EE273 Digital Systems Engineering
Class Project Report

High Speed Router Design

Team Members:

Frank Nothaft/ 005459904 / fnothaft@
Karthika Periyathambi/ 05590444 / karthipd@
Krishna Teja Malladi/ 05582221 / ktej@

1. Introduction

1.1 Problem Statement: The goal of the project is to design a 640Gbps router system architecture that interconnects 16 line cards via crossbar cards with redundancy. This was to be achieved by the design of a signaling system and wire routing in order to strike an optimal balance between cost, speed, reliability, area, routing complexity and practical feasibility while providing functional correctness.

1.2 Key Dimensions:
The approach we followed was to build the lowest cost system that is functionally correct and then use incremental optimizations to increase the performance to the best possible**.**

a) Functional Correctness: For any system to work, functionality is the key criterion. The router needs to support 40Gbps between every line card and the crossbar cards along with the redundant crossbar pair included. Our primary strategy was to develop an interconnect architecture that linked all line-cards and sustained data rate of 40Gbps along with maintaining the timing margin (clock jitter) and noise margin constraints (BER < 10e-14).

b) Cost: Our next most important goal was to achieve a low cost for the entire design. In order to achieve this, we initiated our design with cheapest option available and then switched to more expensive components to meet the performance requirements. Starting from the choice of plane position (backplane, mid-plane, orthogonal) to connectors (Tinman/Amphenol), amount of bit-slicing, choice of package (BGA vs. flip-chip), and choice of dielectric, everything was optimized in the same manner. Reducing complexity in the wiring due to our orthogonal architecture allowed for cheaper boards with fewer signal layers. Orthogonal connectors enabled high frequency allowing us to use bond wire package, high loss tangent material and fewer data chips to minimize cost.

c) Performance and Reliability: Though low cost was important, we focused on obtaining the maximum data rate for the given cost and thus improved our reliability and BER. Orthogonal routing simplified the routing and enabled high frequency data. Differential signaling with current mode provided performance with minimal cost increment. Seven tap DFE further increased performance. Furthermore, usage of strip-lines over micro strip-lines for PCB stackup reduced our forward crosstalk and complexity to let us have better margins thus allowing for better data-rate. Striplines do not have the problem of varying dielectric constant and hence non-uniform electric fields. Increasing the pitch between wire spacing and employing offset reducing circuitry at receiver end resulted in tighter BER and better overall performance.

1.3 Challenges:
As mentioned in the previous section, the major conflict arises in choosing the lesser expensive option or costlier high performing one. Few such decisions have been mentioned below where we strike the balance between cost and performance.

a) **Choice of Signaling**: Single ended signaling has the advantage of lesser number of pins and lesser cost (including return current pins) but has lower noise immunity and lower performance so we chose differential although our goal is low cost system.
b) **Routing** : Orthogonal planes have a higher cost (more expensive connectors) but give lesser routing complexity, fewer signal layers, cheaper PCB board and also better performance, again making it a challenge to choose and design for low cost.
c) **BER and clock jitter**: Given that we are aiming for maximum bandwidth for a given cost, the margins on maximum RMS noise and clock jitter were very tight. We solved this problem by improving our equalization setup and using better offset nullifying circuitry at the receiver end.

2. System Design Description

Through iterative analysis, we struck the optimal point of operating frequency at **9.375 Gbps** in order to meet the requirements of the project. This meant 6 differential pair per data card chip of each line card to meet the 40Gbps BW requirement and we achieve it with a system costing $**3508.80** and are highly robust as seen in section 5.3.

2.1. Cost Summary

PCB Costs

	# Layers	Tan	Cost/cm²	Card Area	Card Cost	Total Cost
Line Card:	4	0.03	$0.1	40*20=800cm²	$80	$1280.00
XBar Card:	4	0.03	$0.1	48*20=960cm²	$96	$384.00

Package Costs

	Type	#Balls	Pkg Cost	# Pkgs/card	Card Cost	Total Cost
Line Card:	BondWire	128	$10+0.05/pin	2	$32.80	$524.80
XBar Card:	BondWire	768	$10+0.05/pin	1	$48.40	$193.60

Connector Costs

	Type	#Signals	Piece Cost	# Pieces	Card Cost	Total Cost
Line Card:	Orthogonal	32	$0.4 perdiff-pair	4	$25.60	$409.60
XBar Card:	Orthogonal	32	$0.4 perdiff-pair	16	$102.4	$409.60

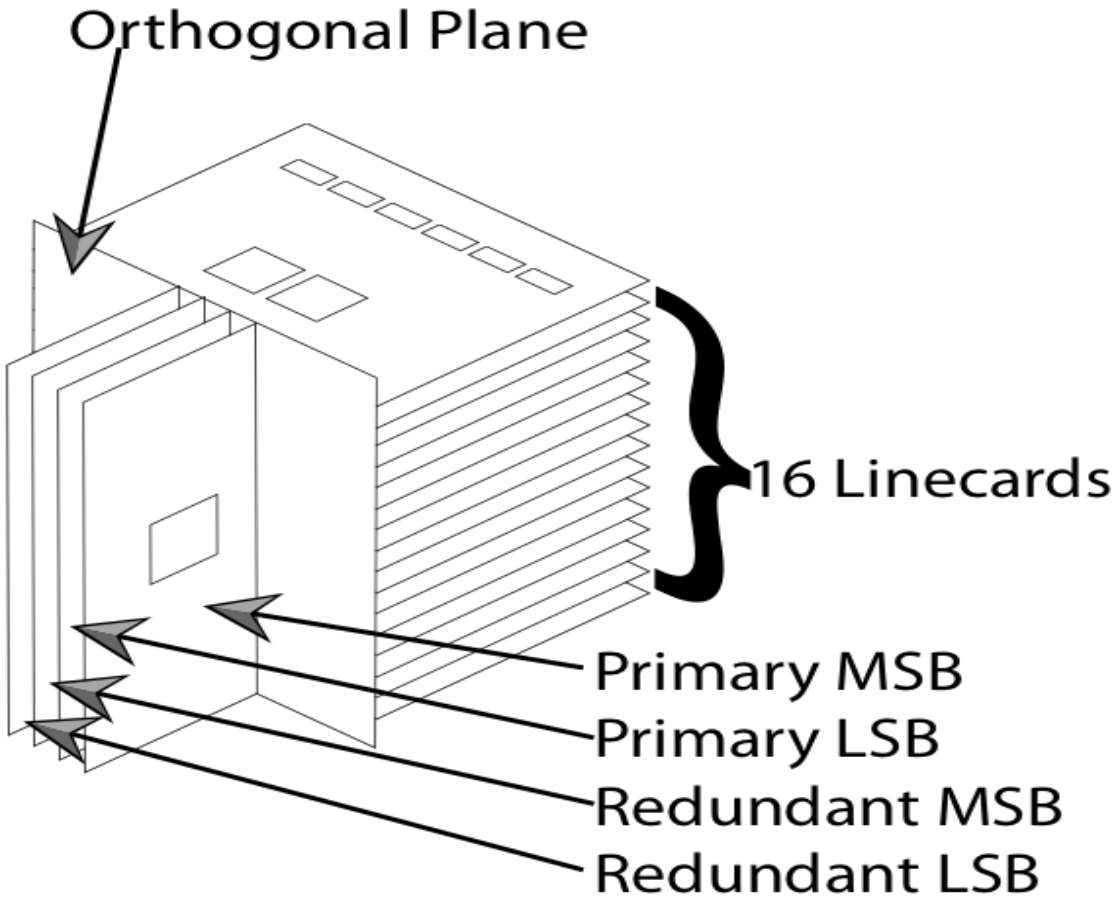
Power Cost

	I_{LINK} (mA)	VDD	Link Pwr	# Links	Pwr Cost	Total Cost
Full System:	25mA	1.2	30mW	512 (16*2*16)	$20 / W	$307.20

Total System Cost: **$__3508.80__**

2.2. Block Diagrams

Our overall system architecture involves connecting 16 line cards through an orthogonal architecture to the primary and redundant crossbar card. Each line card has two data cards with lines operating at a frequency of 9.375 GBps.



Fig(1) 3-D View of our design

2.2.1 Line Card Design:

Each line card needs to achieve a bandwidth of 40Gbps to each of the primary and redundant crossbars. A line card can be bit-sliced into N chips so that the signaling rate on each line is smaller, but this comes at a cost of more wires. Each chip needs to provide a bandwidth of (40*2)/N Gbps of the (40*2)=80Gbps coming from each line-card. We choose N=2 (bit slicing=2) for the following reasons:

a) Our orthogonal architecture implied high connector cost per pin. To optimize for our low cost budget, we desired to minimize the number of Xcede connectors, hence the number of wires and thus chose 2 data chips instead of four or more per line card.

b) Owing to our higher frequency of operation (9.375Gbps instead of 6.25Gbps), minimizing crosstalk interference to improve noise margin demanded more spacing between signal wires. Having a smaller bit slicing allowed us to have larger intra-wire spacing on the board and minimized crosstalk.

Each line card needs to have 4 connector sets (one for each of the two crossbars (primary and redundant) with crossbar slicing of two). Each line card has 2 chip packages that each has an in-out bandwidth if (40*2/2) = 40Gbps.

Per data card chip inout bandwidth = (40 * M) / N

Where, M = 2 = Crossbar redundancy

N = 2 = data cards per line card

Hence, per data card chip inout bandwidth = 40 GBps

The maximum frequency of operation achieved by our design is 9.375GBps. For 8B10B coding, this translates to an effective 9.375/1.25=7.5Gbps data rate. So we need

Ceil (40/7.5) = 6 duplex differential pairs

= 12 simplex differential pairs

= 16 simplex differential pairs (nearest multiple of 8 because we have 4 connectors and 2 way or duplex signaling which means we need a integer multiple of 8 links to be coming out of a chip)

= 32 simplex wires (or pins per chip to all connectors included)

= 8 simplex wires (from each chip to each connector set, to each crossbar card)

= 12 pins on each connector from each chip (1.5x more pins to accommodate power and ground)

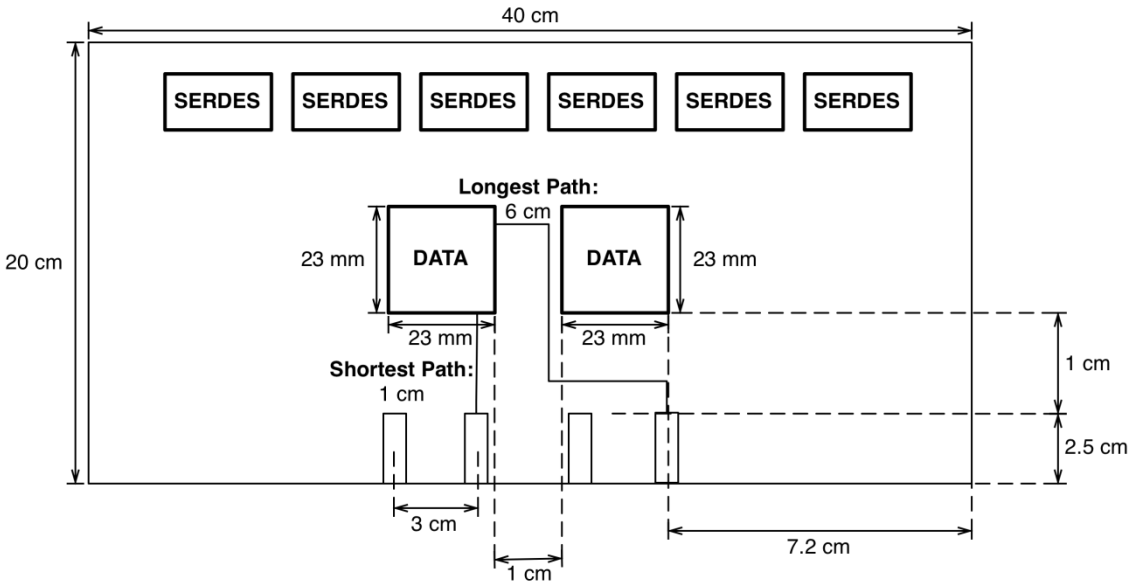


Fig (2) Signal Trace Lengths in Data Line Card

2.2.2 Cross-Bars

There are a total of 4 cross-bars boards after the primary bit slicing. These boards can further be bit-sliced (secondary slicing) to have more cross-bar packages per board but this will increase routing complexity for no obvious benefit in data-rate. Each crossbar board has 16 connector sets to accommodate connections from each of the 16 line cards. The connections are described in the routing section.

