Equalizing Filter Design for High-Speed Off-Chip Buses

by

Jihong Ren

B.Sc., Huazhong University of Science and Technology, 1995
M.Sc., Huazhong University of Science and Technology, 1998
M.Sc., The University of British Columbia, 2000
M.Sc., The University of British Columbia, 2002

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

The Faculty of Graduate Studies

(Computer Science)

The University Of British Columbia

December 2005

Abstract

On-chip speeds and integration densities have grown exponentially over the past several decades creating a corresponding demand for high-bandwidth, chip-to-chip communication. Compared with integrated circuit technology, the technologies for chip-packaging, printed circuit boards, and connectors improve at a much slower rate. This results in a big and growing gap between the I/O bandwidth needed and the I/O bandwidth available. Off-chip bandwidth has become a bottleneck in developing high-speed systems.

At high data rates, high-frequency losses, reflections and crosstalk severely degrade signal integrity and limit the performance of off-chip links. To combat these issues, designers increasingly rely on on-chip signal processing methods. This thesis explores the effectiveness of equalizing filters for high-bandwidth, point-to-point, off-chip buses. In this work, we combine modelling, optimization and prototyping to demonstrate that linear programming provides practical, effective and flexible basis for designing equalization filters that greatly increase the bandwidth of high-speed buses on printed circuit boards. We first show that the common eye-mask measure of signal integrity is a worst-case performance measure that corresponds to the l_∞ metric. We show how eye masks can be parameterized to provide a flexible framework for specifying signal integrity trade-offs. We use these parameterized masks to formulate the l_∞ optimal equalization filter synthesis problem, and show that it can be extended to the unified optimization of pre-equalization, near-end crosstalk cancellation and decision-feedback equalization filters. Our methods work with detailed, realistic channel models and allow the designer to specify practical constraints such as the maximum filter output and bounds on filter coefficients.

Our approach formulates equalization filter synthesis as a linear programming problem. While this makes our approach very flexible, the linear programs that we create can be quite large. To make our methods practical, we implemented a novel linear system solver for use in Mehrotra's interior point linear programming algorithm. Our solver exploits the specific sparsity properties of our optimization problems. We analyze the time and memory requirements of this new implementation as well as its numerical stability.

We evaluated the effectiveness of our algorithms by synthesizing filters for various high-speed signalling scenarios. We used simulation to validate these filters and demonstrate their advantages over existing synthesis methods. To address the practical issues that arise in real systems, we implemented a proof-of-concept test bed that enables preliminary evaluations of our filters with physical, printed circuit board buses. Through this combination of modelling, algorithm design, simulation, and prototyping, we have demonstrate the advantages and practicality of our approach to equalizing filter design.

Contents

List of Tables

List of Figures

Acknowledgements

Five years ago, as a student with no background in Computer Science, I stepped into Dr. Mark Greenstreet's office, hoping for a chance to start my graduate study in this prestigious department. It was my lucky day. Although the interview was almost a complete disaster, he saw my potential and had faith in me. Since then, his spirit has given me inspiration in academia and life in general. I am deeply grateful for the countless hours of lectures, discussions and talk preparations, and for the conversations that encouraged me to continue when I was worn down. Without his extensive support, patience and endless encouragement, my thesis would not have been possible.

I would also like to thank Dr. Anne Condon for her continuous support and inspiration over the years. Anne and Mark showed me that the best way to inspire and teach a student is to set a good example. I would also like to thank Dr. Chen Greif for his very helpful insights on numerical optimization, Dr. Roberto Rosales for his expert assistance in the test lab and his patience when I worked on my test bed, and Holly Kwan for her kindness and assistance throughout the years. Many other members of the CS department also devoted their time whenever I asked for their help. I really enjoyed my five-year graduate study in this department.

I would like to thank Dr. John Poulton for many valuable comments on my thesis proposal and for his suggestion of using the RAMDACs on graphics cards to provide communication channels, which later became the basis of my low-speed test bed. Moreover, I would like to thank the members of my supervisory committee, Dr. Mark Greenstreet, Dr. Anne Condon, Dr. Chen Greif, Dr. Will Evans, and Dr. Res Saleh, for their suggestions and careful proof-reading of my thesis.

Special thanks to Rui Li who encouraged me to apply for graduate school in Computer Science and has supported me since then. Finally, to all of my friends, thank you for such a enjoyable ride!

Chapter 1

Introduction

Advances in digital integrated circuit (IC) fabrication technology have resulted in an exponential growth for the speed and integration levels of ICs with a corresponding demand for high-bandwidth for off-chip buses.[1] Rent's rule [40] predicts that off-chip bandwidth requirements grow as

$$B \propto n^p f \tag{1.1}$$

where B is the bandwidth required by a component, n is the number of devices on the component, f is the operating frequency, and p is a "constant" depending on the type of component, typically in the range of 0.4 to 0.8. Although the number of I/Os has increased from 16 $\sim$ 24 pins in the 1970s, to several hundred pins per IC now [64], this growth is being rapidly out-paced by the bandwidth demands. To continue to improve overall system performance, the per-pin interconnection bandwidth must scale with the speed and integration level of ICs. The ITRS roadmap projects a need for per-pin I/O bit rates that track clock frequencies, reaching 9.6 Gbits/sec by year 2009 and nearly 25 Gbits/sec by 2018 [3, p. 23-25]. However, compared with on-chip technology, the technologies for chip-packaging, printed circuit boards (PCB), and connectors improve at a much slower rate. This results in a large and growing gap between the I/O bandwidth needed and the I/O bandwidth available. Off-chip bandwidth has become a bottleneck in developing high-speed systems. Such bandwidth is especially critical for high-performance memory systems [16], inter-processor communication in shared memory multiprocessors (SMP) [7, 12, 24] and connections between line cards on backplanes for high-speed network routers [43, 69]. To meet this demand, designers are relying increasingly on equalizing filters and other on-chip signal processing techniques to maximize the utilization of off-chip interconnect.

Off-chip interconnects are nowhere near ideal links and present great challenges to achieving multi-Gb/s signalling rates. For example, the inter-board channel shown in Figure 1.1 includes on-chip parasitics, package parasitics, backplane connectors, PCB traces and vias. At high data-rates, each of these contributes to

[1] Appendix A describes the terminology that is used in this thesis to describe high-speed buses. Acronyms are defined with their first use, and a glossary of acronmyms is provided in Appendix B.

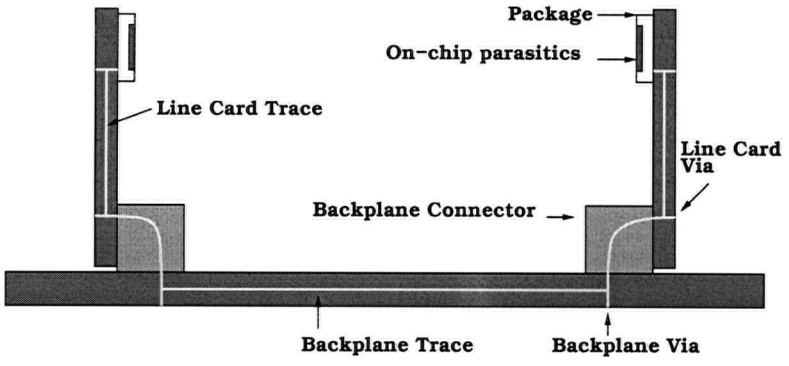


Figure 1.1: A high-speed system backplane channel [39].

severe signal integrity degradation such as dispersive and dielectric losses, reflections and crosstalk. In Section 2.2, we will discuss these channel impairments in greater detail.

PCB traces, connectors and packages have dispersive losses due to their resistances which become more severe at high frequency due to the skin effect. At frequencies greater than 1GHz, dielectric losses dominate. Although low-loss dielectric materials exist, FR-4 (epoxy-fiberglass) is still commonly used due to its low cost and the maturity of the associated manufacturing processes. Furthermore, reflections occur due to impedance discontinuities along the link, i.e. at the interfaces between the connector and the PCB, between the PCB and the package, etc. Both dispersive losses and reflections result in spreading the pulse's energy into the time slots for adjacent bits; a phenomenon called intersymbol interference (ISI) which severely limits the channel bandwidth. Moreover, high bit rates exacerbate the problems of electro-magnetic coupling (crosstalk) between channels in high-speed buses. Crosstalk occurs in connectors, vias, packages and PCB traces. Short signal rise and fall times exacerbate coupling effects, making crosstalk a primary concern for present and future high-speed, high-density interconnect design.

As integration density and data rates increase, dispersive losses, reflections and crosstalk can severely degrade signal integrity. Fortunately, these effects are *linear processes*. Accordingly, simple, on-chip signal processing techniques can compensate for them. This is the basic idea behind equalization as shown in Figure 1.2. An ideal transmission channel would in all cases deliver a delayed version of the input signal $v_{in}(t)$ from the driver without distortion to the receiver, i.e. $v_{out}(t) = v_{in}(t - t_d)$, where t_d is the propagation delay across the channel. Equivalently, an ideal channel would have a frequency response of $e^{-j\omega t_d}I$, where

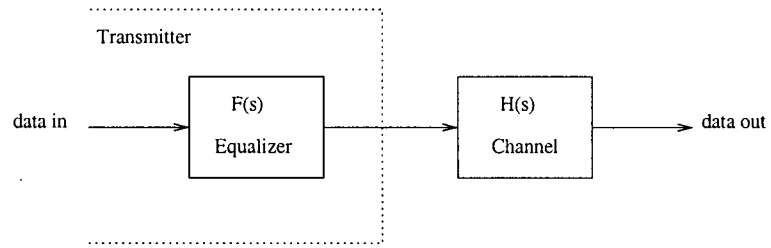


Figure 1.2: Block diagram of an equalized transmission channel (from [18, p. 364]).

$j = \sqrt{-1}$ and I is the identity matrix whose size is the number of input signals. If an equalizing filter has a transfer function that equals the inverse of the transfer function of the channel, then the concatenation of the equalizer and the channel will have a flat frequency and phase response. However, practical constraints such as power and limited switching time motivate studying optimal equalizing filter design under practical constraints.

This thesis explores the effectiveness of equalizing filters for high-bandwidth, point-to-point, off-chip buses. As shown in Figure 1.3, this research encompasses formulating optimal equalization filter synthesis as a linear programming problem, developing efficient numerical techniques for solving these linear programs, and implementing a proof-of-concept test bed to demonstrate the filters in a real, physical setting. We claim that

> **Thesis statement:** *Linear programming provides a practical, effective and flexible framework for designing equalization filters that greatly increase the bandwidth of high-speed, off-chip buses.*

The following are the major contributions supporting this thesis:

1. We present an algorithm for optimal equalization filter synthesis for buses with realistic models where each wire has its own model and boundary effects are modelled. Previously (in [54]) we demonstrated the advantages of multi-input, multi-output pre-equalization filters for reducing far-end crosstalk assuming a simplified bus model where all wires are identical (i.e. cylindrical), coupling depended only on wire separation and edge effects are ignored. This work extends this approach to more realistic models where each wire has its own model and boundary effects are modelled. Moreover, the

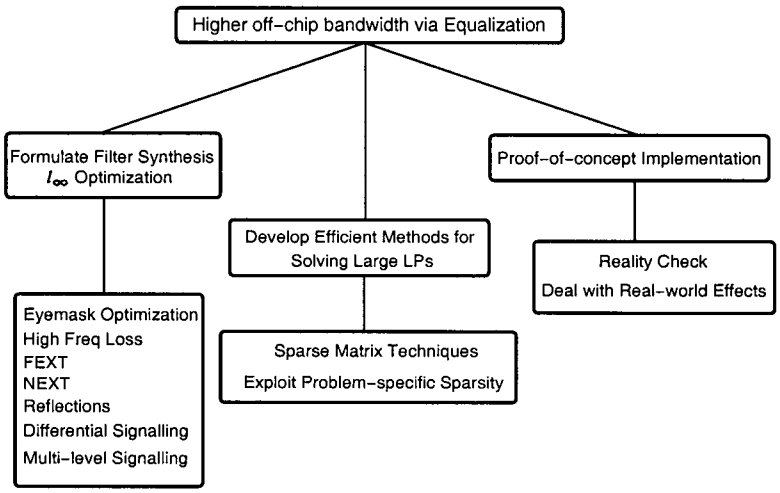


Figure 1.3: Research roadmap.

present approach addresses a comprehensive set of practical issues including the incorporation of differential and bidirectional channels, and modelling of realistic packaging and connector parasitics. Signal integrity issues such as dispersive and resistive losses, reflections, near-end crosstalk (NEXT) and far-end crosstalk (FEXT) are modelled.

2. We show that the common eye-mask measure of signal integrity is a worst-case performance measure that corresponds to the l_∞ metric. We introduce parameterized eye masks by specifying the vertices of the mask. Parameterization allows direct optimization of eye masks. The filter design problem for an optimal eye mask is an l_∞ optimization problem and can be formulated as a linear program.

3. We show that the joint optimization of pre-equalization (PE), near-end crosstalk cancellation (NE) and decision-feedback equalization (DFE) filters can be formulated as a linear programming problem.

4. The size of the resulting linear programs presents a great challenge to the practicality of our approach. To make the approach practical, we implemented a version of Mehrotra's interior point algorithm that exploits the problem-specific sparsity structure of our filter synthesis problems.

