between this and the MMSE-LE method is that the latter technique takes into account the energy of the transmitted signal while the LS approximation does not require this information since it is zero forcing.

The inputs to this process were the desired number of filter coefficients and the the sampled signal due to cross talk. The goal was to force these sampled values to zero with a fixed number of filter

coefficients. In this project, the number of coefficients for the cross talk cancellation filter was chosen to be 3. Like the equalization filter, having more filter taps provided more robustness to sample time jitter and the chosen number allowed for this robustness while limiting complexity.

It should be noted that cross talk cancellation produces a secondary noise coupling. That is, it changes the equalized pulse response on the original line. Hence, an iterative process is needed to equalize the signalling line, cancel cross talk from the neighboring line, and then check the signalling line to make sure it is still relatively equalized. In a complete system, these filter tap coefficients for both ISI and cross-talk cancellation would be determined adaptively.

If the sampled channel response has deviated too far from a delta function, then the process must be repeated. In this project, iteration was not needed since the number of filter taps used kept the channel response from deviating.

The equalized channel response with the cross talk noise signal is shown in Figure A.5.

3.3 Signalling Details

We chose our levels such that the maximum level does not exceed the +/-1.2V operating range of our process. To get the maximum current driven, we sum the absolute values of all of our 30 ISI taps and our 3 cross-talk taps, and multiply that by the drive strength of our driver. This will yield the maximum current sent on the line, and this multiplied by 25 Ohms (50 in parallel with 50) should not exceed 1.2V. With 30mA, our maximum level is measured to be 1.13V. The 5-levels are evenly spaced and the gross margin at the receiver is 125mV.

We noticed that a slower rise time narrowed the eye because it takes longer to get to that level, even though faster rise time increases the effects of crosstalk. By putting drivers in parallel, we were able to achieve 50ps for 30mA.

Termination resistors improved the system greatly because reflections are absorbed, even if it lowers the voltage levels, it significantly reduces ISI. So we use matched 50-ohm terminations.

The driver we use consists of two current-mode drivers in parallel, as we are limited to a drive strength of 20mA for each of them. The receiver is a single-stage Output Offset Storage receiver (see section 6 on Offset cancellation). The terminations are calibrated to match their respective line impedances (see section 6.2).

3.4 Noise Analysis

The transmitter offset in the differential current-mode case for our current-mode drivers will primarily be due to the current drive mismatch (5%). We can evaluate crosstalk and ISI by viewing the impulse response for the ISI and the victim line for the crosstalk. We sample these responses at our sampling point (once per symbol) and add the worst case. However, since our methods for equalization and crosstalk calculation cancel exactly the crosstalk and ISI at the sampling points, we have to sample it at a point shifted by our timing margin.

We typically send 30mA into an effective 25-ohms (750mV). After attenuation by the line we receive 500mV, for a differential swing of 1V swing. For 5-level signalling, our gross margin is 125mV.

TABLE 2. Bounded Noise Sources

Source	Value
ISI	20mV
X-talk	55mV
Rx offset	8mV
Rx sensitivity	5mV[a]
Net Margin	**32mV**

a. 1/2 of 10mV because lines are differential

We determined the Gaussian noise to be that solely due to the backplane. There will be resistive thermal noise, but we calculated it to be on the order of micro-volts and thus deemed it negligible.

For our system, we have a voltage signal-to-noise ratio, VSNR, of approximately 6.4. This yields a BER of 7.8e-17.

4.0 Timing and Synchronization

The global system clock is based on a single crystal oscillator running at 250MHz. This clock is distributed to each chip where it is then used to synthesize the required 2GHz on-chip clock. A standard phase-lock loop (PLL) with a divide-by-eight block in the feedback loop is used for this frequency synthesis and is shown below in Figure 3.

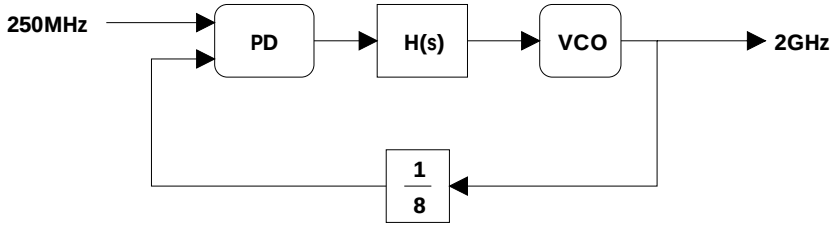


FIGURE 3. Frequency synthesis of 2GHz on-chip clock using phase-locking

The details of PLL operation can be found in [4], or many other useful references, however the basic analysis is as follows. The phase of the reference frequency is compared to that of the divide-by-eight output frequency. This comparison in turn drives a loop filter, H(s), which determines the average value of phase offset as well as ensures loop stability. The average value then drives a voltage-controlled oscillator (VCO) which generates, in locked mode, a frequency that is eight times that of the reference frequency.