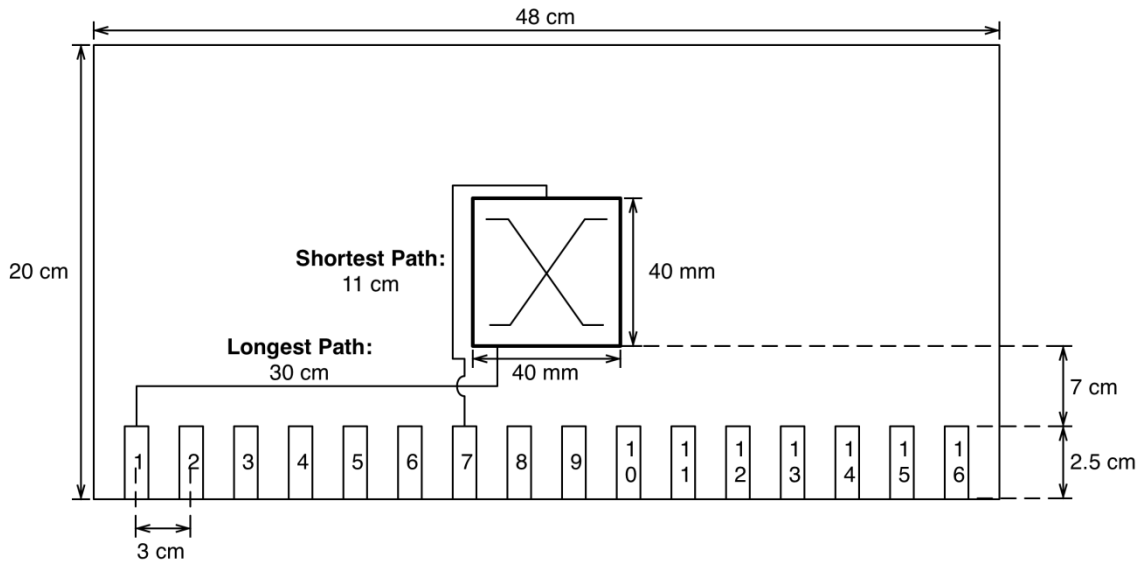


Fig (3) Signal Trace Lengths in Cross Bar Card

2.2.3 Summary of Calculations

Number of line cards	16
Number of data cards per line card	2
Number of crossbar chips	1
Number of simplex wires per data card	32
Number of simplex wires to each crossbar chip	8
Total number of simplex wires coming out of all line cards	1024
Total number of simplex wires reaching a single crossbar chip	256

2.3. **Placement Drawing**

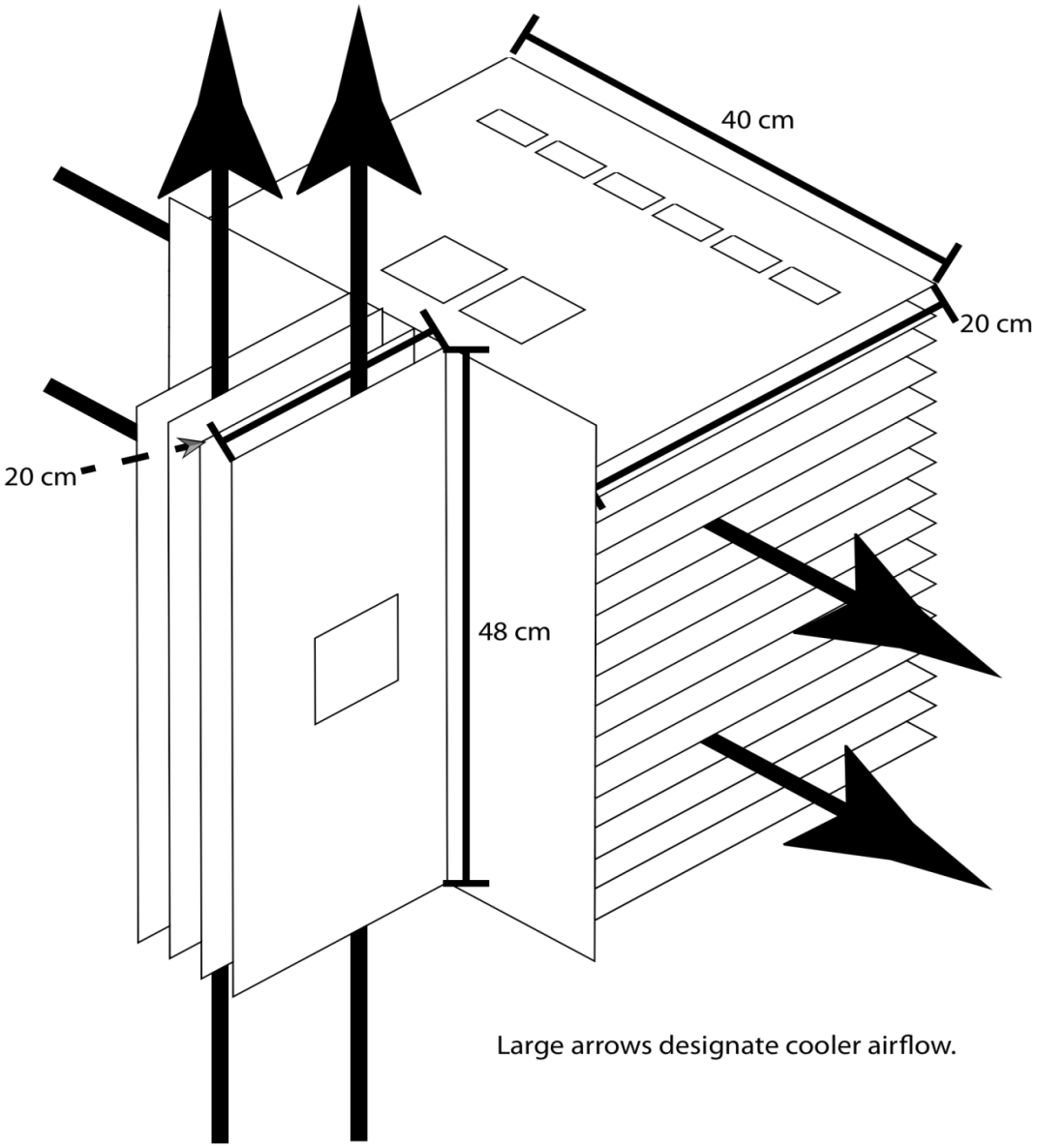


Fig (4) Placement drawing with airflow direction

We used an orthogonal connecting plane to directly link the line-cards to the cross-bar boards. Though the connectors are slightly more expensive, the orthogonal architecture eliminates the need for backplane or mid-plane related complex routing. Hence, the wiring is much simpler and feasible with fewer layers and a cheaper board. Owing to the elimination of long backplane wires and stubs, we can achieve a much higher data rate and can easily migrate to future generations of higher data rates. As the connector model is plug-in type with least routing on the support plane, the mechanical reliability of the system increases tremendously. All components (data cards and crossbar chips) are independent and any single failure can be sustained by replacing the corresponding card. There is no tension of the support plane crashing and the whole system to be replaced, as ours is a plug-in model. The block diagram shows a schematic of this arrangement and also the flow of air from the two cooler systems (orthogonal: bottom to top and side to side).

2.4. **Signal/Connector Mapping**

A diagram illustrating how signals map to our connectors can be found below.

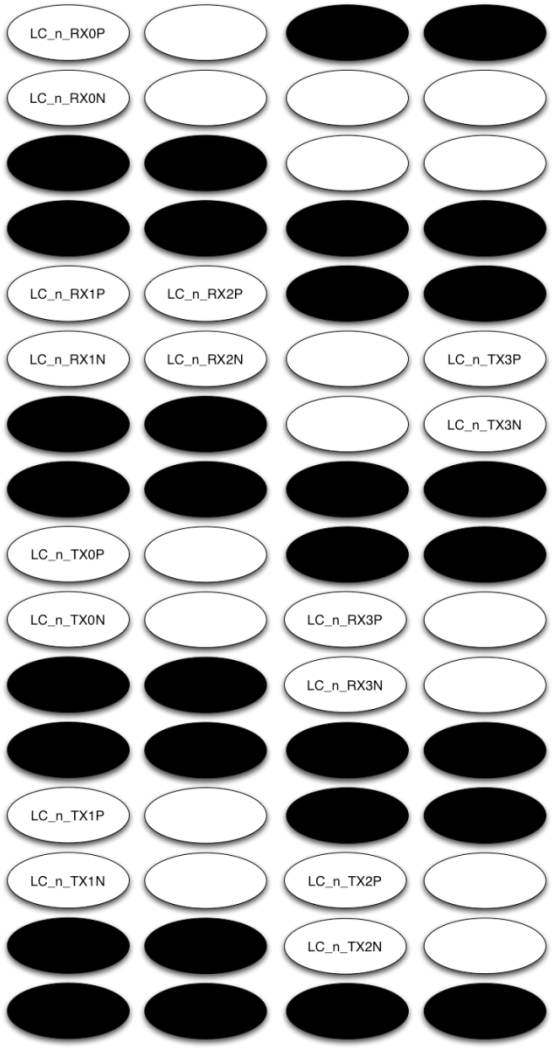


Fig (5) Signal to Connector Mapping

Since we are using special orthogonal connectors and do not perform any routing on the midplane, this is the pinout on both our crossbars and on our linecards.

In our naming scheme, signals take the form:

LC_[15:0]_*d*[3:0]*p*

d denotes direction, and *p* denotes polarity. A signal going from linecard to crossbar would have direction "TX", while the opposite would have direction "RX". Signals travelling to/from the redundant crossbar take the suffix "R", as in "RXR" or "TXR". Polarity denotes either the negative ("N") or positive ("P") side of a differential pair.