5. The unified optimization framework for equalization filter synthesis enables rapid evaluation of design trade-offs and early identification of bandwidth bottlenecks. We demonstrate this approach by evaluating different signalling methods such as bidirectional and unidirectional signalling, 2-level signalling and multi-level signalling for on-board links as well as backplane links connecting two daughter cards. Equalization filters designed using our approach significantly improve off-chip bandwidth in all cases.

6. We implemented a low-speed "scale-model" to provide a physical proof-of-concept demonstration of the novel filters designed using our approach. The low-speed test bed uses commodity graphics cards to provide high-speed RAMDACs and a computer interface for downloading filter coefficients and test sequences. It avoids massive design and expensive test equipment and enables quick turn-around time. Measurements from the test bed validate our synthesis procedures and show that we can get results that are comparable to those predicted by simulation in the presence of many real-world non-idealities such as timing jitter and power noise.

7. We employed an *in situ* approach to channel measurement. This provides a practical solution to the channel estimation problem and demonstrates that our results are quite robust even in the presence of channel estimation error.

The rest of this thesis is organized as follows. Chapter 2 provides an overview of the state-of-the-art in high-speed links, the equalization technique and its various applications. Chapter 3 introduces parameterized eye masks and extends the l_∞ filter design methodology developed previously to realistic link models. Chapter 4 describes a unified optimization framework for the three most common types of equalizing filters: pre-equalization, near-end crosstalk cancellation and decision-feedback equalization. The resulting linear programs are huge. Chapter 5 presents an efficient linear programming solver that exploits the sparsity and structure of the filter synthesis problems. Chapter 6 describes a low-speed test bed which provides a physical proof-of-concept demonstration of the novel filters designed using our approach. Chapter 7 concludes the thesis and describes several possibilities for future work.

Chapter 2

Related Work

Given the critical need for off-chip bandwidth, high-speed links have been an area of intense research in both industry and academia. Serial links have achieved the highest per-pin bandwidths by using aggressive design techniques including equalization, embedded clocks, and multi-level signaling. While bandwidth can be increased by using multiple serial links in parallel, this approach is limited by the latency of serialization and deserialization, power requirements of such links, and the difficulty of implementing a chip with a large number of independent phase-locked loops (PLLs). Thus, buses are used to achieve even higher total bandwidths. Section 2.1 examines recent work in both serial links and buses.

At high bandwidths, signal integrity issues become critical. In Section 2.2 we summarize the dominant signal integrity issues focusing on crosstalk, resistive and dielectric losses, and reflections. These can be addressed by equalization, which we examine in Section 2.3. We note that equalization has been used for decades in telephone subscriber loop and other long-distance communication settings. The use of equalization in high-speed, digital systems is more recent, but is now becoming pervasive. While an ideal equalizer would allow unlimited bandwidth, real designs are constrained by practical constraints of power and sample rate. Thus, we need to design the best possible equalizer given a set of practical constraints. In Section 2.4 we describe the two leading approaches to equalizer design that are distinguished by their optimization criteria. Least-squares based methods provide l_2 optimality, while linear programming based techniques allow optimization with an l_∞ criterion. We examine the connections between these criteria and digital signal integrity in the same section.

2.1 High-Speed Links: State-of-the-Art

Interconnections are moving from traditional multi-drop buses (shared buses) to point-to-point links. Besides the fact that shared bus architectures suffer from contention over the shared bus, in the multi-drop bus environment, impedance mismatches exist at every drop line of the bus, which limits its practical switching frequencies. For example, the conventional peripheral component interface

(PCI) [49] uses reflected-wave switching. Due to impedance mismatches, the ideal 2nd-reflected wave switching is never achieved in practice and multiple round-trip times are required for the waves to settle. Other electrical issues include capacitive loading and stub effects. These limit the performance of the PCI bus when scaling to higher operating frequencies. The latest revision of PCI, PCI-X operates at 133MHz with more complicated designs and higher cost than its predecessors. Compared with multi-drop buses, point-to-point links provide tight control of electrical parameters of the bus and hence higher operating frequencies. For example, 3D graphics and animations demand high bandwidth between the graphics card and main memory. To relieve this bottleneck, the accelerated graphics port (AGP) [35] was developed and dedicated to moving large blocks of 3D texture data between the PC graphics controller and system memory. The most recent version, AGP8X, operates at 533MHz and offers bandwidths of up to 2.12GB/s. The newly emerged PCI-express technology which might gradually replace the PCI, PCI-X and AGP buses over the next decade, is based on point-to-point communication with a switch fabric [50]. Moreover, compared with multi-drop buses, point-to-point links can more easily employ techniques such as equalization to improve bandwidth further. This research focuses on point-to-point links. Other technologies, such as crosstalk transfer logic (XTL) interface [47], have been proposed to improve multi-drop bus performance by using directional couplers to avoid problems of multiple reflections caused by impedance mismatches. While we do not consider such links in this thesis, our framework extends readily to any link with a linear model. Thus, our method could be used to synthesize filters for channels with directional couplers such as XTL.

Point-to-point links can be based on either serial links or parallel buses. Representatives of recent parallel interconnects are parallel RapidIO interconnect [53] for both intra-board and backplane links and the Redwood parallel interface [66] from Rambus for intra-board links etc. The RapidIO parallel interconnect supports 8/16-bit LVDS interface and operates at up to 1GHz. The Rambus Redwood parallel bus interface offers data rates of 400MGb/s to 6.4Gb/s. The PCI-express protocol [50] is serial link based and provides up to 32 lanes of serial links. Each lane contains two differential pairs to provide simultaneous bidirectional communication. First-generation PCI-express operates at a signalling rate of 2.5Gb/s per lane. PCI-SIG projects future signalling rates of 5 and 10Gb/s [50].

A point-to-point parallel bus provides greater bandwidth by increasing either the operating clock frequency or the width of the bus. The need for more pins limits the width of the bus. Moreover, parallel buses usually operate source-synchronously. An external clock is sent along with the data signals. A small skew within the bus could result in wrong data on clock edges. Consequently, routing becomes very challenging since trace lengths have to precisely match the length of the clock

trace. This requires more PCB area and can require more PCB layers. Clock skew among different channels can be mitigated by equalization techniques. Eventually clock skew and channel jitter limit the aggregated bandwidth of a parallel bus.

Compared with point-to-point parallel buses, point-to-point serial links use fewer pins and avoid the trace-matching problem by embedding clock into data. This reduces component cost as well as board layout cost since fewer board layers can be used. However, transceivers for serial links are more complicated than those for parallel buses and use large amounts of die area and power. Moreover, compared with parallel buses which incur minimal latency, serial links have additional latency due to serialization/deserialization, encoding/decoding and clock recovery from the data stream. This makes serial links unsuitable for low-latency applications such as the interconnect between CPU and memory. Parallel interfaces, such as RapidIO and Rambus Redwood, are more suitable for such applications where both latency and bandwidth must be considered. In applications where a single serial link cannot provide sufficient bandwidth, multiple, parallel lanes of serial links are needed. For example, the PCI-express protocol allows up to 32 lanes of serial links. For these reasons, parallel links are of interest, and in this research, we study equalization techniques that improve bandwidth for such off-chip, parallel buses.

2.2 Transmission Channel Limitations

As data rates move into the Multi-Gb/s/channel range, signal integrity issues such as crosstalk, reflections, intersymbol interference (ISI), ground-bounce, timing jitter, and substrate noise become critical and limit the channel bandwidth. The first three of these, crosstalk, reflections, and ISI, are linear processes and can be mitigated by on-chip equalization filters [17, 21, 62, 76]. For the rest of this section, we discuss the channel impairments, in particular, ISI, reflections and crosstalk. In Section 2.3, we then describe various equalization techniques that are currently used to mitigate these effects.

- **Resistive loss:**
  Conductors have resistance, determined by the material (typically copper) and the cross-sectional area of the conductor. At high frequencies, currents flow primarily along the surface of the conductor, resulting in a frequency-dependent increase in the resistance [18, pp. 103-105]. This is called the *skin effect*. At high frequencies, the resistance increases with the square root of the frequency. At low frequencies, current flows nearly uniformly in the entire cross-sectional area of the conductor; hence the conductor has a constant resistance at low frequencies. The onset frequency of skin effect, f_s, is inversely proportional to the area of the cross-section and the conductivity of

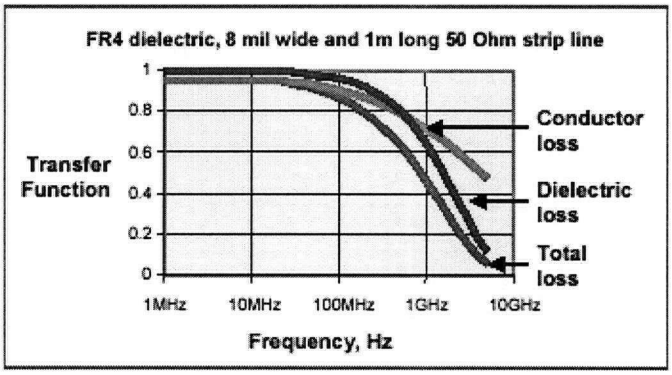


Figure 2.1: Loss as a function of frequency (from [48]).

the material. The frequency-dependent attenuation due to skin effect results in dispersion of travelling waves (see Figure 2.2C) which causes data bits to overlap with each other, a phenomenon called *intersymbol interference*.

- **Dielectric loss:**
  Dielectric materials absorb electro-magnetic energy and turn it into heat. The fraction of electro-magnetic power converted into heat by a dielectric material is called the *dielectric loss*. Similar to the skin effect, this results in dispersion of travelling waves and hence leads to intersymbol interference. The dielectric loss scales proportionally to frequency and dominates resistive loss at high frequencies. Figure 2.1 shows the total loss (resistive loss and dielectric loss) at different frequencies for 8 mil wide and 1m long 50 ohm stripline with FR-4 dielectric [48]. At frequencies greater than GHz, the dielectric loss becomes dominant. At 1GHz, these losses attenuate the signal to only 40% of its strength at low frequencies. Without a special signalling scheme or low-loss materials, channel bandwidth is limited by resistive and dielectric losses. Although better dielectric materials with lower dielectric loss are available, FR-4 is still generally used due to its low production costs and highly developed manufacturing infrastructure. Moreover, FR-4 is used in most legacy backplanes where losses are a major issue at high data rates.

- **Reflections:**
  Most off-chip links are electrically long at high data rates and must be modelled as transmission lines. For example, FR-4 has a dielectric constant of about 4.5 and propagation delay about 71 ps/cm. The electrical length,λ_e, of a 50 ps rising edge is 0.7 cm. As a rule of thumb, distributed models should

be used when the wire length is greater than or equal to $\lambda_e/6$. Thus the critical dimension separating lumped from distributed systems for PCBs is about 1mm which is very short compared with typical chip-to-chip links.

When a signal travelling down a transmission line encounters a boundary with an impedance discontinuity, part of the signal propagates through the boundary and part of it is reflected back. The amount of reflection is directly proportional to the severity of the impedance discontinuity:

$$k_r = \frac{Z_2 - Z_1}{Z_1 + Z_2} \tag{2.1}$$

where k_r is the reflection coefficient and Z_1 and Z_2 are the characteristic impedance of the right and left side of the boundary respectively. Hence, the greater the impedance mismatch, the bigger the reflection. In the extreme cases where the Z_2 is zero (infinite), all energy reflects back and k_r is -1 (+1). With impedance discontinuities along the link, it can take several round trips for the signal to settle and thus reflections can interfere with values sent later on. This results in severe intersymbol interference and also exacerbates crosstalk. Thus, for high-speed links, impedances are carefully controlled. However, for cost-effective designs, manufacturing tolerances limit the specification of the impedances of traces, connectors and package traces to roughly 10% of its nominal value. Moreover, via stubs of through vias for thick boards can cause large impedance mismatches and hence large reflections. The use of surface mount connectors, back-drilling, or blind vias can greatly reduce the via stub effect [39].

- **Crosstalk:**
  A signal on a link induces signals on nearby links due to electro-magnetic coupling between the links. Crosstalk between transmission lines is both inductive and capacitive. A signal propagating down a transmission line gets coupled to adjacent lines and results in waves propagating in both directions on adjacent lines. The forward-travelling wave induced by inductive coupling is negative while the forward-travelling wave induced by capacitive coupling is positive. In an ideal homogeneous environment, the coupling capacitance and inductance are duals and the forward-travelling waves induced cancel each other [18, pp. 272-278]. Moreover, with the high-frequency attenuation of the channel, far-end crosstalk is generally less troublesome than near-end crosstalk. Figure 2.2A shows a coupled microstrip transmission line. Figure 2.2B shows the configuration of one aggressor line and one victim line with proper terminations. Voltage waveforms at the near-end and far-end of both lines obtained by HSPICE are shown in Figure 2.2C. Since

the microstrip lines are inhomogeneous because of the interface between air and PCB dielectric, the forwarding capacitive and inductive coupling waves do not cancel each other. Thus, there is significant far-end crosstalk.