In order to ensure that data is sampled properly and that signals interacting between different clock domains are not degraded, we have implemented timing and synchronization methods, respectively. Those methods will now be discussed.

4.1 Timing

Our timing approach is an amalgamation of the per line and bundled timing methods from [lecture notes]. To save board area and pin count, we transmit one clock per 16 lines. At the receiver the clock is distributed and phase-aligned separately to each of the 16 data signals. Subsequently, the 16 data signals are phase-aligned to each.

The data and clock are triggered by the same local clock at the transmitter, thus the jitter and skew of the local clock will not affect the timing of these two signals. At the receiver, three samples of the data are taken at quadrature intervals of the clock cycle, then a decision is made as to whether the clock is slow, fast, or locked to the correct value. Adjustments are made in a negative feedback loop to adjust the variable-delay line. This is shown below in Figure 4.

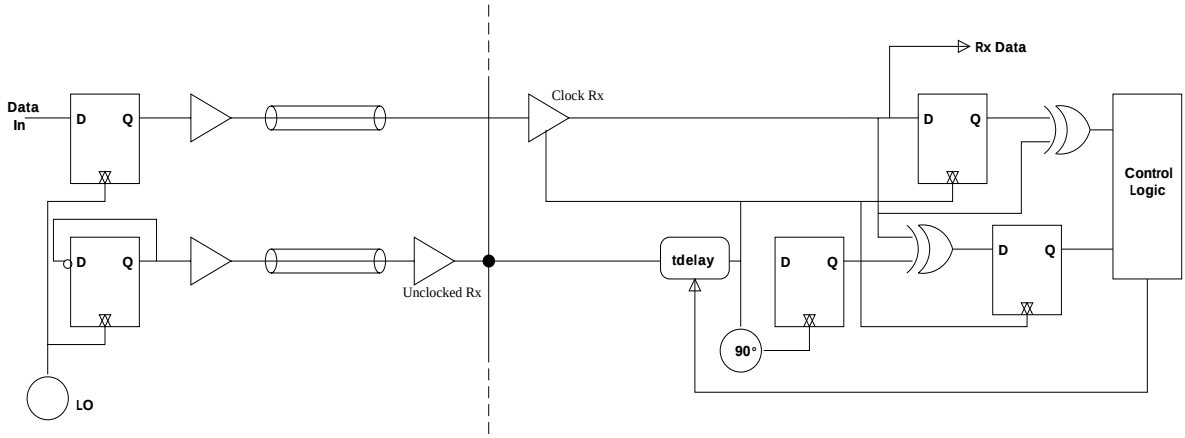


FIGURE 4. Timing circuit

4.2 Timing Noise

The phase alignment method cancels the skew between the received data and clock. The maximum jitter will be 36.25ps (20ps due to the transmitter, 10ps assumed for the unclocked receiver, and 6.25ps for the delay line with a maximum delay of 250ps). Accounting also for an aperture time of 20ps, our required horizontal eye opening is 92.5ps.

4.3 Synchronization

When the various line card and switch card signals interact, they need to be synchronized. The system is plesiochronous due to variations in frequency between on-chip VCOs (20-ppm). A simple method for synchronization of this system is a plesiochronous FIFO synchronizer discussed in [5], which is not discussed in detail here for brevity.

5.0 System Simulations

In this section we summarize the system simulations and results.

5.1 Eye Diagram

Figure 5 shows the eye diagram for our 4Gb/s 5-level system. Here we are not running at 500ps, but rather 490ps (4.08 Gbits/s) to account for the the bits we need to resend when a detectable error is decoded. We need to resend 40 bytes approximately 4 times per second, which is much less than the extra 80 million bits we have when running at this speed. All of the calculations and equalization were done at 500ps, and so the system is actually not optimized for the faster data rate (though, it works quite well nonetheless).

The eye accounts for: transmitter mismatch (5%), impedance mismatch (5%) both at transmitter and receiver, crosstalk, ISI, and data-dependent jitter.