2.5. **Routing Study Drawings**

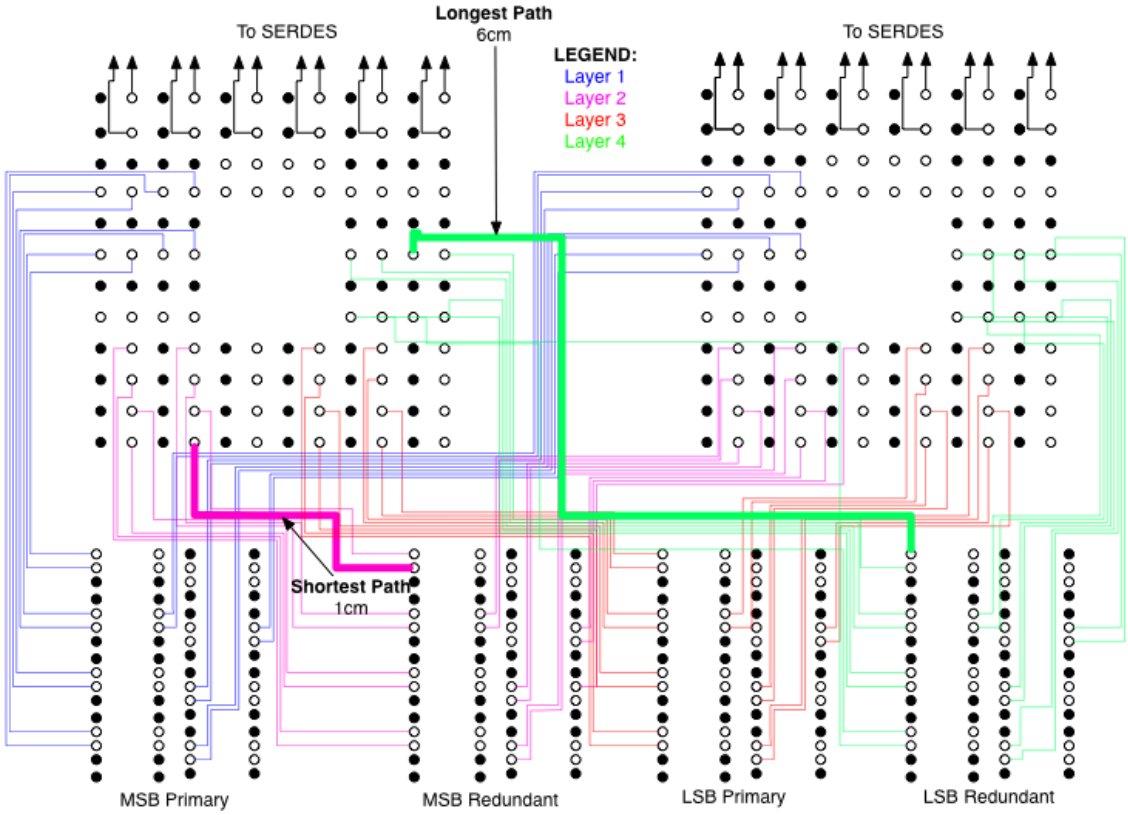


Fig (6) Routing diagram for line card

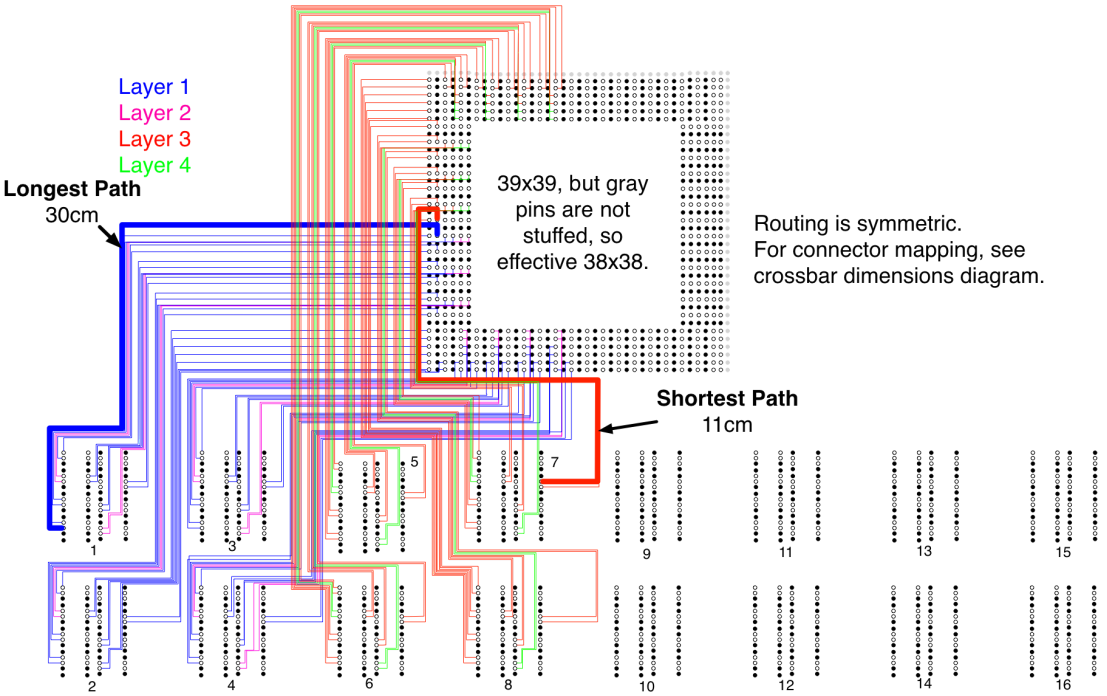


Fig (7) Routing diagram for cross bar card

Our orthogonal architecture eliminates the need for routing signal traces on the mid-plane.

2.6. **Design Discussion**

As mentioned in the section 1, our primary goal was to get a functionally correct design with maximum performance for the lowest possible cost. The following flowchart reflects our thought procedure.

- We decided to start with orthogonal architecture as that has the least routing complexity and higher frequency, though it requires more expensive components.
- As orthogonal architecture has high cost for Xcede connectors and high data rate, we wanted least number of connections. Hence we fixed on 2 data chips per line card and crossbar bit slicing of two.
- As orthogonal was already on the expensive side and had facilitated high bit rate, we decided to start with bond wire package and Xcede connectors (back to back).
- We modeled the source and termination resistance mismatches, along with 5% mismatch in driver amplitude. We realized receiver offset was consuming lot of margin and hence included a circuitry to reduce the offset value as discussed in later section.
- With equalization, we got an amazing eye at 6.25 GBps and slowly enhanced our frequency to 12.5 GBps. As the frequency increased, the need for equalization was evident. So, we ramped up our model to include DFE equalization which gave a decent eye opening at 12.5 GBps

- Here came our main confusion:
  - ⇨ 12.5 GBps halved the number of connections, hence drastically reduced the package sizes, the cost of connectors. But, it induced more cross talk and very low robustness as the eye was just small enough to accommodate the margins specified by the project. Furthermore, it was steep and diamond shaped and just satiating the requirements for the contemporary time, not for future developement.
  - ⇨ 9.375 GBps had more wiring and connections, but was less stressing on cross talk modeling and had higher margin (noise and time sensitivity) to satiate the need for future generations.
- As we desired to obtain optimal balance between cost, performance, robustness and future application, we proceeded with 4 corner testing of system at 9.375 GBps. Next procedure was routing which was greatly simplified by orthogonal architecture.

- There were few issues with routing for line and crossbar chips as mentioned below:
  - ⇨ The crossbar card plane had to be around 48cm long to fit in 16 line cards. But we wanted to avoid secondary bit slicing to minimize cost of connectors, hence the placement of crossbar package and routing it symmetrically was a major issue.
  - ⇨ Also, crosstalk was evident at such a high frequency, hence spacing had to be increased and also differential pairs routed together to combat the effects.
  - ⇨ Placement and position of line cards symmetrically needed efforts to minimize wire lengths on the line card side.

3. Signaling Description

3.1. Link Block Diagram

The diagram clearly indicates the generic drawing of our line card to cross bard card signaling with special mention to orthogonal back to back connector.

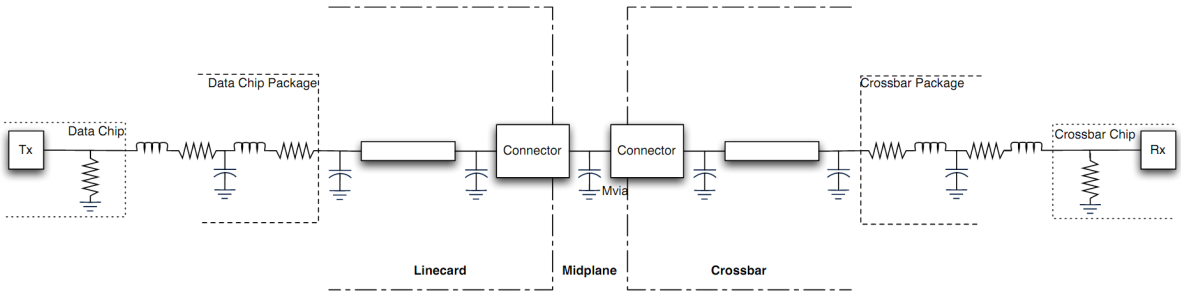


Fig (8) Link Block Diagram

3.2. Signaling Convention

3.2.1 **Unidirectional Differential Signaling**:

The primary reason is that differential signals use lower voltage swings than single-ended signals as the differential receiver threshold value is better controlled than that of a single ended one. The lower swing leads to faster circuits and thus reduces our power consumption. Differential signaling also reduces EMI, since the opposite currents carried on the two traces cancel the electric and magnetic fields at large distances. Similarly, differential signals are less sensitive to crosstalk. Furthermore, we use 8B10B coding embedded with 80% signaling efficiency. 8B10B has simpler PRBS scheme than 64B66B coding and thus convenient to design.

3.2.2 **Current mode driver and receiver:**

Employing current mode signaling in our design facilitates us with the advantage of high output impedance, which translates to crosstalk immunity in the receiver and therefore higher SNR. Also low output voltage swing in current mode leads to higher rates.

3.2.3 **Receiver offset reduction technique**

Reducing the receiver offset is a great gain for the system's noise margin (BER) and robustness for the operating data rate. We used the following receiver circuitry (taken from "Differential Analog Data Path DC Offset Calibration Methods" by T. Asoda and H. Andoh) to achieve this:

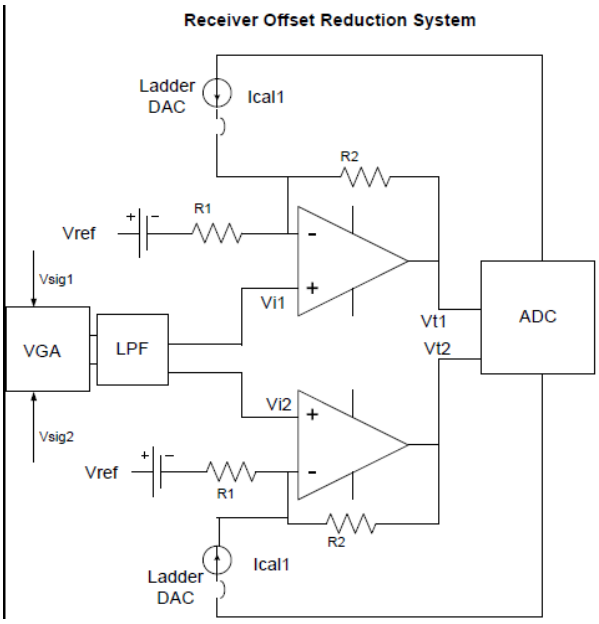


Fig (9) Receiver Offset Circuitry

Functioning:

The input signal V_{sig1} and V_{sig2} are first passed through pair of pass transistors, a variable gain amplifier (set to 1 here but can be adjusted for degraded signal amplitude) and a low pass filter to contain noise. The outputs V_{i1} and V_{i2} are passed to the offset cancellation loop circuit. The basis of operation is to produce two offset cancelling currents $Ical_1$ and $Ical_2$ in the above figure. The trick is to sense the output of the Opamps 1 and 2 (Vt_1 and Vt_2), which are fed-back as a voltage difference to an ADC for sampling. The output digital bits of the ADC control the values of $Ical_1$ and $Ical_2$ using negative feedback and reduce the DC offset in the differential pair as shown by equations below. Writing down the equations for Opamps 1 and 2 (by using virtual short) in the above figure,

$$\frac{V_{ref} - V_{in1,2}}{R_1} = -I_{cal1,2} + \frac{V_{in1,2} - V_{t1,2}}{R_2} \quad (1)$$

$$V_{i1,2} = \frac{R_1}{R_1 + R_2}(V_{t1,2} + I_{cal1,2}R_2 + V_{ref}\frac{R_2}{R_1}) \quad (2)$$

$$V_{off-DM} = V_{i1} - V_{i2} = \frac{R_1 R_2}{R_1 + R_2}(I_{cal1} - I_{cal2}) \quad (3)$$

$$\text{for the condition } V_{t1} = V_{t2} \quad (4)$$

$$V_{off-CM} = \frac{V_{i1} + V_{i2}}{2} = \frac{R_1 R_2}{R_1 + R_2}(\frac{I_{cal1} + I_{cal2}}{2}) \quad (5)$$

By setting $R_2 = 200\Omega$ and $R_1 = 100\Omega$ with a ADC level of n=6, $\delta I_{cal} = 500uA$, we can cancel close to 33 mV of offset (obtained by maximizing equation 3 on both sides; In other words, evaluating the condition on maximum input differential offset that can still make V_{t1} and V_{t2} equal for some feasible value of I_{cal})

Thus, as is evident we can employ this circuit and reduce our receiver offset to 7mV from 40mV. Due to the practical issues of analog circuitry and worst corner analysis, we have assumed the conservative approach here, as the paper claim even further reduction possibility.