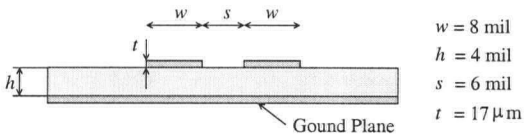
Crosstalk is also produced by connectors, vias and packages and is mostly inductive. The mutual inductance often approaches the level of self-inductance. High-speed packages and connectors control the level of crosstalk by providing more ground pins and carefully arranging the signal return path as close to the signal path as possible.

To maximize packing density, designers attempt to place signal lines as close to each other as possible. The small inter-line spacing and high data rates exacerbate the problem of crosstalk. In many situations, crosstalk can be the dominant concern for signal integrity. Traditionally, crosstalk is reduced by carefully controlling line geometry, shielding with grounded conductors, ensuring close signal return paths, arranging circuits to decrease the coupled line length, etc. Although these methods reduce crosstalk, they do not eliminate it. High performance PCB designs often require many revisions to produce a working design.

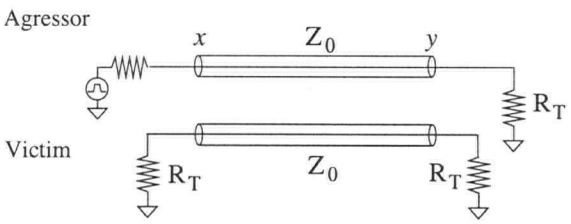
- **Noise:**
  Noise arises in high-speed buses from many sources. These include thermal noise, power supply noise, and crosstalk from signals outside the bus, and ground bounce. For practical designs, the energy per bit is much greater than the thermal noise floor. For example, a channel operating at 100 Gb/sec/wire (much faster than the current state-of-the-art) with a voltage swing of 0.1 volts (somewhat less than the current state-of-the-art) with 100Ω nominal bus impedance uses 10^{-15} Joules/bit. At a temperature of 300K (i.e. 27 Celsius), kT is $4.1 * 10^{-21}$ joules. Thus, the gap between the thermal noise floor and practical high-speed buses remains quite large. Power supply noise, crosstalk, and ground bounce are deterministic as much as they are side-effects of the operation of the system. However, they are very complicated to model, and designers often treat them as random noise source rather than attempting to model the system at a very fine level of detail.

  In this thesis, we will view channels as deterministic, and the optimal filter synthesis problem is to maximize the voltage margin at the receiver's input within this deterministic framework. In practice, this is a reasonable approach to providing robustness in the presence of the other sources of interference described above.

A



B



C

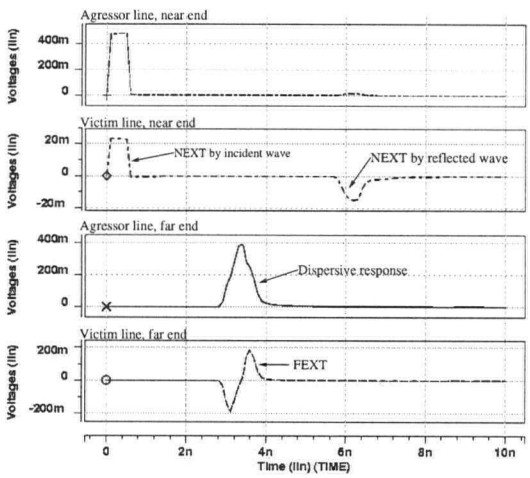


Figure 2.2: A. A pair of coupled microstrip transmission lines. The lines are drawn 25cm long, 8mil wide with 0.5Oz copper and separated by 6mil. The dielectric thickness of FR-4 is 4mil. Traces have characteristic impedance of approximately 50Ω. B. Test configuration of one aggressor and one victim with proper terminations. C. Voltage waveforms at near-end and far-end of both lines.

2.3 Equalization Techniques

As integration density and data rates increase, dispersive losses, crosstalk and reflections severely degrade signal integrity and limit off-chip bandwidth. However, these effects are linear processes and on-chip signal processing technology can compensate for them. In particular, equalization techniques have been widely used to actively compensate for imperfect channels for many applications. Theoretically, the maximum amount of information that can be transmitted over a channel is limited by channel's bandwidth as well as the signal to noise ratio [58]. The theoretical maximum information transfer rate of a channel is called the *Shannon Limit*. Recently, advanced coding techniques such as Turbo codes [10] come close to reaching the theoretical Shannon limit, but at a cost of high computational complexity for coding and decoding. These methods have been very successful in many applications such as wireless network interfaces.

As described in the previous section, high-speed buses operate under conditions where the energy per transmitted bit is far greater than the Shannon limit. Practical considerations of power and latency motivate designing simple filters that address the most significant signal integrity issues of ISI, crosstalk, and reflections. Thus, designers use FIR filters and thresholding receivers for their simplicity. Likewise, deterministic, linear channel models can be used to accurately account for ISI, crosstalk, and reflections. Filters designed with this approach can dramatically increase channel bandwidth and improve information rate with acceptable costs in terms of power, latency, and die area.

For example, equalization has been used effectively for crosstalk cancellation in acoustic applications such as telephone line subscriber systems [15, 30, 31]. In the past decade, designers started using equalization to compensate for the dispersive losses of high-speed off-chip serial links [17, 21, 62] as well as crosstalk cancellation for nearest neighbors for high-speed buses [75]. The rest of this section describes various equalization techniques such as pre-equalization, receive equalization, near-end crosstalk cancellation and decision feedback equalization.

- **Pre-equalization**
  Channel equalization can be performed by the transmitter, as shown in Figure 1.2, preceding the actual channel driver. Transmitters that utilize equalizing filters are called pre-distorting transmitters. Pre-distorting equalizers are commonly realized as finite impulse response (FIR) digital filters. While infinite impulse response (IIR) [36, p. 249] filters can be more flexible than FIR filters, they are generally not used for high data rate transmission because of the difficulty of calculating the IIR recurrence (i.e. feedback) at very high rates. The inputs to the equalizing FIR filters are the present and

past transmitted symbols. The output of the FIR filter is a weighted sum of these symbols. The length of the filter depends on the number of symbols that affect the response of the channel to the current symbol. The filter coefficients depend on the channel characteristics.

Pre-distorting transmitters were first used by Dally and Poulton [17] in a serial channel over copper wires at 4Gb/s to reduce intersymbol interference caused by frequency-dependent attenuation of the channel. Pre-equalizers boost the high-frequency components of the signal relative to the low-frequency components. It is often called pre-emphasis or de-emphasis. Since Dally and Poulton only considered the loss of the channel, to distinguish it from pre-equalizers that also deal with crosstalk cancellation, we call it single-line pre-emphasis (de-emphasis). Later, other groups [21, 62], e.g. Horowitz's group, used the same technique to design high-speed serial link transceivers. Rambus [75] reported using pre-equalizers for far-end crosstalk cancellation for nearest neighbors. In their design, each pin is responsible for cancelling the crosstalk it generates onto its neighbors by its transitions. However, they did not provide detailed information on the configurations of the filters and how they derived the filter coefficients.

FIR equalizing filters built into transmitters are easy to implement at very high speed because of the availability of transmitted symbols at the transmitter end. Furthermore, because each transmitted symbol is either 1 or 0, multiplication with the filter coefficients is easy. For example, in [17], a five-tap FIR filter is implemented with digital adders, and an interleaved digital-to-analog converter (DAC) is used to generate pre-distorted pulses. However, because transmitters generally do not have information about the received signals, FIR filter coefficients are obtained either by characterization of channel properties in advance [17, 21], or by adaptive implementation with feedback information provided by the receiver [27, 62].

- **Receive Equalization**
  An equalizing filter can also be incorporated into the receiver; this is called receive equalization. Receive equalization can be realized either with analog filters preceding the analog-to-digital converter (ADC) or with digital filters following the ADC. The latter one is the usual technique because digital filters are easy to implement and adapt. Compared with transmit equalizers, a receive equalizer can be adjusted using the channel information obtained at the receiver and does not require transferring information back to the transmitter end. However, receive equalization amplifies high frequency noise [32]. Furthermore, high-speed ADC technology typically

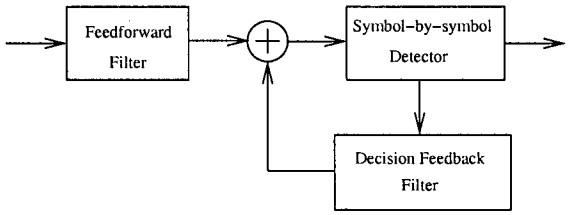


Figure 2.3: Decision feedback equalization.

lags behind high speed DAC technology in terms of maximum operating frequency. Therefore, pre-distorting transmitters are commonly used in high-speed transmission systems that run at multi-GHz speeds. Several years ago, Horowitz's group realized an 8-Gsamples/s ADC in 0.25 μm CMOS, which makes high-speed links with equalization at the receiver end possible with digital filters [71]. More recently, a high-speed I/O group at Intel reported an 8-Gb/s source-synchronous parallel bus with adaptive receive equalization. The filters are implemented as analog 4-tap discrete-time FIR filter and are 8-way interleaved to allow sufficient time for samples to be acquired [37].

- **Decision-Feedback Equalization**
  Compared with linear equalizers, decision-feedback equalizers (DFE) can correct for many of the same effects of crosstalk and intersymbol interference without incurring the linear equalizer's downsides of increased power consumption and noise amplification. DFE works especially well for channels with long impulse responses, i.e. a long channel with impedance discontinuities that result in multiple reflections. The basic idea of DFE is to utilize symbols determined from previous decisions to cancel intersymbol interference and also crosstalk if decisions on neighboring wires are also considered [5]. As shown in Figure 2.3, DFE is usually used in conjunction with a linear equalizer. The linear equalizer which can be a pre-equalizer or a receive equalizer, is called a feed-forward equalizer in this case because it removes intersymbol interference from 'future' symbols, i.e. symbols after the decision instant.

  The latency of the feedback loop limits the application of standard DFE at high data rates. This problem can be mitigated by unrolling the feedback loop and making two decisions at each cycle [60, 63]. Two slicers are used at the receiver, one assumes the previous bit is 1 and the other assumes it is 0. Once the decision for the previous bit is available, the correct answer is selected. This technique is also called partial response DFE. This approach can

be extended to accommodate even greater DFE latencies, but the exponential growth in the number of slicers limits this approach in practical designs.

DFE suffers from error propagation since it is based on previous decisions. An erroneous decision could result in burst of errors. The problem of error propagation can be kept in control by keeping the error probability low with other system choices such as pre-equalization and receive equalization, keeping the feedback filter short and constraining the magnitude of contributions from the feedback filter small.

- **Near-end Crosstalk Cancellation**
  In cases where transmitters and receivers are co-localized together or bidirectional links, near-end crosstalk noise is coupled to the attenuated signal received and can severely degrade signal integrity. Moreover, pre-equalizers exacerbate near-end crosstalk noise since they boost the high-frequency components of the signal relative to the low-frequency components in order to compensate for the high-frequency losses of the links. Active cancellation techniques have been used to alleviate near-end crosstalk noise for high-speed links [33]. Near-end equalizers are used effectively in local telephone subscriber loops. Local telephone subscriber loops use bundles of twisted copper wires and suffer severe crosstalk interference from neighboring channels due to the close physical proximity [30, 31]. The near-end equalizer approximates the near-end crosstalk noise by mimicking the near-end response. Then the approximated near-end crosstalk noise is subtracted from the corrupted received signal.

Equalization can also be used to mitigate the effects of transmit jitter [8]. In [8], Balamurugan and Shanbhag first showed that ISI in typical I/O channels tend to amplify high-frequency transmit clock jitter. Basically, if a transmitted edge is sent slightly later than its nominal time, the previously transmitted value will have had more time to change the bias closer to its asymptotic value. This increases the voltage swing required for the receiver to detect the current edge, amplifying the jitter. Likewise, if an edge is sent early, dispersion causes the transition to propagate faster, again amplifying the jitter. They then proposed a jitter equalizer attempting to change the jitter transfer function to a more benign response.

Practical designs for high-speed links often combine the equalization techniques described above to take advantage of the pros and cons of each. For example, pre-equalizers suffer from increased power consumption while receive equalizers amplify high frequency noise. DFEs, on the other hand, are limited by error-propagation and the latency of the feedback loop. Many practical serial link designs [63, 74] combine pre-equalizers and DFEs to combat ISI. Hur *et. al.* used a

combination of a receive equalizer and a NEXT noise canceller with a four-level pulse amplitude modulation signaling scheme to achieve 20-Gb/s transmission over backplane channels [33]. Moreover, these filters interact with each other. For example, pre-equalizers tend to boost high-frequency components of the signal relative to its low-frequency components to compensate for the high-frequency losses of the channel. This exacerbates near-end crosstalk noise. To achieve good performance at Multi-GHz speeds, it is crucial for the designers to jointly synthesize these equalizing filters.

Equalization techniques can also be classified in terms of methods for setting the equalization filter coefficients. Most recently, Zerbe et. al. from Rambus compared adaptive and non-adaptive equalization methods in high-performance backplanes with manufacturing variations and environmental variations such as temperature and humidity [73]. They compared three methods. The first method is "lookup table and forget" where channels are characterized in a lab and a best set of equalization filter coefficients are established, saved and used for all backplane links. The second method is "adapt once and forget". The backplane channels are characterized *in situ* on power-up and a set of equalization filter coefficients is derived for the corresponding backplane channel. This method takes manufacturing variations and component aging into account. The third method is "continuous adaptation" which also takes environmental variations into account. Their study shows that the latter two methods significantly outperform the first method at high data rates. The latter two are comparable with continuous adaptation performing slightly better. The equalization filter synthesis method proposed in this study characterizes the channel *in situ* and only optimizes the filter coefficients once. Fully adaptive schemes based on this approach should be possible but beyond the scope of the current research.