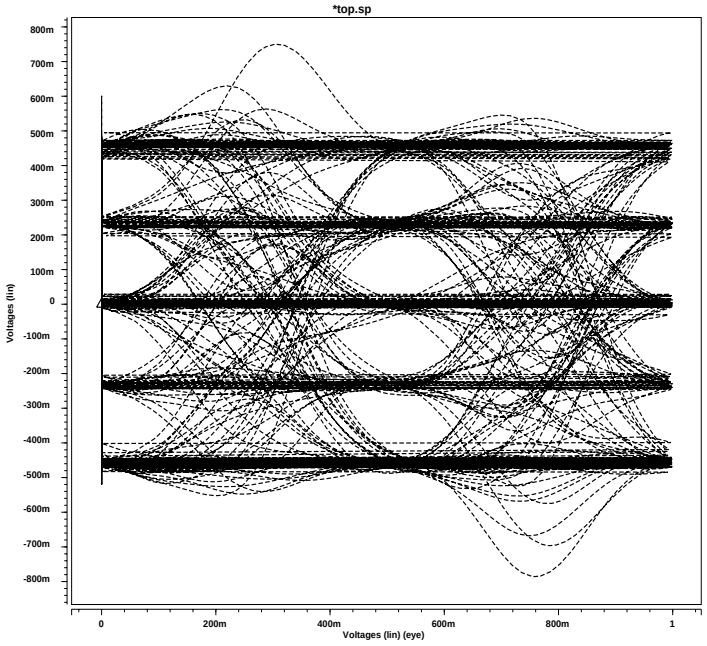
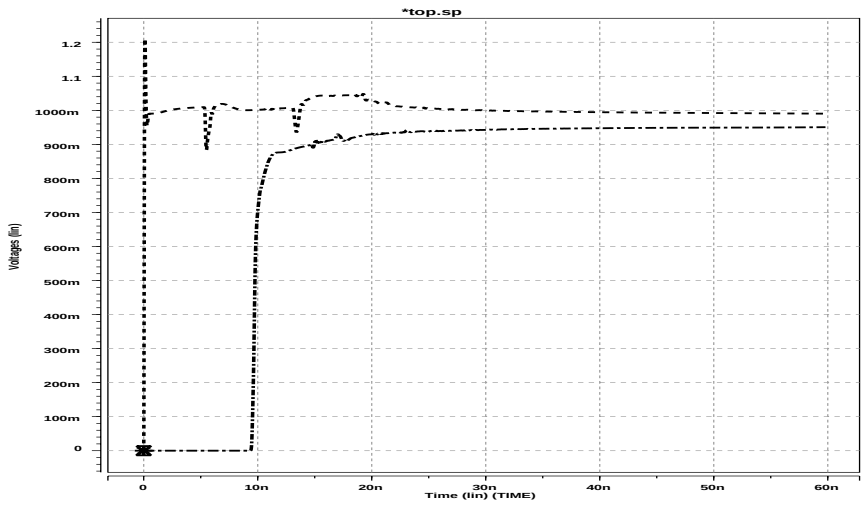


FIGURE 5. Eye Diagram for 4Gb/s system using 5-level signalling

5.2 TDR and TDT

Figure 6 shows the time-domain reflectometer (TDR) and time-domain transmission (TDT) waveforms for our system.



6.0 Additional Circuits

To reduce the effects of several sources of system noise, we use methods to cancel receiver offset as well as calibrate termination resistors. This section provides the details for how these methods are implemented.

6.1 Offset Cancellation

We use a simple method for receiver offset cancellation that stores the offset on switched capacitors at the output of the receive amplifier. This method--known as output offset storage (OOS)--is widely used in high-performance data conversion systems [6]. We chose this method for its simplicty of implementation as we only required a modest reduction of receiver offset.

Figure 6 shows a simplified schematic for a single-stage OOS receiver. When the receiver decodes the ‘calibration’ bit combination, data transmission ceases. During this time, switches S1 and S2 are open while S3, S4, S5, and S6 are shorted to ground and the offset of the preamplifier is multiplied by the gain and stored on the output capacitors. When data transmission resumes, the offset of the preamplifier is completely cancelled and and the input-referred offset of the latch has been reduced by a factor equal to the gain of the preamplifier.

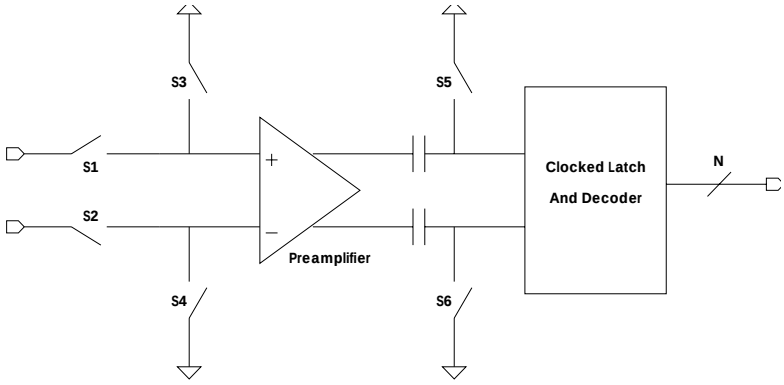


FIGURE 6. Simplified schematic of OOS receiver

The gain-bandwidth product of the preamplifier is limited to be 10-GHz. Thus, the gain was chosen so as not bandlimit the signal. We pessimistically assumed a required bandwidth equal to our clock frequency (2GHz). This allows us a preamplifier gain of 5, so that the input-referred receiver offset was reduced from 40mV to 8mV.