3.2.4 **Equalization with DFE**

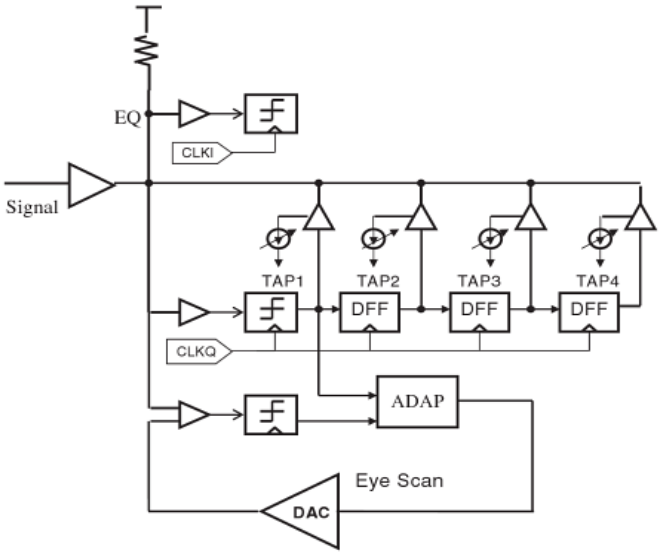


Fig (10) DFE Circuitry

Line equalization is important in serial links to alter the frequency response of the line and achieve better signal recoverability at the receiver. We used the given feed-forward equalizer and an optimizing procedure to set tap weights. Based on the reference of DFE (Analog Circuit Design, Michiel Steyaert, Arthur H. M. van Roermund, Herman Casier), we attempted to model hybrid between FFE and DFE. Instead of modeling the feedback in spice circuitry, we performed a mathematically modeled script to emulate 7 tap DFE. The post and precursor weights modulate the delayed version of the pulse response of the signal and are summed to obtain a signal that is close to the ideal pulse response. We used a sampling rate that is higher than the bit-rate to construct a Topelitz matrix and used min ||Tw-O|| to optimize the tap-weights. We used the initial estimates obtained by the above procedure to set the weights and then tuned each parameter iteratively one after another until an eye with better eye-margins is obtained.

The optimization script used is attached at the end of this report. This script worked within Matlab and used the CVX and HspiceTools toolkits. As discussed above, this script worked by importing a pulse response from an Hspice simulation where the simulation was run with sampling frequency significantly above the line rate. Once this data was brought in, we shifted this data in time in multiple ways to construct a Topelitz matrix. We also created an ideal pulse response to optimize against. After this, we used CVX to find optimal equalization coefficients by minimizing the norm of the coefficients multiplied by the Topelitz matrix minus the ideal pulse (minimizing the norm of the error) subject to the constraints that the coefficients must be positive and sum to one.

3.2.5 Termination Scheme

We have used on-die termination (resistor placed inside the package) in the receiver to match the transmission line impedance. In order to account for the termination resistance variations, we have used worst case analysis of impedance mismatches to obtain realistic lines and termination (This analysis is pessimistic in determining the max rate of operation) by running the simulations for all the 4 corners of source and receiver terminations (source resistor inside the tap element).

3.3. Timing Conventions

We use per-bit timing in the receiver to recover clock and the source is driven by a PRBS7 signal (as given in template). We use the CDR circuit that has been discussed in class as shown below: the system is a Delay-Locked loop (DLL) that consists of a phase comparator, variable delay unit and a loop filter. All the boards have a generated reference clock. These clocks are generated per board using an on-die oscillator that can be distributed using H-tree within the board to modules. The input clock is time-shifted using the delay unit and then compared against the reference clock signal. The error signal is fed back to the delay unit to adjust the phase difference to a small value. The loop filter stabilizes response by cancelling high-frequency noise in the loop.

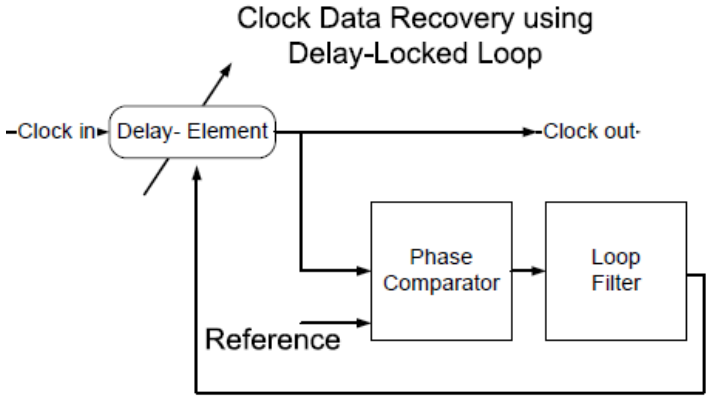


Fig (11) CDR circuitry

4. Interconnect Description

This section should describe in detail the transmission lines and connectors used for interconnection. Describe the PCB traces used in each section of the system: line card, crossbar and backplane. Include the following.

4.1. PCB Stack-ups

4.1.1 Choice of striplines

There are multiple options to realize the PCB copper trace lines physically on board. Micro-striplines have ease of manufacturability, low cost and also smaller space requirements but offers less performance due to cross talk susceptibility owing to EM transmission in a non-homogeneous medium. Due to this, we chose strip-lines for our PCB stackup.

4.1.2 Choice of Dielectric

Following our low cost system design, we decided to use a basic PCB board material with FR4 dielectric with $\tan\delta=0.03$. Owing to our implementation of orthogonal architecture and high frequency and hence lesser routing, we were still able to achieve high frequency using this cheap board.

4.1.3 PCB Stackup description

As seen in the routing figures, both line-card and the crossbar boards need 4 signal layers for transmission and so we needed to build a PCB stackup that has 4 layers of stripline signals. The figure below shows the stackup we used in the design and the overall height is 81.2mil. Because our routing was orthogonal, the need for backplane routing and backplane PCB trace is completely eliminated that further reduced our cost of design.

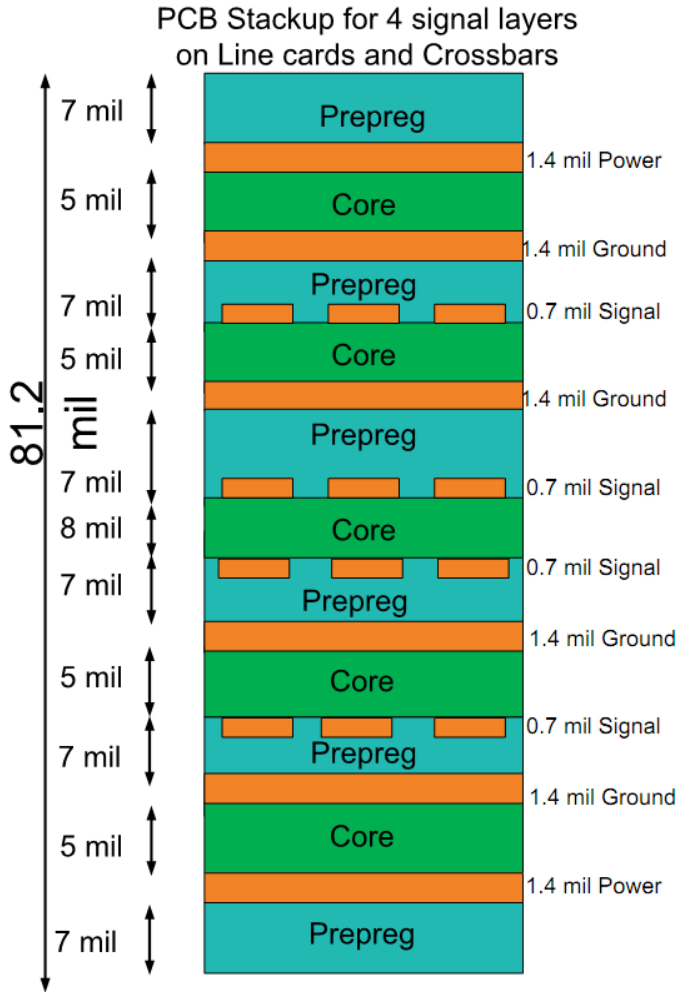


Fig (12) PCB Stack Diagram
for Line Cards and
Cross Bar Cards

Note: The width of the centermost core has been increased from 5 mils to 8 mil to mitigate the effect of crosstalk between the adjacent signal layers

4.2. Signal Trace Descriptions

In our design, we employ PCB traces at least 6mil wide with adequate spacing enough to match the 50 ohm impedance limit. Our maximum hole aspect ratio is maintained at 10 and we need back drilling to connect the plug in connectors of orthogonal system. Between two wires of a differential pair, we maintain a spacing of 4W and 8W between two differential pairs; higher spacing is needed for higher frequencies.

Forward Cross talk coefficient = 0 (as we use striplines)

Backward Cross talk coefficient = $(k_c+k_m)/4 = (L_{12}/L_{11} + C_{12}/C_{11})/4$

$= -5.04$ e-4 = -0.05% < 0.1%

This is included in our spice channel simulations.

4.3. Connectors and Cables

Type of Connector: Orthogonal
Manufacturer and Model: Amphenol Crossbow (Spice modeled by two back-back Xcede connectors)
Size: 4 x 4, minimum 24mm pitch between connectors
Impedance: 100Ω ± 5%
Crosstalk: Datasheet lists multi-line crosstalk as <1.5%

4.4. Integrated Circuit Packages

Line Card → Data Chip
Package Used: BGA with **Bond-wire** package
Sizes: 13x13, All pins stuffed, 169 pins

Crossbar
Package Used: BGA with **Bond-wire** package
Sizes: 39x39, 38x38 outside 6 rows/columns stuffed, 768 pins