2.4 Equalizing Filter Design Methodologies

If an equalizing filter had a transfer function that was exactly the inverse of the transfer function of the channel, the concatenation of the equalizer and the channel would have a flat frequency and phase response and allow arbitrarily high bandwidth. However, practical limitations such as power and switching time motivate studying approximate filters. A straightforward approach is to first calculate the desired frequency response of the equalizing filter from the channel frequency response. Then, standard filter design procedures can be used to design equalizing filters that approximate the desired frequency response. For example, the following two steps can be used to obtain a more manageable impulse response function:

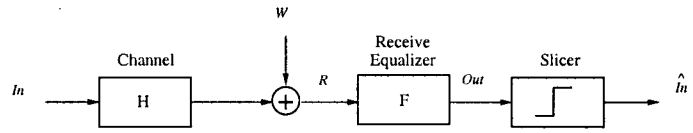


Figure 2.4: A linear equalizer receiver.

- Windowing: $h(n) = h_d(n)W(n)$ where $h_d(n)$ is the desired impulse response and $W(n)$ is the windowing function. This step is needed to obtain a filter with a finite number of taps.
- Delaying: $h(n)$ is shifted to the right until the samples are all indexed by non-negative integers to obtain a causal filter.

In practice, large windows must be used to obtain effective equalizing filters. Accordingly, many researchers have turned to using optimization methods to obtain good, approximate equalizing filters. Currently, there are two main classes of equalizing filter design methods for high speed links. One seeks to minimize the l_2 norm of the difference between the received signal and the transmitted signal. The other minimizes the l_∞ norm. For the rest of this section, we briefly describe these approaches.

- **l_2 and Minimum Mean Square Error**
  With an ideal transmission channel, the received signal is a delayed version of the transmitted signal. One can seek to minimize the l_2 norm of the difference between the received signal and a delayed version of the transmitted signal. This criterion is commonly used to design equalizers for telephone subscriber systems [15, 30, 31], wireless communication [9, Chapter 8] as well as optimal pre-emphasis equalizing filters for high-speed serial links [17, 63, 71]. The following briefly describes a few approaches based on l_2 optimality.

  Consider a single channel with receive equalization and additive white Gaussian noise as shown in Figure 2.4. As noted in Section 2.2, designers often assume a Gaussian noise model due to the impracticalities of accurately modelling the behavior of complex digital systems and the resulting power supply noise, ground bounce, crosstalk, etc. The received signal is given by:

$$\mathsf{Out} = f * h * \mathsf{In} + f * w \tag{2.2}$$

  where $*$ denotes convolution, h is the impulse response of the channel, f is the impulse response of the filter, In is the transmitted data, w is zero-mean

white Gaussian noise. Given the statistics of the input signal and the noise, the mean square error is defined as:

$$\text{MSE} = E_{\text{In},w}[(\text{Out} - \text{Out}_{ideal})^2] \tag{2.3}$$

where Out_{ideal} is the desired output, normally a delayed version of the input signal. Assuming that all of the random processes are wide-sense stationary with known statistics, the optimal minimum mean square error (MMSE) equalizer is given by:

$$f_{\text{MMSE}} = (E[rr^T])^{-1} E[\text{Out}_{ideal}\, r] \tag{2.4}$$

where $r = [R_L, \ldots, R_0, \ldots, R_{-L}]^T$, R is the input signal to the equalizer, $L = (N-1)/2$, and N is the number of equalizer taps [9, Chapter 9].

A one-shot solution to equation 2.4 is to invert the covariance matrix. Alternatively, an iterative procedure can be used by calculating the gradient of MSE and descending along the steepest descent direction. In practice, the covariance matrices are generally not available [9, Chapter 9]. A popular algorithm known as the *stochastic gradient* (SG) algorithm, sometimes also called the *least mean square* algorithm, provides an adaptive solution by substituting a time average for the expectation in the MMSE solution. This approach, although called LMS, only approximates the MMSE solution. This is the price paid for neither requiring the channel to be stationary nor knowing the channel's characteristics in advance. A simple, commonly used variation of the LMS algorithm is sign-sign LMS [63, 73]. The sign-sign LMS algorithm replaces the MSE calculation by using the sign of the input data and the measured error to update the filter coefficients.

If all symbols are equally likely and the additive noise is 0, then the MMSE equalizer above reduces to a filter that forces the ISI and crosstalk components of the channel's impulse response to be zero. This is called zero-forcing [9, Chapter 9]:

$$f_{ZF} = \arg\min_f \{\| f * h - e_\delta \|_2\} \tag{2.5}$$

where δ is the delay of the channel and e_δ denotes a column vector that is zero everywhere except for a one at δ.

- **l_∞ and Linear Programming**
  The MMSE and LMS methods described above minimize the *average* error of the channel. However, for digital systems, a more important objective is to minimize the *worst-case* error. In particular, the receiver in a digital link

includes some thresholding circuit. If the input is above the threshold for a logically high signal or below the threshold for a logically low signal in a large enough interval around the sampling time, then the signal will be received correctly. If the signal is between the two thresholds, an error may occur. Note that getting a signal that is already above the logically high threshold even closer to the high target value produces a correct result, but no more correct than for one that just barely exceeded the threshold. Likewise for low signals. Consider a channel where input patterns in_1 and in_2 both produce traces that satisfy the thresholds of the receiver. It may be possible to improve the l_2 metric by bringing the response to in_1 much closer to the target levels while pushing the response to in_2 slightly into the region between the thresholds. As this example shows, optimizing with respect to the l_2 metric is not guaranteed to maximize *digital* signal integrity. In [54], we presented a method for synthesizing optimal, pre-equalization filters for buses assuming a simplified bus model. We showed that the l_∞ metric corresponds to digital signal integrity and presented a filter synthesis method based on linear programming. We describe a more general formulation of this approach in chapters 3 and 4.

An advantage of the MMSE and LMS methods is that the effects of noise can be readily included in the model, although this does not necessarily translate into a precise analysis of the channel's bit-error rate (BER). The l_∞ methods consider worst-case behavior, and therefore do not handle noise and other statistical phenomena naturally. We revisit these issues in Chapter 7.

Linear programming has been used for FIR filter design for several decades [14, 34, 52, 57]. Given a desired frequency response, the design of a FIR filter can be viewed as a linear programming problem by forming a set of constraints for the actual frequency response of the FIR filter on a dense grid of frequencies. In addition to constraints on the frequency response, constraints on the time response such as maximum ripple in response to a step can also be imposed [52]. Samueli [57] used linear programming to design FIR filters that meet a spectral mask while satisfying requirements on ISI, e.g. the impulse response of the filters has uniformly spaced zero crossings. The work closest to ours is [13] in which linear programming techniques are explored for controlling ISI in a digital modem's signal path. They obtain the peak ISI by summing absolute values of system-pulse samples and explicitly limit or minimize peak ISI. The linear programs for these techniques are either solved with a general purpose LP solver such as Simplex [19, Chapter 5] or a special purpose LP solver that incorporates Fast Fourier Transformation (FFT) techniques to speed up the computation [41].

Previously (in [54]), we formulated pre-equalization filter synthesis for maximum eye height as a linear programming problem. We assumed a simplified bus model with all wires identical; in other words, we ignored edge effects by assuming cylindrical symmetry. Furthermore, we assumed that the the bus could be modelled as a collection of coupled transmission lines; in other words, we ignored discontinuities and parasitics due to packages, vias, connectors, etc. With this simple bus model, we demonstrated the advantages of l_∞ multi-input, multi-output, pre-equalization filters for reducing far-end crosstalk when compared with traditional l_2 filters. This work extends the l_∞ approach to more realistic bus models, addresses a comprehensive set of practical issues, provides a unified optimization framework for pre-equalization, near-end crosstalk cancellation and decision-feedback equalization filters, and validates the novel filters with a low-cost, low-speed test bed.

2.5 Summary

Off-chip links have entered the multi-Gb/s era. At such high data rates, channel impairments such as dispersive losses, crosstalk and reflections severely degrade signal integrity and limit off-chip bandwidth. Equalization techniques have been successfully used to compensate for high-frequency attenuation of off-chip channels. With equalization and carefully chosen signalling methods, multi-Gb/s serial links have been built. The single-line pre-emphasis technique has become part of recently introduced communication protocols such as PCI-express. Equalization is also commonly used in telephone subscriber systems to cancel near-end and far-end crosstalk [15, 30, 31].

With increasing integration density and data rates, ISI and crosstalk become even greater problems. In the near future, single-line pre-emphasis will be insufficient. Furthermore, it is crucial to combine different equalization techniques effectively when operating at multi-GHz operating frequencies. While there has been intensive research in high-speed links in the past eight years, there has been little CAD tool support. In this thesis, we address the synthesis of multi-input, multi-output equalization filters to deal with both ISI and crosstalk. Furthermore, we develop a unified optimization framework for jointly synthesizing equalization filters for high-speed off-chip buses. It enables fast evaluation of link performance and allows easy comparison of different link configurations and early identification of bandwidth bottlenecks.

In [54], we introduced l_∞ optimization for synthesizing pre-equalizers for high-speed digital buses. As described in Section 2.4, this approach was limited

by simple bus models, restricted objective functions, and only provided for synthesis of pre-equalizers. We now present new techniques that eliminate each of these limitations. First, we utilize eye masks, a commonly used measure of signal integrity that specifies bounds for the received signal throughout the time interval for transmitting a bit. We parameterize eye masks and then synthesize filters that directly optimize them. Second, we support arbitrary, linear bus models including impedance discontinuities and parasitics due to packages, vias, connectors, etc. Third, we show how to formulate the joint optimization of pre-equalizers, decision feedback equalizers, and near-end crosstalk cancellation filters as a unified linear programming problem.

Chapter 3

Optimal Pre-equalization Filter Synthesis

This chapter formulates the eye-mask optimization problem for single-ended, unidirectional channels with pre-equalization filters as a linear programming (LP) problem. To focus on the key ideas behind the optimization methods, we defer the more general, unified approach for synthesizing multiple filters until Chapter 4. Moreover, for simplification, the channel model used in this chapter ignores packaging parasitics, vias etc. which will also be addressed in Chapter 4. The key ideas presented in this chapter apply in the more general cases presented in later chapters.

In this chapter, we first describe eye masks, a common measure of signal integrity, and show how they can be parameterized to obtain objective functions for optimization. We show that eye masks, are an l_∞ (*worst-case*) measure: whether or not a link satisfies a given eye mask is determined by the worst-case outliers (highest and lowest levels) on the output channel over all possible input sequences. Accordingly, we first compute the worst-case responses of the channel. Because the filter and the bus are both linear, the output of the channel (filter and bus) is linear in the values of the inputs. Thus, the total distortion is the sum of the disturbances caused by each of the individual input bits on other wires and at other bit times. When all of the disturbances have the same sign, we obtain the maximum total disturbance and hence the worst-case response. This can be achieved by setting the sign of each input bit to produce positive disturbances at the sampling time. This leads to the l_∞ optimal filter synthesis approach presented in this thesis.

Section 3.2 states assumptions that simplify the presentation. To make the presentation more succinct and direct, we describe matrix formulations for linear convolutions with multi-input, multi-output channels in Section 3.3. We then show how to synthesize optimal pre-equalization filters to optimize eye masks or minimize mean square error in Section 3.4. Section 3.5 compares our LP based method with the more commonly used least squares approach and shows that our LP approach achieves greater eye height while providing greater control of filter overdrive and overshoot at the receiver.

3.1 Eye Diagrams and Parameterized Eye Masks

Eye Diagrams are the most commonly used method to measure signal integrity. Figure 3.1 illustrates how an eye diagram is formed by overlaying a signal waveform over multiple symbol periods. It is called an eye diagram because the shape of the open region resembles an eye. An eye diagram provides a straightforward visual indication of how much voltage and timing margin is available for the receiver to correctly sample the signal. The eye opening represents the time during which we can safely sample the signal with fidelity. The height and width of the eye opening are often used to quantify signal integrity. We define the eye height as the maximum distortion (overshoot as well as undershoot) of the signal at the sampling instant. We write η to indicate that all logically high signals attain a level of at least +target $-\ \eta$, and all low signals attain of at most −target $+\ \eta$. The eye height is

$$height \quad = \quad \frac{\text{target} - \eta}{\text{target}} * 100\% \tag{3.1}$$

The eye width is the length of the time interval that the separation between high-going and low-going signals is greater than zero. This interval is indicated with the double-arrow labelled w in Figure 3.1, and we write

$$width \quad = \quad \frac{w}{P} * 100\% \tag{3.2}$$

where P is the bit-period, i.e. the time separation from one bit to the next. The eye height and width are both determined by the worst-case outlier signals. For example, the eye height is determined by the greatest overshoot or undershoot of all possible signals at the sampling instant. Hence, eye diagrams provide worst-case measure of signal integrity.