6.2 Calibrated Termination Resistances

To reduce signal attenuation and reflections that degrade signal integrity, our system also uses the available digitally adjustable resistors to calibrate termination resistances to within 5% of the nominal value. The basic method for digitally adjustable resistors is described in [Gabara 1992]. For our system, the calibration works slightly differently. When the system calibration signal is detected, the line transmits a full swing level (30mA in our case). A TDR-like wave form will be detected by the ADC. However, it first passes through a low-pass filter to smooth out sharp transitions due to line discontinuities. The average voltage input to the ADC will be

$$V_{ADC} = 20mA \cdot \frac{R_{cal} \cdot Z_0}{R_{cal} + Z_0}$$

The ADC detects the line level and sends that data to a control system. The control system senses how the input voltage has changed from the previous cycle (if it is the first cycle in the calibration then it always chooses to increase the resistance as a means to generate a reference) and adjusts the DTA until it acheives a maximum value for V_{ADC}. This maximum value corresponds to when the termination resistance equals the magnitude of the line impedance.

Assuming that the ADC is running at the same frequency as the system and that the adjustable resistors vary in increments of 5% from their present value, the system would require approximately 15 cycles (7.5 ns) to calibrate back to 50-ohms from a 25-ohms starting value. A mismatch such as this is assumed only to occur at system startup, so typical calibration times will not be this severe and, similarly, will allow the use of a slower, low power ADC.

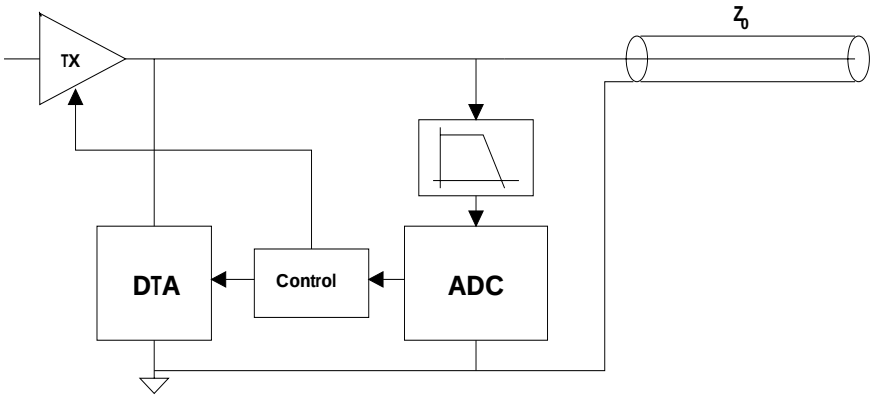


FIGURE 7. Termination resistance calibration system

This method has the advantage that it does not rely on any reference resistor. It also matches to any line impedance, including any slight variation in line impedance. Though Figure 7 shows the system for the transmitter, a similar system, albeit with slight modifications, can be used at the receiver.

With this system, the worst-case impedance mismatch will be approximately 4.5% (2% due to variation from register settings and 2.5% due to the incremental nature of the calibration adjustments).

7.0 Conclusion

We achieved the required data rate increase from 2Gb/s to 4Gb/s over the existing, lossy backplane. Additionally, we implemented a novel 5-level signalling approach which detects the most common errors at the receiver. For our particular backplane model, it may be true that a 4-PAM signalling approach would yield a lower BER. However, the 5-level signalling method allows us greater features, such as detection of random bits streams on the lines and efficient system calibration.

8.0 References

[1] A. Maheshwari, et al, “Board Routing Guidelines with Xilinx Fine-Pitch BGA Packages” *Xilinx Application Note: Virtex Series*, 2002.

[2] W. Dally, *Digital Systems Engineering*, Cambridge University Press, 2001.

[3] M. Hatamian, et al, “Design for Gigabit Ethernet 1000Base-T Twisted Pair Transceivers” *IEEE 1998 Custom Integrated Circuits Conference*, 1998.

[4] T. H. Lee, *Design of CMOS Radio Frequency Integrated Circuits*, Cambridge University Press, 1998.

[5] S. Harris, et al, EE273 Course Lecture Notes, Stanford University, 2003.

[6] B. Rehzavi, *Principles of Data Conversion System Design*, IEEE Press, 1995.

9.0 Appendix A: Detailed Figures and Plots

This appendix includes many detailed plots and figures to supplement our design report.

Figures A.1 and A.2 are supplements to Section 2.