5. Design Analysis

5.1. SPICE Simulation

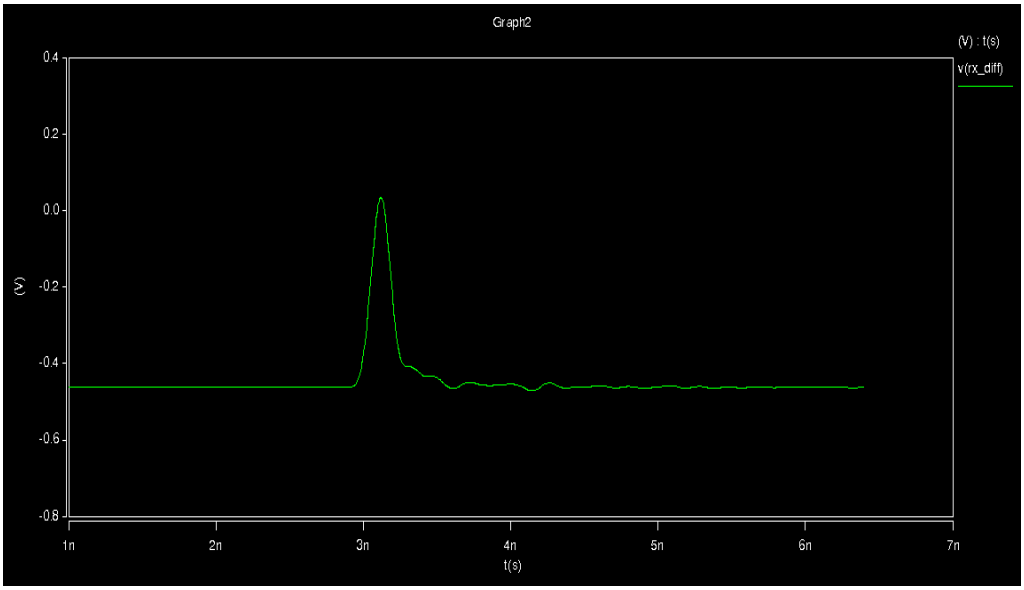


Fig (13) Before Equalization

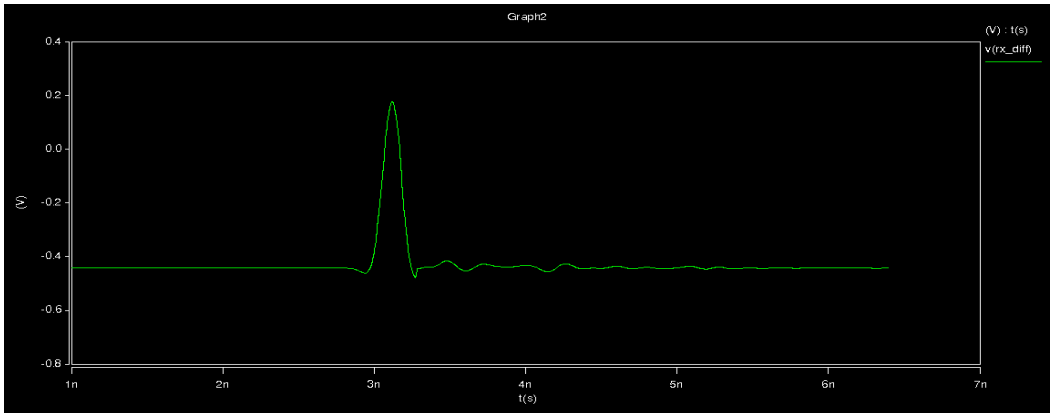


Fig (14) After Equalization

(much faster settling time and reduced front porch)

The main taps setting among the equalizer have been indicated below:

Pre-cursor: 0.1, 1st post-cursor: 0.1, 2nd post-cursor: 0.1

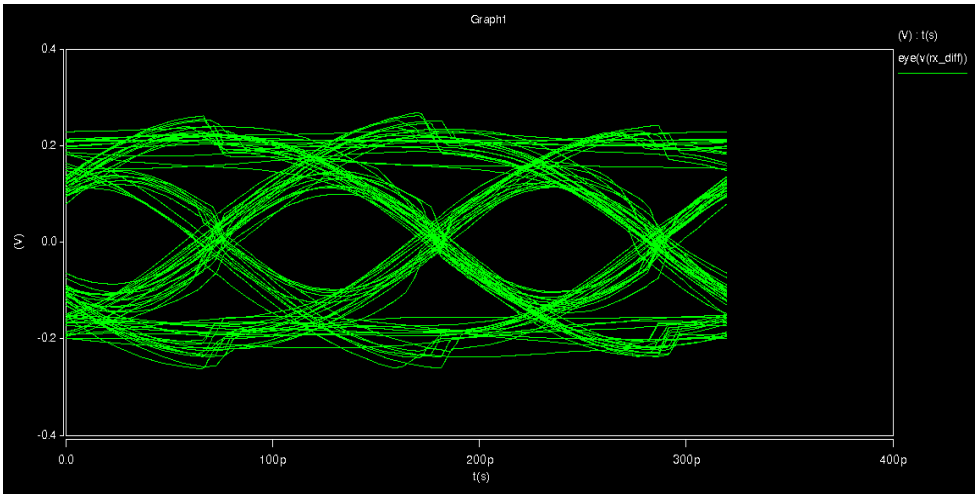


Fig (15) General Big Picture

Note: As the aggressor frequency is not an integral multiple of the operating data rate, hence all eyes are not symmetric and so the worst one is chosen for noise margin and timing margin analysis.

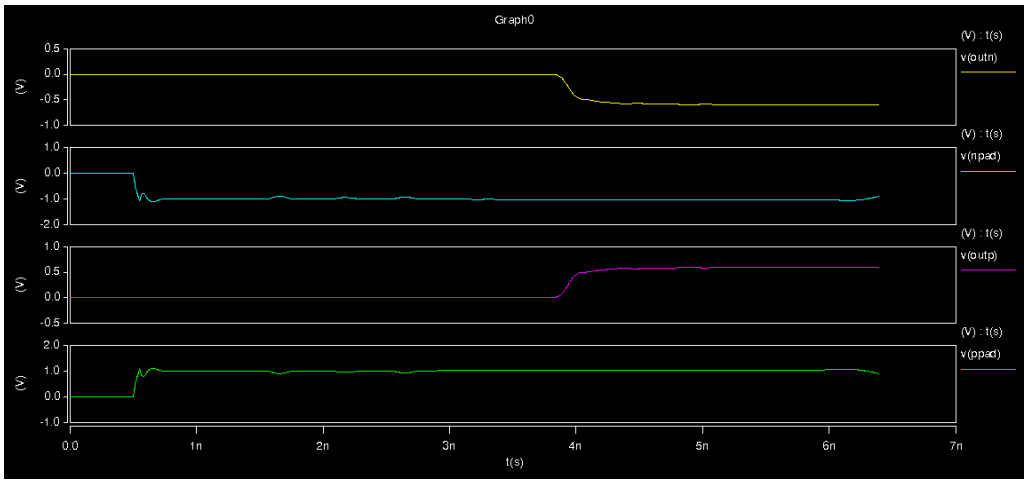


Fig (16) TDR and TDT response

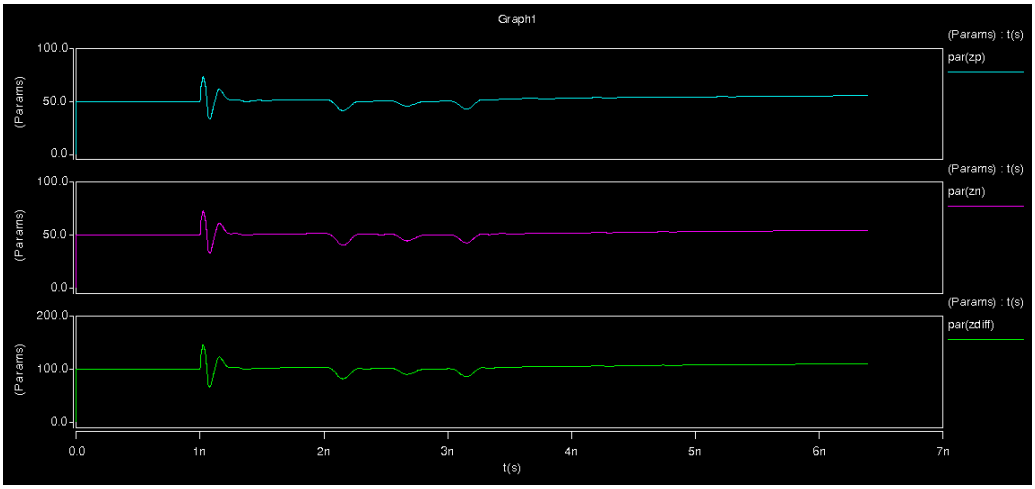


Fig (17) TDR and TDT simulations for impedance

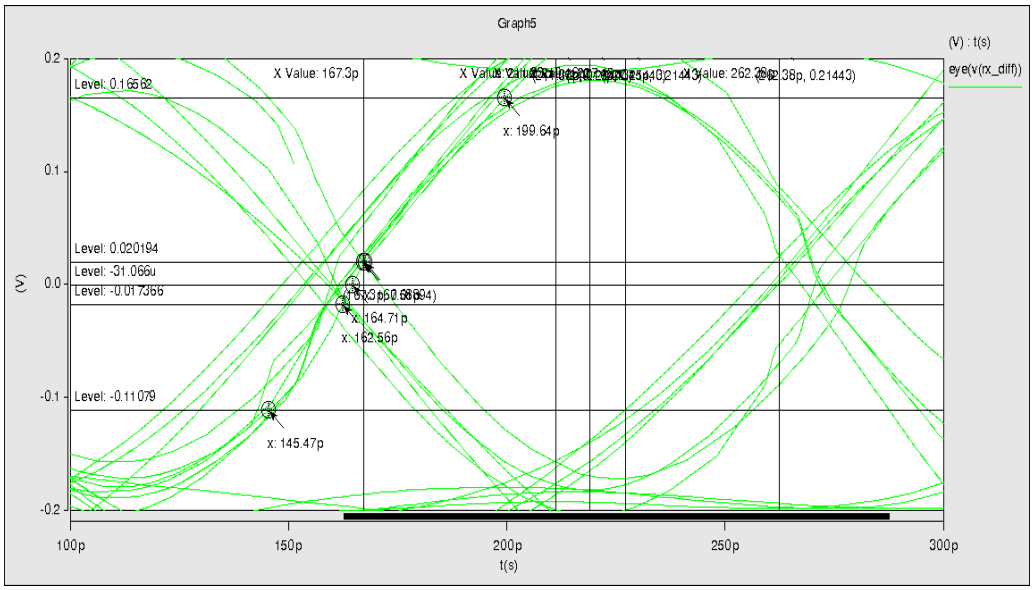


Fig (18) Simulation for shortest trace length

Note: The minimum margins for noise (34 mV vertical) and timing margin (20 % unit interval) are well fit in. Along with it, the 30% CDR uncertainty range on each side is also more than enough compensated. Thus, our eye for shortest trace length, not only satiates the necessary margin requirements, but also is pretty robust for random noise and jitter, so as to apply for future generations.

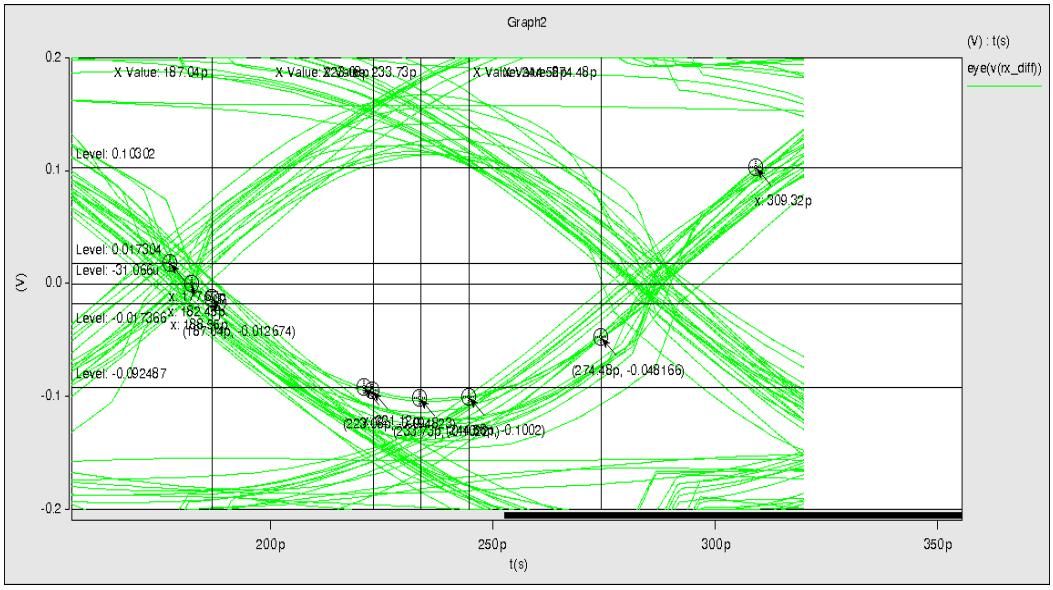


Fig (19) Simulations for Longest Trace Length

Note: The minimum margins for noise (34 mV vertical) and timing margin (20 % unit interval) are well fit in. Along with it, the 30% CDR uncertainty range on each side is also more than enough compensated. Thus, our eye for longest trace length, not only satiates the necessary margin requirements, but also is pretty robust for random noise and jitter, so as to apply for future generations.