Intuitively, if a signal has greater eye height, then it is easier to distinguish high values from low ones. Likewise, greater eye width suggests greater tolerance of jitter in the signal timing or measurement instant. However, it is possible to have a channel with large eye height and width and poor signal integrity. For example, if the channel has large overshoot, then an otherwise large eye-opening can be a small fraction of the total input voltage range (see Figure 3.7B). Large excursions could degrade performance of the receiver's input circuitry and are indicative of excess power consumption. Similarly, with ringing, a channel can have a large eye height that decreases very rapidly in the neighborhood of the measurement point while still maintaining large eye width (see Figure 3.2). Such a channel is very sensitive to timing jitter.

In practice, any electrical interface requires certain voltage and timing margins. For a particular electrical interface, a minimum criteria window with width equal

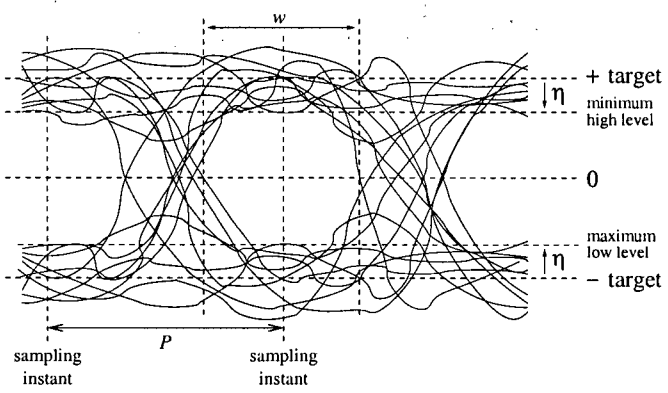


Figure 3.1: An eye diagram.

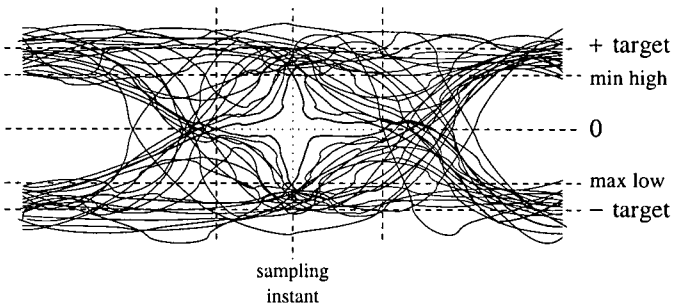


Figure 3.2: An eye diagram with large eye height and width but poor signal integrity.

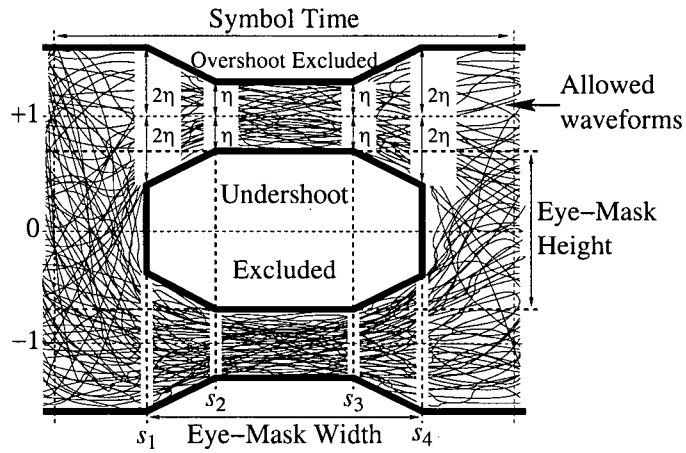


Figure 3.3: A parameterized eye mask.

to its required timing margin and height equal to its required voltage margin is known as the eye mask [11]. If the received signal has an eye opening large enough to contain the eye mask, then the signal has adequate margins for the particular electrical interface to correctly interpret the received data.

We now show how the eye masks can be generalized and parameterized to allow optimization. Figure 3.3 shows such a mask. Here, an eye mask is a set of polygons that specify constraints on the possible waveforms of a signal during each symbol period by excluding regions of undershoot and overshoot. We parameterize these polygons by specifying the vertices. In Figure 3.3, we specify constraints on overshoot and undershoot at four different sample times. This particular eye mask recognizes the transition slope and allows two times more distortion at the transition samples than the central samples of the eye where the sampling instant most likely happens. Note that the eye mask shown in Figure 3.3 restricts not only the voltage distortion during sample interval, but also overshoots during transitions.

Mathematically, we represent an eye mask as a set L of undershoot constraints and U of overshoot constraints. We can define L as a set of (s, η_s) pairs such that for all input patterns where the transmitted value for the wire and sample period of interest is $+1$, the received value must be at least $1 - \eta_s$ at sample time s. The overshoot constraints U is interpreted in the corresponding manner. A simple

objective function for optimization is,

$$\min_{f} \sum_{\forall(s,\eta_s)\in L,U} w_s\eta_s \tag{3.3}$$

where f is the filter coefficient space and w_s is the weight assigned to sample time s. Larger values of w_s correspond to greater weight at sample time s. Usually, one would expect to assign larger weight to the central samples.

For the particular parameterized eye mask shown in Figure 3.3, η_s is given as $\alpha_s\eta$. Pairs of the form (s, α_s) define the shape of the eye mask. The undershoot and overshoot constraints are interpreted similarly as above. For this particular parameterization of the eye masks, minimizing η optimizes the eye mask. For simplicity, we use this eye mask for all examples presented in this thesis, unless otherwise indicated.

3.2 Simplifying Assumptions

To simplify the presentation of our filter synthesis methods, we use the following assumptions:

1. The response of the channel is linear and can be approximated accurately by a finite length impulse response.

2. Each channel is used to convey binary (i.e. two-level) data. The input high and low levels are $+1$ and -1. Likewise, the output target levels are $+1$ and -1.

3. The high and low portions of the eye mask are symmetric, and the same eye mask is used for every channel.

The first assumption pertains to the physical interconnect and holds for most practical off-chip applications. The last two simplify the presentation. Extending the methods presented here to other signaling levels, other eye-mask shapes, etc., is straightforward. We will revisit these assumptions in Section 4.4.

3.3 Matrix Formulations for Convolution

Linear programming and least squares optimization problems are naturally formulated with matrices. The responses of filters, buses, and other multi-input, multi-output (MIMO) systems are naturally formulated as convolutions. Convolution is a

linear operator and therefore has a matrix representation. This section shows how we translate the impulse responses and convolutions of the signal processing world into the language of numerical optimization, matrices and vectors. Section 3.3.1 introduces the simple case of scalar, i.e. single-input, single-output systems. Section 3.3.2 generalizes this to the MIMO systems that are the subject of this thesis.

3.3.1 Scalar Convolution

Let $v \in \mathbb{R}^n$ be a vector of length n. The n components of v are $v(0)$, $v(1)$, ..., $v(n-1)$. We write $\|v\|$ to denote the l_2 norm of v, and $\|v\|_k$ to denote the l_k norm of v. For the sake of calculating matrix-vector products, all vectors in this thesis are column vectors unless otherwise stated. To simplify the upcoming formulas, for $v \in \mathbb{R}^n$, we define $v(i) = 0$ for all $i < 0$ and all $i \geq n$. Let v_1 and v_2 be vectors of lengths n_1 and n_2 respectively. The **convolution** of v_1 and v_2 is written $v_1 \times v_2$ as defined below:

$$(v_1 \times v_2)(i) = \sum_{j=0}^{i} v_1(i-j)v_2(j) \tag{3.4}$$

To express convolution in matrix form, let $n = n_1 + n_2 - 1$ and $v_1^{\times n} \in \mathbb{R}^{n \times n}$ be the matrix with

$$v_1^{\times n}(i,j) = v_1(i-j) \tag{3.5}$$

Pictorially,

$$v_1^{\times n} = \begin{bmatrix} v_1(0) & & & & & & & \\ v_1(1) & v_1(0) & & & & & & \mathbf{0} \\ v_1(2) & v_1(1) & v_1(0) & & & & & \\ \vdots & \vdots & \vdots & \ddots & & & & \\ v_1(n_1-1) & \vdots & \vdots & \vdots & \ddots & & & \\ & \ddots & \vdots & \vdots & \vdots & \ddots & & \\ & & \ddots & \vdots & \vdots & \vdots & \ddots & \\ & & & \ddots & \vdots & \vdots & \vdots & \ddots \\ \mathbf{0} & & & & v_1(n_1-1) & \cdots & v_1(2) & v_1(1) & v_1(0) \end{bmatrix} \tag{3.6}$$

The convolution of v_1 and v_2 in matrix form is:

$$v_1 \times v_2 = v_1^{\times n} \, \mathsf{extend}_0(n_2, n) v_2 \tag{3.7}$$

where $\mathsf{extend}_0(n_2, n)$ is the matrix that pads v_2 with zero elements to produce a vector of size n. The zero-extension matrix is defined as follows:

$$\mathsf{extend}_0(n_1, n) = \begin{bmatrix} I_{n_1 \times n_1} \\ 0_{(n-n_1)\times n_1} \end{bmatrix} \tag{3.8}$$

where $I_{n_1 \times n_1}$ denotes a $n_1 \times n_1$ identity matrix and $0_{(n-n_1)\times n_1}$ denotes a $(n - n_1) \times n_1$ zero matrix.

It follows directly that

$$v_0 \times v_1 \times \cdots \times v_{k-1} = v_0^{\times_n} v_1^{\times_n} \cdots \mathsf{extend}_0(n_{k-1}, n) v_{k-1} \tag{3.9}$$

where $n = \left(\sum_{i=0}^{k-1} n_i\right) + 1 - k$, and n_i is the length of vector v_i.

3.3.2 MIMO Convolution

Our formulation for MIMO convolution is based on the formulation for linear convolution presented above. We address two issues that are particular to the MIMO formulation.

- The impulse response of a MIMO system is a sequence of matrices rather than a sequence of scalars. Thus, the convolution matrix for a channel consists of blocks, where each block is a matrix corresponding to the response of the channel for a particular delay.
- For scalar convolution, the stimulus and the channel impulse response are mathematically interchangeable. Both are represented by vectors, and scalar convolution is commutative: one can apply the signal to the channel or the channel to the signal and produce the same result. For MIMO channels, this duality no longer holds. As noted above, the impulse response of a channel is a sequence of matrices. The input stimulus on the other hand, is a sequence of vectors. Each such vector gives the value of input on each line at a particular time.

We address these two issues and present our matrix formulation for MIMO convolution below.

Let

$$H = \begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ H_{n-1} \end{bmatrix} \tag{3.10}$$

denote the impulse response of a MIMO channel where $H_t(i,j)$ is the response at the i^{th} output after a delay of t time steps to an impulse stimulus at the j^{th} input. We adopt the convention that $H_t = 0$ for $t < 0$ or $t \geq n$.

We write the input as a vector so that we can model the channel as a matrix, and compute the output of the channel as their product. In particular, we define an input vector In, such that $\mathsf{In}(wt+j)$ is the value at time t at the j^{th} input, where w is the number of inputs of the MIMO channel.

For simplicity, we assume that the number of inputs and outputs are the same. The extension to channels with different numbers of inputs and outputs is straightforward. Now consider a channel of width w (w inputs and w outputs), and an impulse response of n_1 time steps. Let $H \in \mathbb{R}^{n_1 w \times w}$ be the impulse response of the channel as described above. Likewise, let $\mathsf{In} \in \mathbb{R}^{n_2 w}$ represent an input of width w and n_2 time steps. Let $H^{\times_n} \in \mathbb{R}^{nw \times nw}$ be the matrix with

$$H^{\times_n}(i,j) = H_{(i \operatorname{div} w)-(j \operatorname{div} w)}(i \bmod w, j \bmod w) \tag{3.11}$$

Pictorially,

$$H^{\times_n} = \begin{bmatrix} H_0 & & & & & & & \\ H_1 & H_0 & & & & & & \mathbf{0} \\ H_2 & H_1 & H_0 & & & & & \\ \vdots & \vdots & \vdots & \ddots & & & & \\ H_{n_1-1} & \vdots & \vdots & \vdots & \ddots & & & \\ & \ddots & \vdots & \vdots & \vdots & \ddots & & \\ & & \ddots & \vdots & \vdots & \vdots & \ddots & \\ & & & \ddots & \vdots & \vdots & \vdots & \ddots \\ \mathbf{0} & & & & H_{n_1-1} & \cdots & H_2 & H_1 & H_0 \end{bmatrix} \tag{3.12}$$

The response of the channel to In is

$$\mathsf{Out} = H^{\times_n} \mathsf{extend}_0(n_2 w, nw)\mathsf{In} \tag{3.13}$$

where $n = n_1 + n_2 - 1$ and $\mathsf{extend}_0(n_2 w, nw)$ is the matrix that pads In with zero elements to produce a vector of size nw. The vector Out in equation 3.13 has the same basic structure of that of In: $\mathsf{Out} \in \mathbb{R}^{nw}$, and $\mathsf{Out}(tw+j)$ is the response at time t at the j^{th} output.