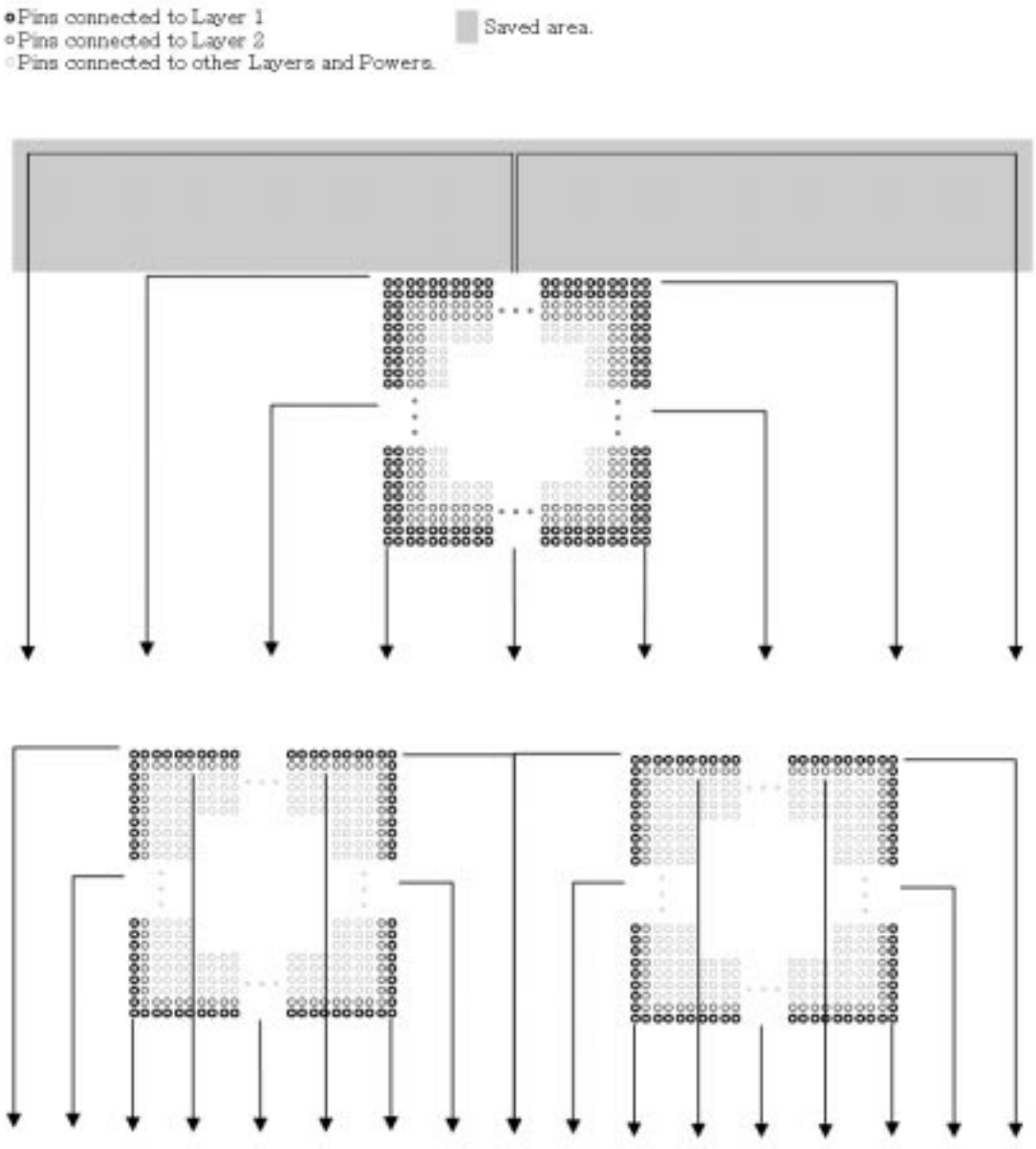


FIGURE A.1. Description of escape routing

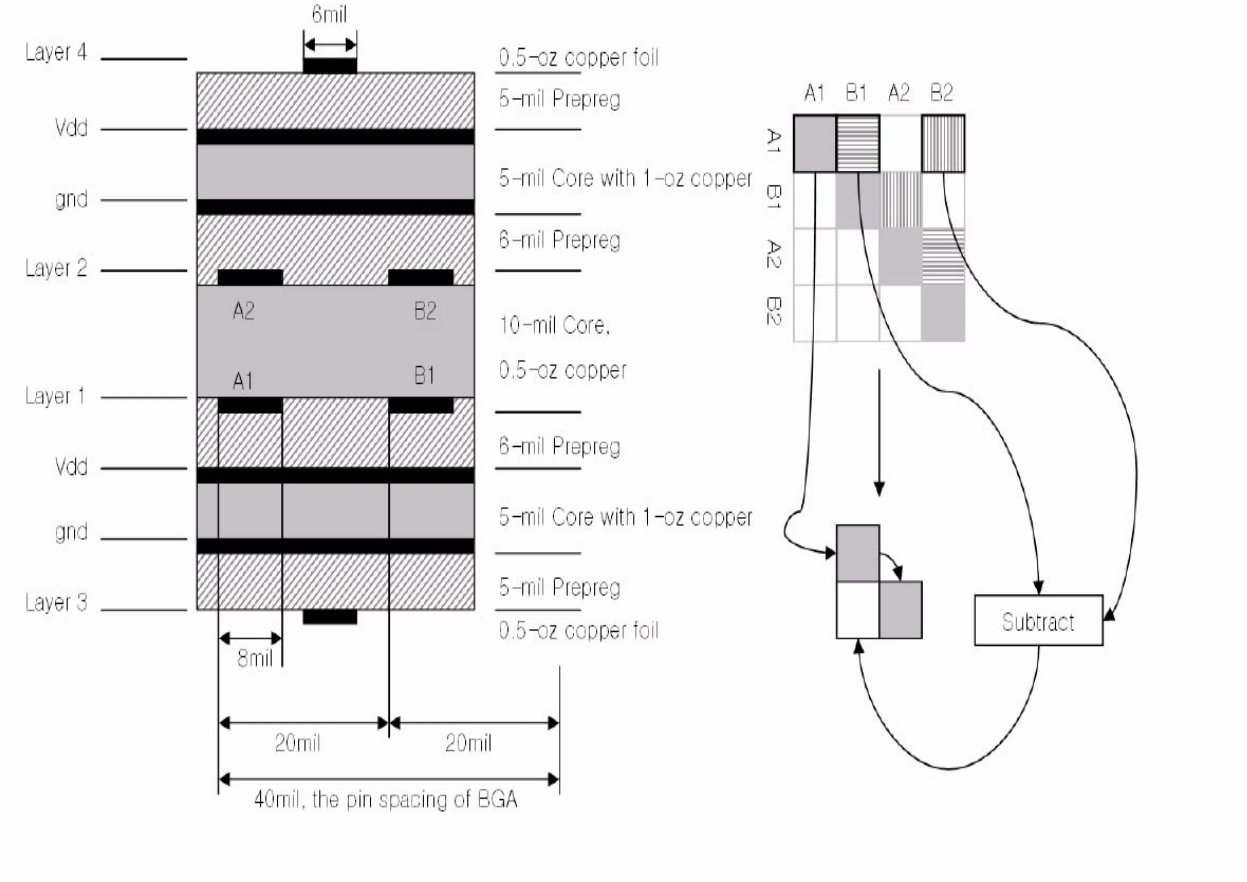


FIGURE A.2. Cross-sectional view of PCB board stackup

Figure A.3 on the following page shows our equalized lone pulse simulation results. The lone "1" is on the center