5.1.1 **Estimates of Path Lengths:**

Orthogonal architecture had clearly indicated that the backplane wire routing was nullified. The biggest trouble was the length of crossbar card plane that peaked up to 48cm owing to accommodation of 16 line cards each spaced at 3cm width. Hence, the longest wire would be roughly half of this figuring to 24cm. We conservatively approximated it to 10” and the minimum one to around 1” (3cm), assuming the biggest possible package sizing and double folding the path for minimum wiring. On the data card side, it was much easier as there were just two data chips. Approximating the Xcede connectors width to be 1” and the crossbar cards to be spaced at the same spacing as data-line cards, i.e. roughly 1” spacing, we figured out the maximum length would be 3” (left most chip connects to rightmost crossbar chip) and minimum would be around 1”. We proceeded with these values in Spice modeling.

5.1.2 **System Modeling:**

As mentioned in the section 2.6, we have modeled the packages (bond wire), connectors (back to back Xcede for orthogonal modeling), daughter card and backplane

via capacitances in our Spice deck. We modeled the cross talk noise and worst cases too in our model as explained in the following sections.

Equalization:

Once we set the package and connector modeling, the eye was distorted without equalization. We estimated the tap weights in order to eliminate the front and back porch by iterative analysis. While the pre cursor played a main role in attenuating the front porch, the first post cursor was the main element in reducing the ISI at the receiver. Eventually the DFE script and feedback loop helped us in fine-tuning the seven taps, with main importance to the first post cursor tap.

5.1.3 Worst Case Analysis

Once we had fixed on the bond wire package, Xcede connectors, FR4 dielectric with loss tangent of 0.03, we decided to include the mismatch modeling. We performed 4 corner analysis on the following parameters:

- **Source Termination**: To account for 10% mismatch, we changed the rsource modeled in source end to 45 to 55 ohms
- **Receiver Termination**: Similarly, we varied the rterm from 45ohm to 55ohm to model the 10% mismatch.
- **Driver Amplitude**: Since the current mode source is modeled using the VCVS from the voltage input, we vary 'vd' from 593.75mV to 656.25 mV to account for the 5% mismatch.
- **Crosstalk modeling**: We modeled the cross talk by placing the signal differential pair between two aggressor signals operating at the same voltage amplitude, but at non integral multiple of operating frequency. This saves us the trouble of over estimating crosstalk and adding to our system's model.

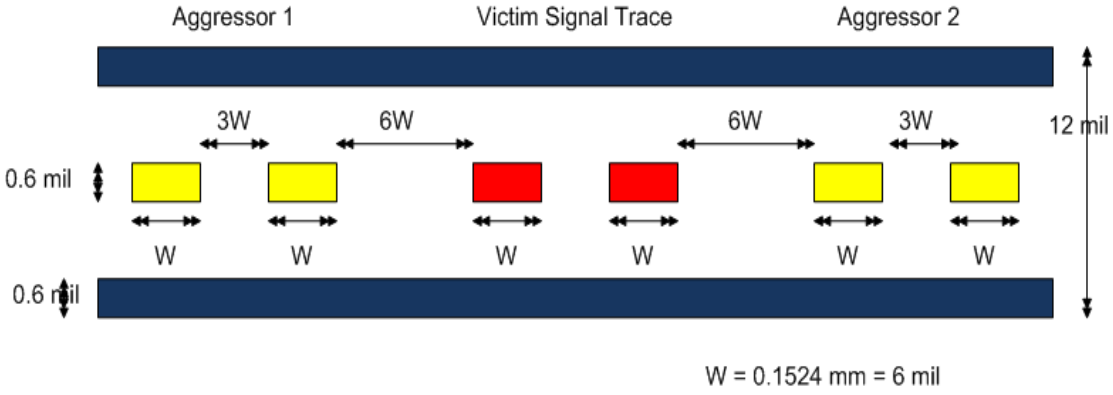


Fig (20) W element stripline model
for crosstalk simulation

5.1.4 Constraints:

Every iteration when we made a choice to use any component based on cost, we validated if our choice affected our eye by checking the following constraints. The same constraints helped us decide whether to operate at 12.5GBps and lose on noise margin and robustness, or on the contrary work at 9.375GBps for a better noise immune robust system (as discussed in section 2.6).

⇨ **Noise Margin**:

The problem statement specifies receiver offset of 10 mV and sensitivity of 40mV, thus implying at least a minimum height of 50 mV on each side, or 100mV total opening for zero noise margin case. But, as per our discussion in section 3.2.3, we could reduce the receiver offset to around 7mV. Hence, the minimum height needed was 17 mV on each side or around 34mV in both directions.

⇨ **Timing Margin:**

The problem statement indicates positional tolerance of 10% bit time along with 30% CDR uncertainty, amounting to a total of 40% on each side. The eye width was checked accordingly to avoid any violation problems.

⇨ **Crosstalk Margin:**

Owing to out modeling of crosstalk with aggressor signal operating at non integer multiple of original data rate, we ensured that we checked the simulation for more than 1 cycle and we performed our analysis for the worst case.

5.2. Noise and Jitter

5.2.1 Noise Sources

5.2.1.1. Included in SPICE

Noise Source	Modeled value
Transmitter offset	5%
Source termination	10%
Receiver termination	10%
ISI	Included
Crosstalk	Included in strip-line model
Receiver Offset/Sensitivity	34mV
Power supply noise	Differential current mode signaling eliminates the receiver crosstalk and other common mode noise issues

Assumption: The differential amplifier's current tail source helps eliminating the common mode noise. Also, SSO noise has been modeled by the 2:1 ratio in Power-GND: Signal wires design in our routing.

5.2.1.2. Not modeled in SPICE

The fixed noise sources (proportional and non-proportional) have been included in our spice model, but the statistical random noise sources and RMS noise values have not been included. Their allowable magnitude can be estimated from the noise budgeting in the following section.

5.2.2 Timing Noise Sources

5.2.2.1 Included in SPICE

The CDR tolerance has been verified after the running of SPICE modeling. Each simulation was verified with the minimum eye width required as calculated in section 5.1.4. This accounts for the positional tolerance of 10%, CDR uncertainty of 30%. As our system involves no pipelining, hence we did not need any synchronization and hence noise issues concerning them.

5.2.2.2 Not modeled in SPICE

Similar to section 5.2.1.2, clock jitter could not be accounted into spice model owing to its random nature. But, it has been accounted for in the BER calculation of timing margin in the following section. The following sources produce jitter, which we have not accounted for in here:

- ⇨ Asymmetric and layout mismatches dual strip line design
- ⇨ Jitter and high frequency phase noise in reference clocks generators
- ⇨ Transmission line length mismatch between differential pairs
- ⇨ Power supply noise and other voltage noise which translate to jitter at receiver's end.

5.3 Bit Error Rates

To get BER of 10e-14, we need SNR of 7.65063 (as SNR/sqrt2 comes out to be 5.68)

From the longest trace length simulation,

Noise Margin, Vnm = 75.121 mV

Hence, Maximum random noise that can be tolerated is:

sigma = 75.121 /7.65 = **9.82mV** (rms value)

Similarly for the timing margin:

From the worst case of the longest trace length simulation,

Vnm = 30.42 ps = 28.52 % of bps

Hence, Maximum jitter tolerate by system is:

Sigma = 28.52 % / 7.65 = **3.72% (bit time) = 3.98 ps**

5.4 Power Dissipation and Cost

Power Analysis: There are a total of 32 differential links per each line card. There are a total of 16 line cards => Total of 16*32 links and assuming 25mA (max current per link) => a total IO power of 12.8W.

Cost Analysis: The total cost of the system is **$3508.80** (The breakup of the individual cost components has been analyzed already in section 2).

6 Conclusion

Our design performed well and easily met the requested specifications (640Gbps, BER < 1E-14). We were pleased with our design choices and found that an orthogonal architecture is more complex but has higher performance (as well as easier design) due to the lack of a backplane that needs to be routed. Since we did not need to route in the backplane, we were able to reduce backplane costs from having a simpler (fewer layers needed → reduced PCB cost) and smaller (backplane would need to be taller if it was not orthogonal) mid/backplane. Additionally, since we were able to remove backplane routing, we did not experience any of the extremely long wires that can be found in a large backplane. That being said, we did trade long backplane wires for large crossbar cards and long wires on our crossbar. Currently, our crossbar is able to run at a speed of up to 960Gbps due to our use of 9.375Gbps (7.5 Gbps effective) lines with two Tx and Rx pairs between each crossbar and each datachip.

From our current perspective, if we needed to increase performance, we would probably add more signal lines or work on improving our equalization. Currently, we cannot directly increase effective bandwidth to 10 Gbps (which would provide 1.28 Tbps), but we are close to being able to achieve a sufficiently open eye. If we optimized our equalizer, this would provide sufficient eye headroom to increase the bitrate of our system. We could also increase the performance of our system simply by adding links. While we currently use 12 differential links between each data chip and each crossbar, we were able to successfully route 16 a design with 16 differential links between each data chip and each crossbar. While this would increase system cost (increased package cost, power, and etc.) this would quickly provide 1.5x bandwidth. The combination of these two maneuvers would easily be enough to increase the bandwidth of our router beyond 1.28Tbps.

One thing we had hoped to do a better job at was the optimization of our equalization circuitry. We created a Matlab/CVX analysis script (attached) to perform a linear optimization on the equalization circuitry, which would have theoretically provided us with significantly better equalization. Unfortunately, this script did not work properly, and we were forced to find our own equalization coefficients. This would have greatly improved our speed of work (testing different coefficients is time consuming) and the quality of our results.

7 Acknowledgements

We are extremely thankful to Professor Jim Weaver for guiding us at every step and phase of the project, for those patient long meetings and his valuable suggestions. Also, special thanks to Priya Chawla for the review sessions which helped us in building our spice model. We would like to thank to Pete Stevenson and Piyush Keshri for their useful discussions and feedback on DFE equalizer.