Let $M \in \mathbb{R}^{n_1 w \times w}$ and $W \in \mathbb{R}^{n_2 w \times w}$ be the impulse response matrix for two channels. Let $\mathsf{In} \in \mathbb{R}^{n_{\mathsf{In}} w}$ be a stimulus applied to the first channel M, with the

output of M applied as the input to the second channel W. It follows directly from equation 3.13 that, Out, the output from W, is given by

$$\mathsf{Out} = W^{\times_n} M^{\times_n} \mathsf{extend}_0(n_{\mathrm{In}} w, nw)\mathsf{In} \tag{3.14}$$

where $n = n_1 + n_2 + n_{\mathrm{In}} - 2$. Note that we have defined n to be large enough to ensure that $M^{\times_n}$ $\mathsf{extend}_0(n_{\mathrm{In}} w, nw)\mathsf{In}$ has enough trailing zeros to allow proper convolution when multiplied by $W^{\times_n}$. The extension to the concatenation of three or more channels is straightforward.

It is important to note that linear convolution for MIMO is associative but generally not commutative. It is commutative if the products of the submatrices are commutative, for example, if the sub-matrices are all symmetric or all circulant [26, Section 24.2].

3.4 Optimal Pre-equalizer Synthesis

This section describes how to synthesize pre-equalization filters to optimize eye masks. This procedure supports arbitrary, linear bus models. Thus, each wire can have a different model and parasitic effects of bonding wires, vias, connectors, terminators, etc. can be taken into account.

Figure 3.4 shows the structure of a typical channel with a pre-equalization filter. A filter is assigned to each wire of the bus. Each filter takes as input a data bit and one or more neighboring bits in each direction and outputs a pre-distorted signal for one wire of the bus. Thus, the collection of filters form a multiple-input, multiple-output (i.e. MIMO) filter. In addition to reducing crosstalk, the pre-equalizer can also compensate for other linear distortions of the bus, including high-frequency attenuation due to resistive and dielectric losses.

Implementing a filter where every output depends on the values received on every input wire would consume a large amount of chip area and introduce unacceptable latency, especially for large buses. Noting that the largest contributions to crosstalk typically come from nearby wires, we consider filters where each output is computed from the input value for the wire itself and each of its w_{fir} closest neighbors in both directions[1]. For example, the filter shown in Figure 3.4 is a design with $w_{\mathrm{fir}} = 2$.

To compensate for high-frequency losses, equalization filters typically have tap rates that are a small multiple of the data rate. Furthermore, note that we can only specify eye-mask constraints on a limited number of sample times. There are no constraints for the signals in between the sample times. To include high frequencies

[1] Appendix C summarizes the definitions of all variables that are used globally in the thesis.

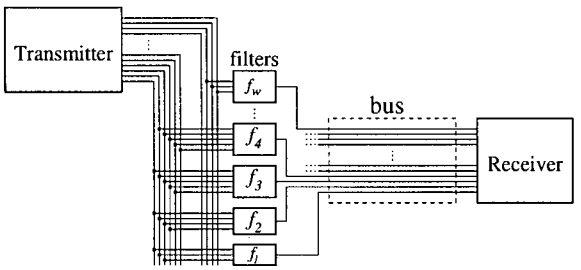


Figure 3.4: A typical channel with pre-equalization filters for crosstalk cancellation.

that might cause ringing, the sample rate for our analysis is a small multiple of the filter's tap rate. Moreover, in practice, the high-frequency roll-off of the channel and the limited slew rate of the transmitter limit the amount of high-frequency components. From experience, we get well-behaved results if the sample rate is twice the filter's tap rate.

We write r_{tap} to denote the number of sample points used in the impulse response functions per filter tap time and r_{bit} to denote the number of sample points per bit. Figure 3.5 shows these three time scales: bit, tap, and sample. As depicted in the figure, we write δ_0 for the delay from the input of the filter to the output of the bus. The examples in this thesis use a delay slightly greater than the LC delay of the bus for δ_0. In the remainder of this thesis, we write n_x to indicate the number of bit-times for quantity x, m_x for the number of tap times, and q_x for the number of sample times with the relation:

$$q_x \;=\; r_{\text{tap}} m_x \;=\; r_{\text{bit}} n_x \tag{3.15}$$

For example, while m_{fir} denotes the number of taps in the filter, $q_{\text{fir}} = r_{\text{tap}} m_{\text{fir}}$ gives the length of the filter impulse response in sample times. In the examples presented in this chapter, we use $r_{\text{tap}} = 4$ and $r_{\text{bit}} = 8$ (i.e. 2 samples per tap). We define $\mathbb{M}_{\text{bit}}$ as the set of integer multiples of r_{bit}.

As shown in Section 3.1, minimizing η optimizes the eye mask. The worst-case waveform from all possible input patterns for the channel determines η. To find the worst-case input pattern for the channel, we consider the response to each transmitted bit from each wire separately. Section 3.4.1 derives the channel output as a linear function of the filter coefficients. We then use this formulation in Section 3.4.2 to formulate the eye-mask optimization problem as a linear program.

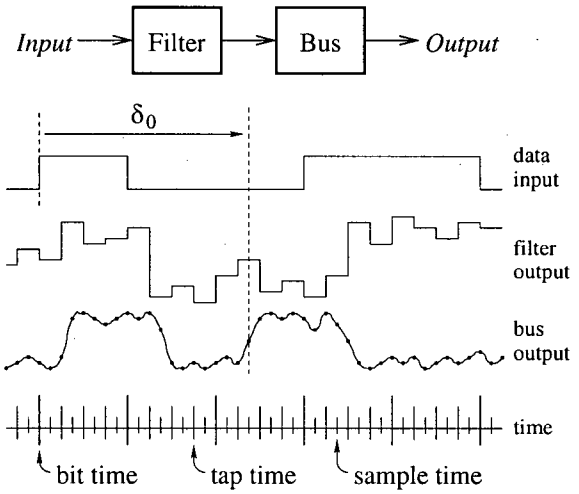


Figure 3.5: Bit, tap, and sample times.

3.4.1 The Bus Output is Linear in the Filter Coefficients

Because the filter and the bus are both linear, for any fixed input and bus impulse response, the output from the bus is a linear function of the filter coefficients.

Let w_{bus} denote the width of the bus. Let $\mathsf{B} \in \mathbb{R}^{q_{\text{bus}} w_{\text{bus}} \times w_{\text{bus}}}$ be the impulse response of the bus, where q_{bus} denotes the length of the bus impulse response in sample times. Input signals and filter coefficients are naturally given in bit times and tap times respectively. Because the analysis is done in sample times, oversampling is needed to convert the input (resp. filter coefficients) from a sequence in bit times (resp. tap times) to a sequence in sample times. To simplify the presentation, we first assume that bit times, tap times and sample times are the same. After we derive the bus output as a linear function of the filter coefficients, we take oversampling into account in the final formulation.

Let $\mathsf{F} \in \mathbb{R}^{q_{\text{fir}} w_{\text{bus}} \times w_{\text{bus}}}$ be the impulse response of the filter, i.e. the filter coefficient matrix. F consists of q_{fir} blocks where block $\mathsf{F}_t \in \mathbb{R}^{w_{\text{bus}} \times w_{\text{bus}}}$ gives the coupling from the input of the filter to the output after a delay of t time units. We consider filters where each output is computed from the input values for the wire itself and each of its w_{fir} closest neighbors in both directions. Hence, each F_t is a band matrix with width $2w_{\text{fir}} + 1$. To simplify the presentation, we assume full filters, i.e. each filter output is computed from input values for all wires. Based on this formulation, optimizing for narrower filters is straightforward. Let $\mathsf{In} \in \mathbb{R}^{q_{\text{In}} w_{\text{bus}}}$ be

the input vector. From equation 3.14, the output of the bus, Out, is given by:

$$\mathsf{Out} = \mathsf{B}^{\times_n}\, \mathsf{F}^{\times_n}\, \mathsf{extend}_0(q_{\mathrm{In}} w_{\mathrm{bus}}, n w_{\mathrm{bus}})\mathsf{In} \tag{3.16}$$

where $n = q_{\mathrm{bus}} + q_{\mathrm{fir}} + q_{\mathrm{In}} - 2$. Equation 3.16 shows that for a given input, In, and bus impulse response, B, the output of the channel, Out, is a linear function of the filter coefficients.

LP constraints are generally formulated as $Ax \leq b$ where A is the constraint matrix, b is a constant vector and x is a vector of LP variables which in this case are the filter coefficients. Thus, we need to swap In and F in equation 3.16. This requires formulating F as a vector and In as a matrix.

Let f be the vector of size $w_{\mathrm{bus}}^2 q_{\mathrm{fir}}$ of filter coefficients such that $f(tw_{\mathrm{bus}}^2 + iw_{\mathrm{bus}} + j)$ denotes the contribution of the input on wire j to the filter output for wire i after a delay of t time units. Let f_t denote the column vector of length w_{bus}^2 for the filter coefficients for delay t. In particular, $f(tw_{\mathrm{bus}}^2 + iw_{\mathrm{bus}} + j) = f_t(iw_{\mathrm{bus}} + j) = \mathsf{F}_t(\mathsf{i},\mathsf{j})$. Likewise, let In_t be the column vector of size w_{bus}, where $\mathsf{In}_t(j)$ is the input on the wire j at time t, i.e. $\mathsf{In}_t(j) = \mathsf{In}(tw_{\mathrm{bus}} + j)$. Let IN_t be the block diagonal matrix of size $w_{\mathrm{bus}} \times w_{\mathrm{bus}}^2$, where all of the diagonal blocks are identical row vectors, In_t^T, of length w_{bus}. Let $\mathsf{IN} \in \mathbb{R}^{q_{\mathrm{In}} w_{\mathrm{bus}} \times w_{\mathrm{bus}}^2}$ be the matrix consisting of a column of q_{In} matrices with IN_t as its t^{th} submatrix. By this construction,

$$\mathsf{F}_i \cdot \mathsf{In}_j = \mathsf{IN}_j \cdot f_i \tag{3.17}$$

Hence,

$$\begin{aligned} (\mathsf{F} \times \mathsf{In})(i) &= \textstyle\sum_{j=0}^{i} \mathsf{F}_{i-j}\mathsf{In}_j \\ &= \textstyle\sum_{j=0}^{i} \mathsf{IN}_j f_{i-j} \end{aligned} \tag{3.18}$$

In matrix representation,

$$\begin{aligned} \mathsf{Out} &= \mathsf{B}^{\times_n}\, \mathsf{IN}^{\times_n}\, \mathsf{extend}_0(q_{\mathrm{fir}} w_{\mathrm{bus}}^2, n w_{\mathrm{bus}}^2) f \\ &= \mathsf{G}_0(\mathsf{B}, \mathsf{In}) f \end{aligned} \tag{3.19}$$

where $n = q_{\mathrm{bus}} + q_{\mathrm{fir}} + q_{\mathrm{In}} - 2$ and $\mathsf{G}_0(\mathsf{B}, \mathsf{In}) = \mathsf{B}^{\times_n}\, \mathsf{IN}^{\times_n}\, \mathsf{extend}_0(q_{\mathrm{fir}} w_{\mathrm{bus}}^2, n w_{\mathrm{bus}}^2)$ is a $n w_{\mathrm{bus}} \times q_{\mathrm{fir}} w_{\mathrm{bus}}^2$ matrix determined by the bus impulse response, B, and the input sequence In.

We now consider oversampling. We write $\mathsf{In}_{\mathrm{bit}}$ to denote the input to the channel quantized in bit times and $\mathsf{In}_{\mathrm{sample}}$ to denote the input quantized in sample times. Thus, $\mathsf{In}_{\mathrm{bit}}(w_{\mathrm{bus}} t + i)$ gives the input signal on wire w_i at bit-time t. The r_{bit} oversample of $\mathsf{In}_{\mathrm{bit}}$ is given by,

$$\mathsf{In}_{\mathrm{sample}} = \mathsf{oversample}(n_{\mathrm{In}}, r_{\mathrm{bit}}, w_{\mathrm{bus}}) \cdot \mathsf{In}_{\mathrm{bit}} \tag{3.20}$$

where $\mathsf{oversample}(n_{\mathrm{In}}, r_{\mathrm{bit}}, w_{\mathrm{bus}})$ is a $r_{\mathrm{bit}} n_{\mathrm{In}} w_{\mathrm{bus}} \times n_{\mathrm{In}} w_{\mathrm{bus}}$ matrix given by

$$\mathsf{oversample}(n_{\mathrm{In}}, r_{\mathrm{bit}}, w_{\mathrm{bus}})(i,j) = \begin{cases} 1 & \text{if } (i \text{ div } (w_{\mathrm{bus}} r_{\mathrm{bit}})) = (j \text{ div } w_{\mathrm{bus}}) \\ & \text{and } (i \bmod w_{\mathrm{bus}}) = (j \bmod w_{\mathrm{bus}}) \\ 0 & \text{otherwise} \end{cases}$$