of the diagram, and the lone "0" is on the eye that wraps up. Equalization enables us to pull the lone pulses from rail to rail.

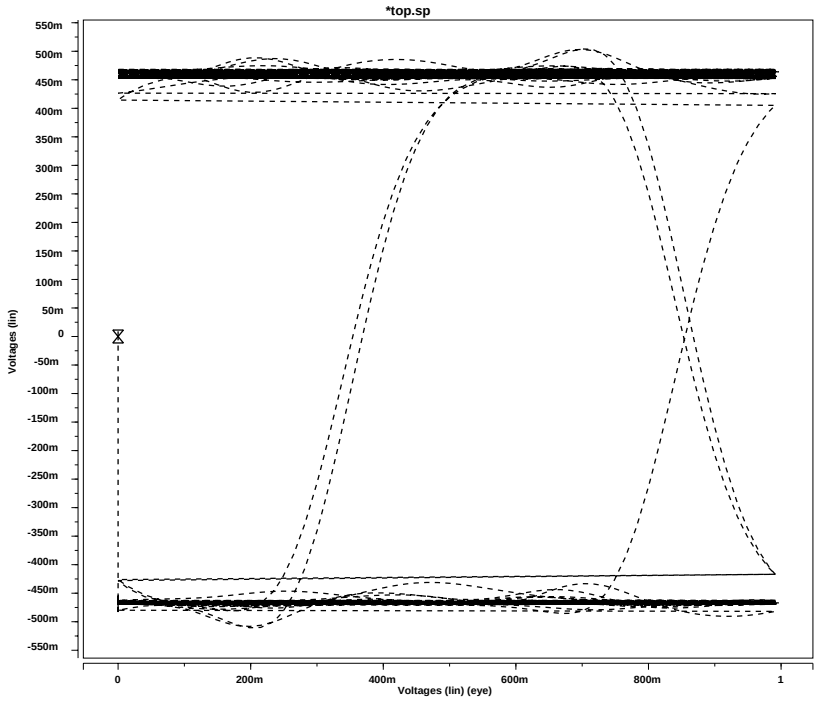


FIGURE A.3. “Lone pulse” simulation

Figure A.4 shows the transmitter to receiver unequalized channel frequency response.