8 Spice Deck

Project.sp

```
*Project Channel Template
**************************************************************************
**************************************************************************
*      The following code implements a simpified channel for the         *
*      design project                                                                                    *
*                                                *
*      The signal rx_diff gives the differential receive signal.   *
*                                                *
**************************************************************************
**************************************************************************
**************************************************************************
*                              Parameter Definitions                                  *
**************************************************************************
* Simulation Run Time *
 .PARAM simtime            = '60/bps'   * Use/adjust for single pulse response
 *.PARAM simtime           = '256/bps' * Use/adjust for and eye diagram
```

```
* Driver Pre-emphasis Levels - ALTER AS REQUIRED BY YOUR DESIGN *
.PARAM pre1 = 0.00 * Driver pre-cursor pre-emphasis
.PARAM post1 = 0.00 * Driver 1st post-cursor pre-emphasis
.PARAM post2 = 0.00 * Driver 2nd post-cursor pre-emphasis
.PARAM post3 = 0.00 * Driver 2nd post-cursor pre-emphasis
.PARAM post4 = 0.00 * Driver 2nd post-cursor pre-emphasis
.PARAM post5 = 0.00 * Driver 2nd post-cursor pre-emphasis
.PARAM pre1feed = 0.00 * DFE pre-cursor pre-emphasis
.PARAM post1feed = 0.00 * DFE 1st post-cursor pre-emphasis
.PARAM post2feed = 0.00 * DFe 2nd post-cursor pre-emphasis
.PARAM post3feed = 0.00 * DFE 2nd post-cursor pre-emphasis
.PARAM post4feed = 0.00 * DFE 2nd post-cursor pre-emphasis
.PARAM post5feed = 0.00 * DFE 2nd post-cursor pre-emphasis

* Driver Voltage and Timing - ALTER AS REQUIRED BY YOUR DESIGN *
 .PARAM vd     = 593.75m * Driver peak to peak drive voltage, volts, including 5% misatch
 .PARAM trise  = 60p                 * Driver rise time, seconds
 .PARAM tfall  = 60p                 * Driver fall time, seconds
 .PARAM bps    = 9.375g              * Bit rate, bits per second

* PCB Line Lengths - ALTER AS REQUIRED BY YOUR DESIGN *
 .PARAM len1   = 3                   * Line segment 1 length, inches
 .PARAM len3   = 9                   * Line segment 3 length, inches

* Backplane Connector Via Parameters - ALTER TO REFLECT YOUR DESIGN *
* If you plan to back-drill, then leave len2via set to 0.03
* If you don't back-drill, then set len1via & len2via to simulate
* one trace nearest and one farthest from the connector
*
 .PARAM len1via = 0.1                * Active via length, inches
 .PARAM len2via = 0.03               * Via stub length, inches

* Receiver Parameters - ALTER AS REQUIRED BY YOUR DESIGN *
 .PARAM cload = 2p                   * Receiver input capacitance, farads Not used as included in C_esd of package
 .PARAM rterm = 45                   * Receiver input resistance, ohms including 10 % mismatch

**************************************************************************
*                                                                        *
*                          Signal Source & Driver                        *
*                                                                        *
**************************************************************************
* Single Pulse Signal Source *
Vs  inp 0    PULSE (1 0 0 trise tfall '(1/bps)-trise' simtime)

* PRBS7 Signal Source - REPLACE AS NEEDED *
 *Xs  inp inn  (bitpattern) dc0=0 dc1=1 baud='1/bps' latency=0 tr=trise

* Behavioral Current Mode Driver with Pre-emphasis - REPLACE AS REQ'D        *

*    1 pre-cursore & 2 psot-cursor taps                                     *
 Xf  inp in   (RCF) TDFLT='0.25*trise'
 Xd in ppad npad (tx_4tap_diff) ppo=vd bps=bps a0=pre1 a2=post1 a3=post2 a4=post3 a5=post4 a6=post5

* Rsource variation  has been accounted in the tc_4tap_diff file
Vprobe jp1 jp1t 0
**************************************************************************
*                                                                        *
*                  Interconnect - ALTERED TO REFLECT OUR CHANNEL         *
*                                                                        *
**************************************************************************
* True Interconnect *
 Xpp1    ppad  jp1   (bondwire_pkg)                          * Driver package model
 Xvp1    jp1t  jp2   (via)      Cvia=0.3p          * Package via
*Xlp1    jp2   jp3   (strilpline6_fr4)  length=len1     * Line seg 1
 Xlp1    jpt2  jnt2  jp2   jn2   jpu2  jnu2 jpt3  jnt3  jp3  jn3  jpu3  jnu3  (stripline6_fr4)  length=len1          * Line seg 1
 Xvp2    jp3   jp4   (via)      Cvia=0.7p          * Daughter card via
 Xvp3    jp5   jp6   (mvia)                          * Backplane via
 Xvp5    jp9   jp10  (via)      Cvia=0.7p          * Daughter card via
 Xlp3    jpt10 jnt10 jp10  jn10  jpu10 jnu10 jpt11 jnt11 jp11 jn11 jpu11 jnu11  (stripline6_fr4) length=len3 * Line seg 3
 Xvp7    jp11  jp12  (via)      Cvia=0.3p          * Package via
 Xpp2    jrp   jp12  (bondwire_pkg)                          * Recvr package model

* Compliment Interconnect *
```

```
 Xpn1    npad  jn1   (bondwire_pkg)                          * Driver package model
 Xvn1    jn1   jn2   (via)      Cvia=0.3p          * Package via
 Xvn2    jn3   jn4   (via)      Cvia=0.7p          * Daughter card via
 Xvn3    jn5   jn6   (mvia)                                  * Backplane via
 Xvn5    jn9   jn10  (via)      Cvia=0.7p          * Daughter card via
 Xvn7    jn11  jn12  (via)      Cvia=0.3p          * Package via
 Xpn2    jrn  jn12   (bondwire_pkg)                          * Recvr package model

* Orthogonal Connector Modelling *
 Xk1  jp4 jp5 jn4 jn5 0 0 0 0 (xcede_2diffpair_conn)
 Xk2  jp6 jp9 jn6 jn9 0 0 0 0 (xcede_2diffpair_conn)
**********************************************************************
*                                                                    *
*                          Crosstalk Variation                       *
*                                                                    *
**********************************************************************
.PARAM bps1 = 8.74g
* PRBS7 Signal Source *
 Xs1  inpt1 innt1  (bitpattern) dc0=0 dc1=1 baud='1/bps1' latency=0 tr=trise
 Xs2  inpt2 innt2  (bitpattern) dc0=0 dc1=1 baud='1/bps1' latency=0 tr=trise
* Behavioral Current Mode Driver with Pre-emphasis - REPLACE AS REQ'D       *
*    1 pre-cursore & 2 psot-cursor taps                                     *
 Xf1  inpt1 int1   (RCF) TDFLT='0.25*trise'
 Xf2  inpt2 int2   (RCF) TDFLT='0.25*trise'
 Xd1  int1  ppadt1 npadt1  (tx_4tap_diff) ppo=vd bps=bps1 a0=pre1 a2=post1 a3=post2 a4=post3 a5=post4 a6=post5
 Xd2  int2  ppadt2 npadt2  (tx_4tap_diff) ppo=vd bps=bps1 a0=pre1 a2=post1 a3=post2 a4=post3 a5=post4 a6=post5
* True Interconnect *
 Xppt1    ppadt1  jpt1   (bondwire_pkg)                        * Driver package model
 Xvpt1    jpt1   jpt2   (via)    Cvia=0.3p          * Package via
 Xvpt2    jpt3   jpt4   (via)    Cvia=0.7p          * Daughter card via
 Xvpt3    jpt5   jpt6   (mvia)                                * Backplane via
 Xvpt5    jpt9   jpt10  (via)    Cvia=0.7p          * Daughter card via
 Xvpt7    jpt11  jpt12  (via)    Cvia=0.3p          * Package via
 Xppt2    jrtp   jpt12  (bondwire_pkg)                        * Recvr package model

* Compliment Interconnect *

 Xpnt1    npadt1  jnt1   (bondwire_pkg)                       * Driver package model
 Xvnt1    jnt1   jnt2   (via)    Cvia=0.3p          * Package via
 Xvnt2    jnt3   jnt4   (via)    Cvia=0.7p          * Daughter card via
 Xvnt3    jnt5   jnt6   (mvia)                                * Backplane via
 Xvnt5    jnt9   jnt10  (via)    Cvia=0.7p          * Daughter card via
 Xvnt7    jnt11  jnt12  (via)    Cvia=0.3p          * Package via
 Xpnt2    jrtn  jnt12   (bondwire_pkg)                        * Recvr package model

* True Interconnect *
 Xppu1    ppadt2  jpu1   (bondwire_pkg)                              * Driver package model
 Xvpu1    jpu1   jpu2   (via) Cvia=0.3p          * Package via
 Xvpu2    jpu3   jpu4   (via) Cvia=0.7p          * Daughter card via
 Xvpu3    jpu5   jpu6   (mvia)                                       * Backplane via
 Xvpu5    jpu9   jpu10  (via)Cvia=0.7p           * Daughter card via
 Xvpu7    jpu11  jpu12  (via)          Cvia=0.3p          * Package via
 Xppu2    jrup   jpu12  (bondwire_pkg)                        * Recvr package model

* Compliment Interconnect *
 Xpnu1    npadt2  jnu1   (bondwire_pkg)                              * Driver package model
 Xvnu1    jnu1   jnu2   (via) Cvia=0.3p          * Package via
 Xvnu2    jnu3   jnu4   (via) Cvia=0.7p          * Daughter card via
 Xvnu3    jnu5   jnu6   (mvia)                                       * Backplane via
 Xvnu5    jnu9   jnu10  (via)Cvia=0.7p           * Daughter card via
 Xvnu7    jnu11  jnu12  (via)          Cvia=0.3p          * Package via
 Xpnu2    jrun  jnu12   (bondwire_pkg)                        * Recvr package model

* Orthogonal Connector Modelling *
 Xkt1  jpt4 jpt5 jnt4 jnt5 jpu4 jpu5 jnu4 jnu5 (xcede_2diffpair_conn)
 Xkt2  jpt6 jpt9 jnt6 jnt9 jpu6 jpu9 jnu6 jnu9 (xcede_2diffpair_conn)
**********************************************************************
*                                                                    *
*         Behavioral Receiver - REPLACE WITH YOUR RECEIVER           *
*                                                                    *
**********************************************************************

 Rrp1  jrp 0  rterm     *Lecture 11: Slide 6 On die termination is better*
```

```
Rrn1  jrn 0  rterm

* Differential Receive Voltage *
* DFE tap *
Xfeed  jrp  jrn  rx_diff  0  (tx_4tap_feed_diff)  ppo=vd  bps=bps  a0=pre1feed  a2=post1feed  a3=post2feed  a4=post3feed  a5=post4feed
a6=post5feed
* Eye Diagram Horizontal Source *
 Veye1 eye 0 PWL (0,0 '1./bps',1 R TD='edui/bps')
 Reye  eye 0 1G
**************************************************************************
*                                                                        *
*                    Sub-Circuit Definitions                             *
*                                                                        *
**************************************************************************
* Motherboard Via Sub-circuit *
*     zvia    = via impedance, ohms
*     len1via = active via length, inches
*     len2via = via stub length, inches
*     prop    = propagation time, seconds/inch
*
 .SUBCKT (mvia) in out  zvia=42 len1via=0.09 len2via=0.03 prop=180p
   T1  in  0 out 0  Z0=zvia TD='len1via*prop'
   T2  out 0 2   0  Z0=zvia TD='len2via*prop'
 .ENDS (mvia)

* Daughter Card Via Sub-circuit -- typical values for 0.093" thick PCBs *
 .SUBCKT (via) in out  Rvia=1m Lvia = 0.5n Cvia = 0.7p
    X1  in 1    (section_t) R_sec='0.5*Rvia' L_sec='0.5*Lvia' C_sec='0.5*Cvia'
    X2  1  out  (section_t) R_sec='0.5*Rvia' L_sec='0.5*Lvia' C_sec='0.5*Cvia'
 .ENDS (via)

* Generic "T" Section *
 .SUBCKT (section_t) in out  R_sec=1m L_sec=5n C_sec=2p
    Rs1  in 1    '0.5*R_sec'
    Ls1  1  2    '0.5*L_sec'
    Cs1  2  0     C_sec
    Ls2  2  3    '0.5*L_sec'
    Rs2  3  out  '0.5*R_sec'
 .ENDS (section_t)
********
* Package parameters for Bond Wire Package *
.SUBCKT (bondwire_pkg) in out C_pkg_esd=1p L_pkg_bump=3n Trace_len_pkg=1cm L_pkg_ball=0.5n
X1 in 1 (section_t) R_sec=0 L_sec=L_pkg_bump C_sec=C_pkg_esd
T1 1 0 2 0 Z0=50 TD=60p
X3 2 out (section_t) R_sec=0 L_sec=L_pkg_ball C_sec=0
.ENDS (bondwire_pkg)
********
*    Simple connector model - We have employed Orthogonal connector model *
********
**************************************************************************
*                                                                        *
*          Included Files - MODIFY TO POINT TO YOUR LOCATIONS            *
*                                                                        *
**************************************************************************
 .INCLUDE './stripline6_fr4.inc'
 .INCLUDE './prbs7.inc'
 .INCLUDE './tx_4tap_diff.inc'
 .INCLUDE './tx_4tap_feed_diff.inc'
 .INCLUDE './filter.inc'
 .INCLUDE 'connectors/Xcede/Spice_Models/xcede_2diffpair_conn.inc'
**************************************************************************
*                                                                        *
*                          Simulation Controls and Alters                *
*                                                                        *
**************************************************************************
 .OPTIONS post
 .TRAN 5p simtime SWEEP DATA=pdata
 .DATA  pdata
      len1 len2 len3 pre1 post1 post2 post3 post4 post5 edui pre1feed post1feed post2feed post3feed post4feed post5feed

 *    0.393  0.0  5.51 -0.10 -0.10 -0.10 0.00 0.00 0.00 0.06 0.02 0.00 0.00 0.00 0.00 0.00
  *   2.36  0.0  9.84 -0.10 -0.10 -0.10 0.00 0.00 0.00 0.06 0.02 0.00 0.00 0.00 0.00 0.00
        2.36  0.0  9.84  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 .ENDDATA pdata
```

```
.END
```