In this equation, the clause $(i \text{ div } (w_{\mathrm{bus}} r_{\mathrm{bit}})) = (j \text{ div } w_{\mathrm{bus}})$ matches sample times with the corresponding bit times, and the clause $(i \bmod w_{\mathrm{bus}}) = (j \bmod w_{\mathrm{bus}})$ matches each wire with itself. The $\mathsf{oversample}(n_{\mathrm{In}}, r_{\mathrm{bit}}, w_{\mathrm{bus}})$ matrix is block diagonal and each of its diagonal blocks is a column matrix consisting of r_{bit} identity matrices of size $w_{\mathrm{bus}} \times w_{\mathrm{bus}}$. Pictorially,

$$\mathsf{oversample}(n_{\mathrm{In}}, r_{\mathrm{bit}}, w_{\mathrm{bus}}) = \begin{bmatrix} I_{w_{\mathrm{bus}} \times w_{\mathrm{bus}}} & & & \mathbf{0} \\ I_{w_{\mathrm{bus}} \times w_{\mathrm{bus}}} & & & \\ \vdots & & & \\ I_{w_{\mathrm{bus}} \times w_{\mathrm{bus}}} & & & \\ & I_{w_{\mathrm{bus}} \times w_{\mathrm{bus}}} & & \\ & I_{w_{\mathrm{bus}} \times w_{\mathrm{bus}}} & & \\ & \vdots & \ddots & \\ & I_{w_{\mathrm{bus}} \times w_{\mathrm{bus}}} & \ddots & \\ & & \ddots & \\ & & \ddots & I_{w_{\mathrm{bus}} \times w_{\mathrm{bus}}} \\ & & & I_{w_{\mathrm{bus}} \times w_{\mathrm{bus}}} \\ \mathbf{0} & & & \vdots \\ & & & I_{w_{\mathrm{bus}} \times w_{\mathrm{bus}}} \end{bmatrix} \tag{3.21}$$

where $I_{w_{\mathrm{bus}} \times w_{\mathrm{bus}}}$ denotes an identity matrix of size $w_{\mathrm{bus}} \times w_{\mathrm{bus}}$. Oversampling the filter coefficient vector from tap times to sample times is done similarly. Note that oversampling is a linear operator. Taking oversampling into account, equation 3.19 becomes

$$\begin{aligned} \mathsf{Out} &= \mathsf{G}_0\,(\mathsf{B}, \mathsf{oversample}(n_{\mathrm{In}}, r_{\mathrm{bit}}, w_{\mathrm{bus}})\mathsf{In}_{\mathrm{bit}})\,\mathsf{oversample}(m_{\mathrm{fir}}, r_{\mathrm{tap}}, w_{\mathrm{bus}}^2) f \\ &= \mathsf{G}(\mathsf{B}, \mathsf{In}_{\mathrm{bit}}) f \end{aligned} \tag{3.22}$$

where $\mathsf{G} = \mathsf{G}_0\,(\mathsf{B}, \mathsf{oversample}(n_{\mathrm{In}}, r_{\mathrm{bit}}, w_{\mathrm{bus}})\mathsf{In}_{\mathrm{bit}})\,\mathsf{oversample}(m_{\mathrm{fir}}, r_{\mathrm{tap}}, w_{\mathrm{bus}}^2)$. Similar to oversampling, it is straightforward to use matrix multiplication to extend a narrow filter with width w_{fir} to a full filter with width w_{bus}. Hence, for a given input sequence and bus impulse response, the bus output is a linear function of the filter coefficients represented by the matrix G.

3.4.2 Eye-Mask Optimization as an LP

As described in Section 3.1, eye masks specify constraints that must hold for all possible input patterns; in other words, eye masks specify constraints on the *worst-case* waveforms. This is an l_∞ optimization problem that can be solved by linear programming. This section describes how we formulate the filter synthesis problem for optimal eye masks as a linear programming problem.

To optimize the parameterized eye mask for a channel, we need to determine its worst-case input pattern over all possible input sequences. To find this worst-case input pattern for the channel, we consider the response to each transmitted bit from each wire separately. Let bit_i be a vector that represents a one-bit wide pulse on wire i starting at time 0 with all other wires held constant at 0. For any pair of wires, $i, j \in [0 \ldots w_{\text{bus}} - 1]$, and any sampling time, $s \in [0 \ldots (n_{\text{bus}} + n_{\text{fir}} + 1)r_{\text{bit}}]$, let $g(i, j, s)$ be the vector such that $g(i, j, s)^T f$ is the response on wire j at time s to input bit_i given filter f. Using equation 3.22, the g vectors are readily derived from the impulse response of the bus:

$$g(i, j, s)^T = \mathsf{G}(\mathsf{B}, \mathsf{bit}_i)(j + sw_{\text{bus}}) \tag{3.23}$$

That is, row $j + sw_{\text{bus}}$ of $\mathsf{G}(\mathsf{B}, \mathsf{bit}_i)$ is $g(i, j, s)^T$. Moreover, in this case, column $i + jw_{\text{bus}}$ of $\mathsf{B}^{\times n}\,\mathsf{IN}^{\times n}$ in equation 3.19, is the bit response on wire j to an input bit at time 0 on wire i. Hence, the g vectors can be derived from the bit responses of the bus, which is much easier to obtain than the bus's impulse response in real physical settings.

By the assumption that the channel is linear, the received signal is a linear combination of these bit responses:

$$\mathsf{Out}(j, s) = \sum_{i=1}^{w_{\text{bus}}} \sum_{k \in \mathbb{M}_{\text{bit}}} v(i, k) g(i, j, s + k)^T f \tag{3.24}$$

where $\mathsf{Out}(j, s)$ is the output on wire j at sample time s, and $v(i, k) \in \pm 1$ is the data value sent on wire i at sample time k. We now calculate the crosstalk and crosstalk-free components of the received signal. Without loss of generality, we consider the case where a value of $+1$ is sent on wire j at time 0; in other words, $v(j, 0) = +1$. Let

$$u(j, s) \;=\; g(j, j, s)^T\, f \tag{3.25}$$

The target delay of the channel is δ_0. Thus, $u(j, \delta_0) \ldots u(j, \delta_0 + r_{\text{bit}} - 1)$ gives the undisturbed response of the channel to the $+1$ bit sent on wire j at time 0.

The disturbances are the contributions from all other wires, and from the wire itself at other bit times. These can be determined from the bit responses of the bus

as well. Because the inputs are either $+1$ or -1, we can compute the responses for a $+1$ input bit on each wire at each bit time and sum the absolute values to get the maximum disturbance. In other words, if the disturbance produced by a $+1$ input on some wire and at some bit time is positive, then the value of this input will be $+1$ in the input pattern that creates the greatest total overshoot and -1 in the pattern that creates the greatest undershoot. Noting that the response to a -1 is simply the negation of the response to a $+1$, we find the worst-case disturbance by taking the absolute values of the responses to inputs of $+1$. Let $d_\Sigma(i,s)$ denote the maximum total disturbance on wire i at sample time s:

$$d_\Sigma(j,s) = \left(\sum_{i=1}^{w_{\text{bus}}} \sum_{k \in \mathbb{M}_{\text{bit}}} \left| g(i,j,k+s)^T f \right| \right) - |u(i,s)| \tag{3.26}$$

The minimum value that wire j can have at time s is $u(j,s) - d_\Sigma(j,s)$ and the maximum is $u(j,s) + d_\Sigma(j,s)$. The worst-case channel response at any given sample time and any wire should satisfy the undershoot and overshoot constraints specified by the parameterized eye mask.

The eye mask specifies constraints over a single bit period, and this specification is independent of the target delay. Thus, the eye-mask constraints are for times $0 \ldots r_{\text{bit}} - 1$, and constraint (s, α_s) pertains to the bus output at time $s + \delta_0$. We can now write the linear program for eye-mask optimization over a multi-wire bus:

$$\begin{aligned} &\min_f \eta \ \text{ s.t.} \\ &\quad \forall j \in [0 \ldots w_{\text{bus}} - 1], \\ &\qquad \forall (s, \alpha_s) \in L,\ u(j, s+\delta_0) - d_\Sigma(j, s+\delta_0) \geq 1 - \alpha_s \eta \\ &\quad \wedge \ \ \forall (s, \alpha_s) \in U,\ u(j, s+\delta_0) + d_\Sigma(j, s+\delta_0) \leq 1 + \alpha_s \eta \end{aligned} \tag{3.27}$$

Note that at the optimal vertex of the LP, the worst-case channel response is computed and the worst-case input pattern for the channel is identified. This useful byproduct of our approach allows us to test and measure the worst-case performance of the channel.

We implemented the LP optimization routine using MATLAB. LP constraints are generally formulated as $Ax \leq b$ where A is the constraint matrix, b is a constant vector and x is a vector of LP variables. To write the constraints in matrix form, we define matrices G_u and G_d such that $G_u f$ and $G_d f$ compute the undisturbed responses and disturbances respectively:

1. G_u is $r_{\text{bit}} w_{\text{bus}} \times k_{\text{fir}}$ and its rows are the set of $g(i,j,s)^T$ vectors with $i = j$ and $\delta_0 \leq s < \delta_0 + r_{\text{bit}}$,

2. G_d is $(w_{\text{bus}}^2 r_{\text{bit}}(n_{\text{fir}} + n_{\text{bus}}) - w_{\text{bus}} r_{\text{bit}}) \times k_{\text{fir}}$ and contains the rest of g vectors.

We write the LP as:

$$\min_{f,d,\eta} \eta \text{ s.t.}$$
$$\begin{bmatrix} -I & G_d & 0 \\ -I & -G_d & 0 \\ W & G_u & -\alpha \\ W & -G_u & -\alpha \end{bmatrix} * \begin{bmatrix} d \\ f \\ \eta \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} \tag{3.28}$$

The rows with the G_d matrices compute d, the absolute values of the disturbances. The rows with the G_u matrices compute the maximum undershoot and overshoot respectively: G_u computes the undisturbed responses; W computes the sum of the disturbance terms; and α is a column vector of scaling terms for each measurement point of the eye mask. We describe the detailed structure of these matrices and our implementation of numerical methods to solve these linear programs in Chapter 5.

3.4.3 MSE and Least-Squares Optimization

In order to compare our l_∞ methods with traditional l_2 techniques we implemented least-squares optimization for pre-equalizer synthesis. Minimizing mean square error requires a knowledge of the underlying distributions of the transmitted data. For simplicity, we assume each bit transmitted on each wire at each bit time is an independent, evenly weighted, Bernoulli random variable. We note that designers frequently make this assumption in practice because detailed statistical models are often unavailable. We model the bits as taking on values of ± 1; thus, each bit has a zero-mean.

With these assumptions, for each wire and sample time, each contribution to the distortion on that wire and time is an independent, mean-zero, random variable. Thus, the expected value of the square of the sum of the disturbances is equal to the expected value of the sum of the squares. From Section 3.4.2, we know that $G_d f$ gives the errors from the disturbances and $G_u f - e$ gives the errors from the "undisturbed" channel response, where e is a column vector of all ones.

Noting that the values of the samples within the measurement interval are usually much more critical than those of samples outside it, we assign a weight to each sample point in the bit period, $\Upsilon(s)$, where $\Upsilon(s)$ is periodic with period r_{bit}. Otherwise, if all sample times throughout the bit period are weighted equally, the least-squares optimizer will compromise eye-height trying to reduce the error during transitions. On the other hand, if transition samples are ignored, the optimizer will often find solutions with extreme overshoot during transitions. Weighting the sample points allows us to avoid such undesirable solutions.

The weighted, mean-square optimization problem is:

$$\min_f \left\| \begin{bmatrix} \Upsilon_u G_u \\ \Upsilon_d G_d \end{bmatrix} f - \begin{bmatrix} \Upsilon_u e \\ 0 \end{bmatrix} \right\| \tag{3.29}$$

where Υ_u is the diagonal matrix that scales rows of G_u with the weights corresponding to their sample times and likewise for Υ_d.

We used MATLAB's least-squares linear system routine mldivide (a.k.a. \) to solve the LSQ problem. We briefly examine the time and space requirements of this approach in Section 5.5.

3.5 Evaluation

To evaluate the filter design methods described in the previous section, we implemented the optimization routines using MATLAB [65]. For the least-squares based optimization, we used MATLAB's mldivide. For the linear-programming based optimization, we wrote our own implementation of Mehrotra's algorithm with a problem-specific linear system solver that exploits the sparsity structure of our constraint matrix (see Chapter 5). We used HSPICE [6] to derive bus models and their bit-response functions.

3.5.1 A Realistic Bus Model

In our previous work [54], we used a simplified mathematical model for buses where all wires were identical (i.e. a cylindrical bus). In the preceding sections, we have described filter synthesis that support realistic bus models where each wire has its own model, boundary effects are modelled, and the parasitic effects of bonding wires, vias, connectors, etc. can be taken into account.

For the results presented in this section, we used the field solver in HSPICE [6] to extract a model for microstrip lines in 1 Oz copper, 5 mil wide with 5 mil separation between lines, running above a ground plane with a dielectric thickness of 10 mil and a dielectric constant of 4.5. Each line is terminated with a resistor that matches the line's characteristic impedance. For simplicity, the bus is single-ended and unidirectional. We extend this approach to differential signalling and bi-directional links in Chapter 4.

With the extracted RLGC matrices, we used HSPICE simulations to obtain the responses on all wires to a single bit input on wire i for each wire of the bus. Using the results of these HSPICE simulations, our filter design procedure synthesizes the optimal filters and generates their corresponding worst-case input sequences.