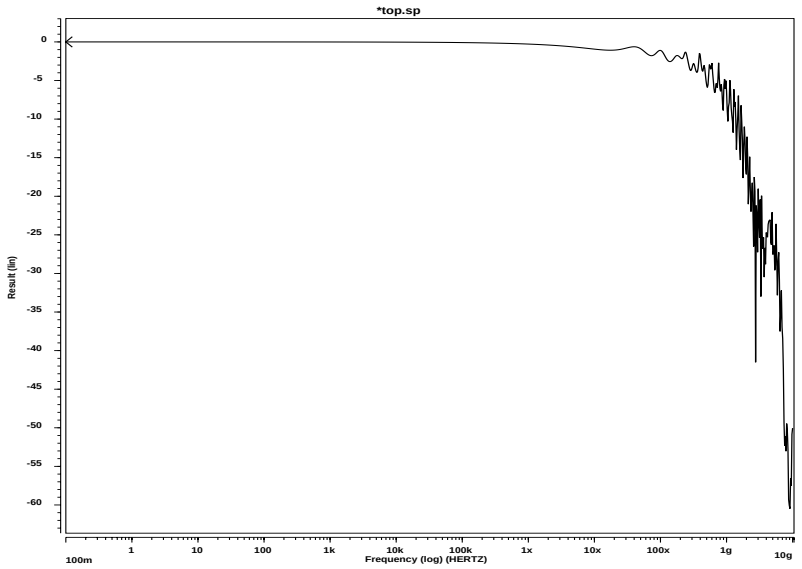


FIGURE A.4. Unequalized channel frequency response in dB

Figure A.5 shows the equalized channel impulse response.

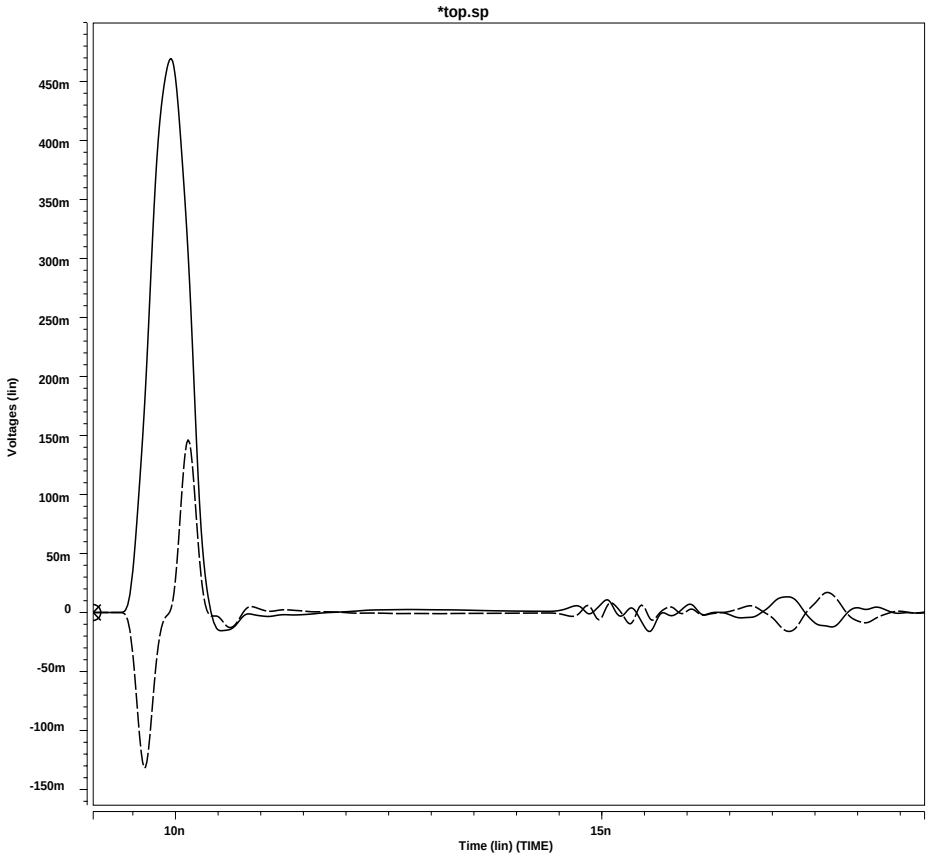


FIGURE A.5. Equalized channel impulse response

10.0 Appendix B: Simulation Code

The simulation code we used to generate tap coefficients (Matlab), generate data input (PERL), and run simulations (SPICE) is attached on the following pages.