tx_4tap_diff.inc

```
*************************************************************************
*                                                                       *
*                   Behavioral Four Tap Equalizer                       *
*                                                                       *
*************************************************************************
*
* *** RESTIRCTIONS: ***
* 1. This differential driver must be loaded with 100 ohms to give correct levels
* 2. The input swing must be zero to 1 volt
*
.SUBCKT (tx_4tap_diff) in outp outn  ppo=700m bps=840p a0=0 a2=0 a3=0 a4=0 a5=0 a6=0
  R1  in 1  1G                          * Terminate "open" input
  V1  1  0  DC 0.5
  Ed1 2  0  DELAY (in,1) TD='1/bps'             * One bit delay
  Ed2 3  0  DELAY (in,1) TD='2/bps'             * Two bit delay
  Ed3 4  0  DELAY (in,1) TD='3/bps'             * Three bit delay
    Ed4 5  0  DELAY (in,1) TD='4/bps'                 * Four bit delay
    Ed5 6  0  DELAY (in,1) TD='5/bps'                 * Five bit delay
    Ed6 7  0  DELAY (in,1) TD='6/bps'                 * Six bit delay
  R2  2  0  1G                          * Delay termination
  R3  3  0  1G                          * Delay termination
  R4  4  0  1G                          * Delay termination
    R7  5  0  1G                        * Delay termination
    R8  6  0  1G                        * Delay termination
    R9  7  0  1G                        * Delay termination
  *
  G0  outp outn  (in,1) '0.5*ppo*a0/14'           * Pre-cursor source
  G1  outp outn  (2,0)  '0.5*ppo*(1-(a0+a2+a3+a4+a5+a6))/14'  * Cursor source
  G2  outp outn  (3,0)  '0.5*ppo*a2/14'           * 1st post-cursor source
  G3  outp outn  (4,0)  '0.5*ppo*a3/14'           * 2nd post-cursor source
    G4  outp outn  (5,0)  '0.5*ppo*a4/14'         * 3rd post-cursor source
    G5  outp outn  (6,0)  '0.5*ppo*a5/14'         * 4th post-cursor source
    G6  outp outn  (7,0)  '0.5*ppo*a5/14'         * 4th post-cursor source
  R5  outp 0    45                      * Driver output termination
  R6  outn 0    45                      * Driver output termination
.ENDS (tx_4tap_diff)
```

tx_4tap_diff_feed.inc

```
*************************************************************************
*                                                                       *
*                   Behavioral Four Tap Equalizer                       *
*                                                                       *
*************************************************************************
.SUBCKT (tx_4tap_feed_diff) in1 in2 outp outn  ppo=700m bps=840p a0=0 a2=0 a3=0 a4=0 a5=0 a6=0

  .PARAM cur = 10e-3

  *************************** Differential Receive VCCS ****************************
  Gx  outp outn  (in2,in1)  0.01
  Rx  outp outn  100                    * Driver output termination
  ***************************************************************************************

  ********************************** Comparator ***********************************
  Ecomp  outcomp 0  PWL(1)  outp  outn  -1f,-1v  1f,1v
  ***************************************************************************************

  R1  outcomp 0  1G                     * Terminate "open" input
  Ed1 2  0  DELAY (outcomp,0) TD='1/bps'            * One bit delay
  Ed2 3  0  DELAY (outcomp,0) TD='2/bps'            * Two bit delay
  Ed3 4  0  DELAY (outcomp,0) TD='3/bps'            * Three bit delay
  Ed4 5  0  DELAY (outcomp,0) TD='4/bps'            * Four bit delay
  Ed5 6  0  DELAY (outcomp,0) TD='5/bps'            * Five bit delay
  Ed6 7  0  DELAY (outcomp,0) TD='6/bps'            * Six bit delay
  R2  2  0  1G                          * Delay termination
  R3  3  0  1G                          * Delay termination
  R4  4  0  1G                          * Delay termination
  R7  5  0  1G                          * Delay termination
  R8  6  0  1G                          * Delay termination
  R9  7  0  1G                          * Delay termination
  G1  outp outn  (2,0)  'a0*cur'    * Cursor source
  G2  outp outn  (3,0)  'a2*cur'        * 1st post-cursor source
```

```
   G3  outp outn  (4,0)  'a3*cur'          * 2nd post-cursor source
   G4  outp outn  (5,0)  'a4*cur'          * 3rd post-cursor source
   G5  outp outn  (6,0)  'a5*cur'          * 4th post-cursor source
   G6  outp outn  (7,0)  'a6*cur'          * 5th post-cursor source
.ENDS (tx_4tap_feed_diff)
```

stripline6_fr4.inc

```
.SUBCKT (stripline6_fr4) inp1 inn1 in inn inp2 inn2 outp1 outn1 out outn outp2 outn2 length=1 *inch
  W1 inp1 inn1 in inn inp2 inn2  0 outp1 outn1 out outn outp2 outn2 0 RLGCMODEL=stripline6_fr4 N=6 l='length*0.0254'
.ENDS (stripline6_fr4)
```

stripline_calc.sp

```
*Example: 6 mil 50 ohm stripline trace in FR4; Half ounce copper
V1  s    0    PULSE 0v 2v 5n 0.5n 0.5n 50n
R0  s    in1  50
R1  out1 0    50
**************************************************************************
*                                                                        *
* The W Element is a frequency sensitive transmission line model. See    *
* the online spice manual for more details. On UNIX machines, it can     *
* be found at: /usr/class/ee/hspice/Z-2007.03-SP1/hspice/docs/                *
*                                                                        *
**************************************************************************
*  This use of the W, with the key word FSmodel, invokes the field solver *
W1 inp1 inn1 inp2 inn2 inp3 inn3 0 outp1 outn1 outp2 outn2 outp3 outn3 0 FSmodel=stripline6_fr4 N=6 l=0.5 *meters
*                                                                        *
**************************************************************************
*  The following lines specify the materials and geometries for the      *
*  field solver to analyze.                                                    *
*                                                                        *
* Specify FR4 dielectric material:                                            *
.MATERIAL diel_1 DIELECTRIC ER=4.0, LOSSTANGENT=3e-2
* Specify copper conductor material:                                         *
.MATERIAL copper METAL CONDUCTIVITY=57.6meg
* Specify rectangular shape                                                  *
.SHAPE rect RECTANGLE WIDTH=0.1524mm HEIGHT=0.01524mm
* Specify layer stack-up -- PEC means reference planes                       *
.LAYERSTACK stack_1, LAYER=(PEC,0.01524mm), LAYER=(diel_1,0.3708mm),
+ LAYER=(PEC,0.01524mm)
* Compute skin resistance & dielectric loss                                  *
.FSOPTIONS opt1 PRINTDATA=yes ACCURACY=high COMPUTEGD=yes COMPUTERS=yes
* Build model 'stripline6_fr4' & write to file 'stripline6_fr4.rlgc'         *
.MODEL stripline6_fr4 W MODELTYPE=FieldSolver, LAYERSTACK=stack_1,
+ FSOPTIONS=opt1, RLGCFILE=stripline6_fr4.rlgc
+ CONDUCTOR=(SHAPE=rect,ORIGIN=(0,0.1778mm),  MATERIAL=copper)
+ CONDUCTOR=(SHAPE=rect,ORIGIN=(0.6096mm, 0.1778mm), MATERIAL=copper)
+ CONDUCTOR=(SHAPE=rect,ORIGIN=(1.8288mm,0.1778mm),  MATERIAL=copper)
+ CONDUCTOR=(SHAPE=rect,ORIGIN=(2.4384mm, 0.1778mm), MATERIAL=copper)
+ CONDUCTOR=(SHAPE=rect,ORIGIN=(3.6576mm,0.1778mm),  MATERIAL=copper)
+ CONDUCTOR=(SHAPE=rect,ORIGIN=(4.2672mm, 0.1778mm), MATERIAL=copper)
*                                                                        *
**************************************************************************
* After the RLGC file is computed, subsequent uses of the model should *
* use the following W element form to avoid re-invoking the field        *
* solver (the .INCLUDE line is needed to include the RLGC model file): *
*                                                                        *
* .include './stripline6_fr4.rlgc'
* W1 in1 0 out1 0 RLGCMODEL=stripline6_fr4 N=1 l=0.5 *meters
**************************************************************************
.OPTION probe post
.TRAN 0.5n 100n
.PROBE V(in1), V(out1)
.END
```

#####################FFE Equalization Parameters MATLAB Script ######################

```
function [ coeffs target z y ] = emphasis ( trace, bit_rate, sample_rate, offset_in, delay, swing, offset_target )
% Emphasis optimizer. Requires HspiceTools and CVX.
% Frank Austin Nothaft - fnothaft@stanford.edu - 3/1/2011
% trace -> trace file for analysis
% bit_rate -> bitrate that system is running at
% sample_rate -> rate that system is sampled at
% offset_in -> DC offset of trace in
% delay -> time offset of pulse arrival
```

```
% swing -> desired swing of equalized signal
% offset_target -> targeted DC offset of equalized signal

% load hspice trace
x = loadsig (trace);
y = evalsig (x, 'rx_diff');

% find samples in a bitcell
samples_per_bitcell = sample_rate / bit_rate;

% build matrix for optimization
pre_3 = [y' (offset_in * (ones (1, 6 * samples_per_bitcell)))]';
pre_2 = [(offset_in * ones (1, samples_per_bitcell)) y' (offset_in * ones (1, 5 * samples_per_bitcell))]';
pre_1 = [(offset_in * ones (1, 2 * samples_per_bitcell)) y' (offset_in * ones (1, 4 * samples_per_bitcell))]';
cursor = [(offset_in * ones (1, 3 * samples_per_bitcell)) y' (offset_in * ones (1, 3 * samples_per_bitcell))]';
post_1 = [(offset_in * ones (1, 4 * samples_per_bitcell)) y' (offset_in * ones (1, 2 * samples_per_bitcell))]';
post_2 = [(offset_in * ones (1, 5 * samples_per_bitcell)) y' (offset_in * ones (1, samples_per_bitcell))]';
post_3 = [(offset_in * ones (1, 6 * samples_per_bitcell)) y']';

z = [pre_3 pre_2 pre_1 cursor post_1 post_2 post_3];
o = ones (1, 7);

% build target signal
target = (offset_target * ones (1, length (pre_3)) + [(zeros (1, delay)) (swing * ones (1, samples_per_bitcell)) (zeros (1, (length (pre_3) -
delay - samples_per_bitcell)))])';

% solve convex problem
cvx_begin
  variable coeffs(7);
  minimize ( norm (z * coeffs - target) );
  subject to
    coeffs >= 0;
    o * coeffs == 1;
cvx_end

t = 1:length (y);
figure;
hold on;
plot (z);
plot (target)
plot (z * coeffs);
end
```