calc.m

```
clear;


% ISI cancellation
load pkTF;
pkT = pkTF' * 1e-3;
Ep = pkT*pkT';
q = conv(pkT,fliplr(pkT))/Ep;
%Ex = ((.8/2)^2 + (.8/6)^2)/2;
%Ex = ((1/2)^2 + (1/6)^2)/2;
Ex = (.750)^2;
n = (5e-3)^2;
SNRmfb = Ex*Ep/n;
10*log10(SNRmfb)
qdfe = q;
m = (length(q)+1)/2;
qdfe(m) = qdfe(m) + 1/SNRmfb;
r = roots(qdfe);
%r = roots([.9/1.81 1.1 .9/1.81]);
g = qdfe(1);
%g = .9/1.81;
for a=1:length(r),
  if (abs(r(a)) > 1),
    g = g*r(a);
  end;
end;
g = abs(g)
SNRdfe = g*SNRmfb;
SNRdfeu = g*SNRmfb - 1;
10*log10(SNRdfeu)
```

```
d=0;[SNR w]=dfecolor(1,pkT,40,0,d,Ex,n*[1 zeros(1,39)])
w=w’/w(1)

% X-TALK cancellation
% example: say xtF = [18, -18, 1] (sampled from xrchip2)
% say pkT has 33 taps
% goal: want 3 taps for X-TALK cancellation
% so, output has 33+3-1=35 taps
% 1] since want to cancel xtF, multiply by -1
% 2] now extend xt to 35 taps
% 3] setup convolution matrix; need R(1st row) and C(1st col)
%    note that R has 3 entries (matches number of X-TALK cancel taps)
%    note that C has 35 entries (matches output tap number)
% 4] use toeplitz(C,R)
% 5] use least squares approx formula/method where "y" is "xt"
load xtF;
xt = [xtF; zeros(length(pkT)-1,1)] * -1e-3;
R = [pkT(1) zeros(1,length(xtF)-1)];
C = [pkT’; zeros(length(xtF)-1,1)];
A = toeplitz(C,R);
x = (A’*A)^-1*A’*xt
```

DFEcolor.m

```
function [dfseSNR,w_t]=dfsecolorsnr(l,p,nff,nbb,delay,Ex,noise);
% ------------------------------
%**** only computes SNR ****
% l     = oversampling factor
% p     = pulse response, oversampled at l (size)
% nff   = number of feedforward taps
% nbb   = number of feedback taps
% delay = delay of system <= nff+length of p - 2 - nbb
% Ex    = average energy of signals
% noise = noise autocorrelation vector (size l*nff)
```

```
% NOTE: noise is assumed to be stationary
% outputs:
% dfseSNR = equalizer SNR, unbiased in dB
% created 4/96
% ------------------------------

size = length(p);
nu = ceil(size/l)-1;
p = [p zeros(1,(nu+1)*l-size)];

% error check
if nff<=0
  error('number of feedforward taps must be > 0');
end
if delay > (nff+nu-1-nbb)
  error('delay must be <=  (nff+(length of p)-2-nbb)');
end
if delay < -1
  error('delay must be >= 0');
end
%if delay == -1 %assume a DFE
%  delay = nff-1;
%end


%form ptmp = [p_0 p_1 ... p_nu] where p_i=[p(i*l) p(i*l-1)... p((i-1)*l+1)]
ptmp(1:l,1) = [p(1); zeros(l-1,1)];
for k=1:nu
  ptmp(1:l,k+1) = conj((p(k*l+1:-1:(k-1)*l+2))');
end

% form matrix P, vector channel matrix
```

```
P = zeros(nff*l+nbb,nff+nu);


%First construct the P matrix as in MMSE-LE
for k=1:nff,
  P(((k-1)*l+1):(k*l),k:(k+nu)) = ptmp;
end
P;
%Add in part needed for the feedback
P(nff*l+1:nff*l+nbb,delay+2:delay+1+nbb) = eye(nbb);
%P
temp= zeros(1,nff+nu);
temp(delay+1)=1;
Rn  = zeros(nff*l+nbb);
Rn(1:nff*l,1:nff*l) = toeplitz(noise);
%Rn
Ex*P*P';
Rn;
Ry  = Ex*P*P' + Rn;
Rxy = Ex*temp*P';


%Force the inverse of Ry to be hermitian.
%There are numerical problems sometimes, so that inv(Ry) is not hermitian when it
should be
%This next line was done to force it to be hermitian
IRy=.5*(inv(Ry)+inv(Ry)');


w_t = Rxy*IRy;
sigma_dfse = Ex - w_t*Rxy';
dfseSNR = 10*log10(Ex/sigma_dfse - 1)
```