3.5.2 Simulation Results

All of the results reported in this section are for a 32-bit bus (i.e. $w_{\text{bus}} = 32$) using the bit-response model extracted by HSPICE. All designs have four filter taps per bit-time and two sample times per tap time (i.e. $r_{\text{tap}} = 2$ and $r_{\text{bit}} = 8$). We write that a filter is $m_{\text{fir}} \times w_{\text{fir}}$ to indicate that the filter has m_{fir} taps and a width (number of neighbors considered to each side) of w_{fir}. Unless otherwise noted, we use a bit-time of 500ps (i.e. 2Gbits/sec/wire). To test our filters, we generated the worst-case input sequences for each filter [55], based on the observation that the worst-case disturbance occurs when all of the individual disturbances have the same sign. This is easily achieved by determining the disturbance caused by each input wire and bit-time and setting the sign of each input bit appropriately.

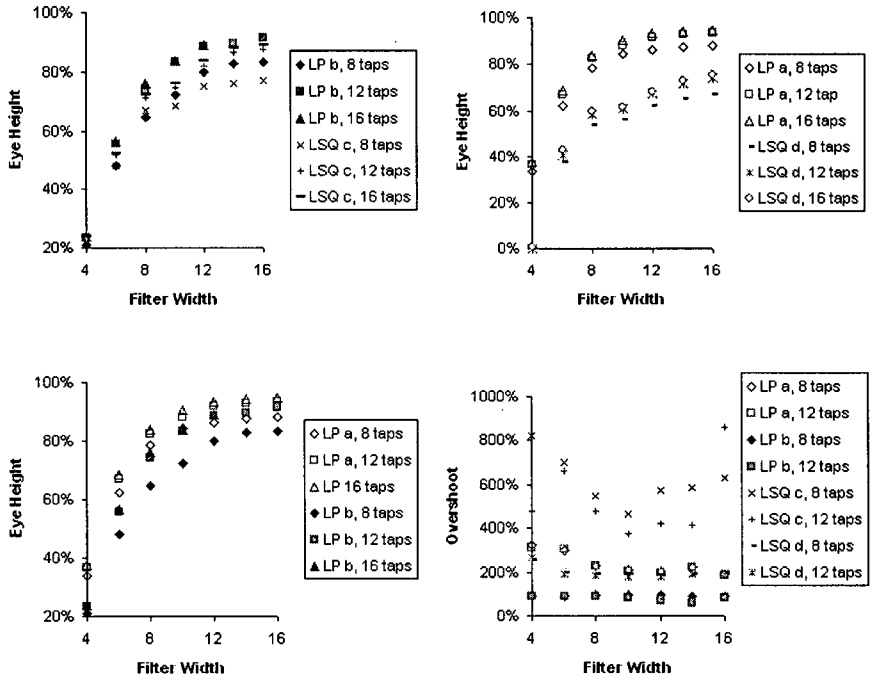
To obtain practical designs that achieve good crosstalk reduction, we must choose the width and length of the filter appropriately. In general, larger filters achieve better crosstalk cancellation at an increased cost for the hardware implementation. Figure 3.6 shows simulation results for filters with various lengths and widths and the following synthesis methods:

LP(a): The LP method using the eye mask from Figure 3.3. This mask has no constraints for sample times between measurement intervals (i.e. during transitions between bits). We constrained the magnitude of the filter output to be at most three times the target at all times.

LP(b): The same LP as above with further restrictions that the magnitude of the signals at the output of the bus must be less than twice the target level at all times. This reduces the worst-case overshoot with some attendant reduction in eye height.

LSQ(c): The LSQ method with weight $\Upsilon(\tau) = 1$ for the four sample points of the measurement interval and $\Upsilon(\tau) = 0$ for sample points between measurement intervals (see equation 3.29). This means that we only optimize for the four points of the measurement interval, and allow any response elsewhere. Moreover, the maximum magnitude of filter coefficients is not constrained; thus, the overdrive ability of the filter is unconstrained. This tilts our comparison in favor of the LSQ method by giving it more latitude than the LP approaches.

LSQ(d): The LSQ method with $\Upsilon(\tau) = 1$ for four points of the measurement interval and $\Upsilon(\tau) = 0.4$ for sample points between measurement intervals. This prevents extreme overshoot at transition taps. The maximum magnitude of filter coefficients remains unconstrained.



Filter design methods:

LP(a): linear programming method without constraints outside the eye mask;

LP(b): linear programming method with constraints constraining overshoot outside the eye mask;

LSQ(c): least-squares optimization without constraints for sample points between measurement intervals.

LSQ(d): least-squares optimization with weight 0.4 assigned for sample points between measurement intervals.

Figure 3.6: Performance of equalizing filters for a 32-bit bus.

Figure 3.7 shows eye diagrams for the bus with no equalization and with 12×10 filters synthesized by the LP and LSQ design methods.

We make several observations. First, the filters designed using the LP methods outperform their LSQ counterparts with the same width and length in all cases terms of eye height, overshoot, and other eye-mask parameters. This shows the advantage of optimizing the eye mask directly rather than approximating it with mean-square error. Second, the LP filters have much lower overshoot than their LSQ counterparts. Furthermore, the overshoot for the LP filters can be greatly reduced with only a small penalty in the eye height, and this penalty is smaller for larger filters. The overshoot of the LSQ filters can be reduced by including sample points outside of the measurement interval in the objective function. However, LSQ eye height decreases significantly as overshoot is reduced. For example, compared with no constraints on transition taps (LSQ(c)), by setting the weight to 0.4 for transition taps (LSQ(d)), the 12×10 filter designed by the LSQ method has a much smaller overshoot (189% vs. 463%). However, the eye height decreases from 75% (LSQ(c)) to 61% (LSQ(d)). Naturally, signal integrity improves as the filter width and length are increased, and the improvement decreases as the filter grows larger. For the bus and bit-rate considered, twelve-tap filters with a width, w_{fir}, of eight or ten appear to be promising designs. Conversely, narrow filters afford little improvement in signal integrity. We synthesized filters for independent pre-emphasis for each line ($w_{\text{fir}} = 1$), and simulated the bus with no filter. In all of these cases, the eye height was zero.

The filters designed by the LP method require much less overdrive from the filter than those designed by the LSQ method. For example, the maximum filter output for the LSQ 8×8 filter is 6.53 times the target value, whereas the maximum filter output for the 8×8 filters designed by the LP methods are 1.639 and 2.227 times the target value respectively.

Another way to evaluate the performance of an equalizing filter is its maximum operating bit rate. We define the maximum operating bit rate as maximum bit rate at which the height of the eye is slightly greater than 50% and the eye width is over 25%. The bus impulse response matrix B is different for different bit rates; hence, filter synthesis was done for each bit rate we examined. The maximum bit rate for the bus without a filter is 333MHz. Single line pre-emphasis using a 12 tap filter designed by the LP method increases the bandwidth to 500MHz. Nearest neighbor crosstalk cancellation ($w_{\text{fir}} = 1$) raises the bandwidth to 800MHz. With a 12×10 filter designed with the LP method, the channel has a maximum bit rate of approximately 2.5GHz; Figure 3.7(D) shows the eye diagram. With a 12×10 filter designed by the LSQ method, the system has a maximum bit rate at approximately 2GHz; the eye is completely closed at 2.5GHz.

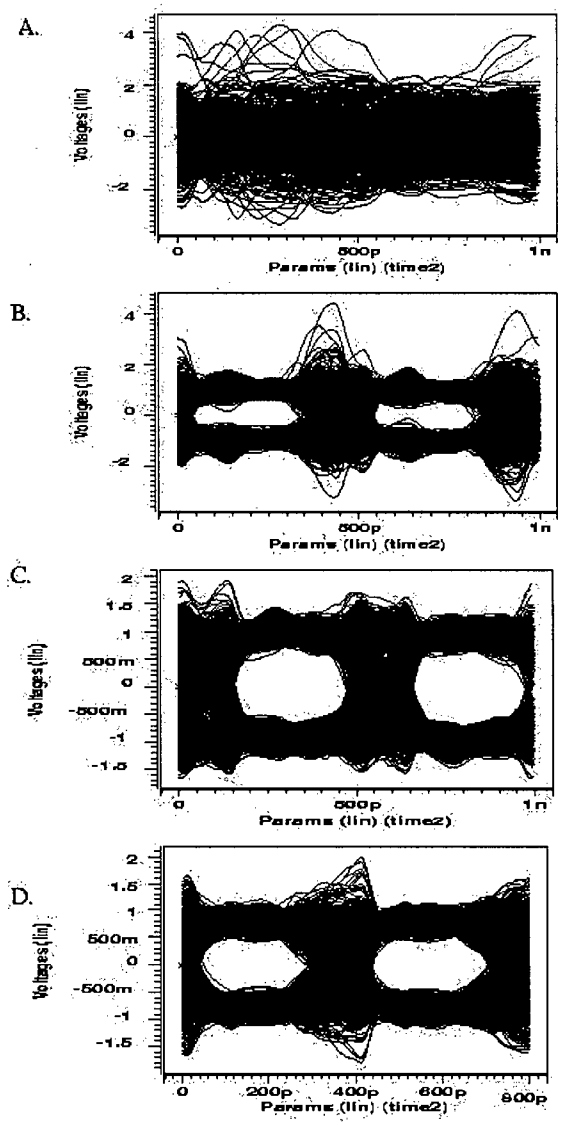


Figure 3.7: Eye diagrams at 2Gbits/sec/wire. A. without filter. B. with 12 × 10 equalizing filters designed by the LSQ method without constraints on transition taps. C. with 12 × 10 equalizing filters designed by the LP method with constraints on overshoot between measurement intervals. D. An eye diagram at 2.5Gbit/sec/wire with a 12 × 10 equalizing filter designed by the LP method with constraints on overshoot between measurement intervals.

3.6 Summary

In this chapter, we formulated objective functions for optimal pre-equalization synthesis using parameterized eye masks. These objective functions give the designer great flexibility for specifying trade-offs between eye height, eye width, and other details of the eye shape. We presented an LP-based approach for synthesizing optimal pre-equalization filters for crosstalk cancellation for off-chip buses. In Chapter 5, we will present a problem-specific LP solver to solve these LPs very efficiently.

Filters designed by our method can increase bus bandwidths by more than a factor of seven compared with a bus with no filters and more than a factor of three compared with a bus with only pre-emphasis and nearest neighbor crosstalk cancellation. The ability to directly specify critical parameters such as eye masks, output magnitude, and overshoot is a clear advantage of our approach. This is in contrast with the commonly used least-squares (LSQ) optimization techniques for filter design where error is minimized only in the average sense. Accordingly, filters designed with our method significantly outperform LSQ designed filters in terms of eye height, overshoot, and other eye-mask parameters.

Chapter 4

A Unified Optimization Framework

In Chapter 3, we presented crosstalk cancelling pre-equalizers for single-ended, unidirectional links with relatively simple channel models. This chapter extends this approach to jointly optimize pre-equalizers, decision-feedback equalizers and filters for near-end crosstalk cancellation. Moreover, this chapter addresses a comprehensive set of practical issues including the incorporation of differential and bidirectional links, and multi-level signalling. The examples in this chapter include packaging and connector parasitics, demonstrating the use of our methods with detailed models of realistic buses. We present simulation results for on-board links as well as backplane links connecting two daughter cards.

High-speed links often use pre-equalization by the transmitter, decision feedback equalization by the receiver, and near-end cross-talk cancellation for bidirectional links. Figure 4.1 shows a w-bit bidirectional link with differential signalling and these three forms of equalization. Each transceiver consists of a digital block that transmits and receives data, a pre-equalizer (PE), an equalizer for near-end crosstalk cancellation (NE), and a decision-feedback equalizer (DFE). We include the differential drivers and receivers in the channel model as well as the packages, PC board buses, and connectors that provide the connections between the communicating chips. Each of the three equalizers brings its own strengths and weaknesses to the system, and designing an optimal channel requires managing these trade-offs effectively. For example, the PE filter can improve far-end signal integrity by boosting the high-frequency components of the transmitted signal; however, this exacerbates near-end crosstalk and places greater demands on the NE filter. The DFE can correct for many of the same effects of crosstalk and ISI as the pre-equalizer, without incurring the PE's downsides of increased power consumption and near-end crosstalk. However, the DFE is sensitive to errors in the received data stream, and creates a tight feedback cycle that is not present in the other filters. Thus, practical designs make use of pre-equalization, DFE and near-end crosstalk cancellation. This chapter presents a method for synthesizing optimal combinations of pre-equalization, decision-feedback equalization and near-end crosstalk cancellation filters. We show that the synthesis problem remains a linear program.

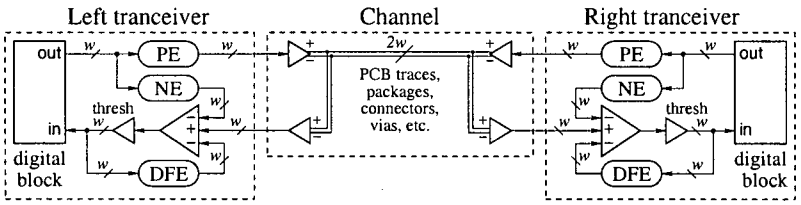


Figure 4.1: A bidirectional link with equalization filters.

As in the previous chapter, we begin by stating assumptions that simplify the presentation. Section 4.2 describes our mathematical programming formulation for optimizing signal integrity. In Section 4.3, we demonstrate our approach by examining the symbol rates at which various eye heights can be achieved for several channel configurations. We show how this allows us to quickly evaluate different designs for high-speed buses. Finally, Section 4.5 presents a possible hardware implementation of the filters and shows the area and latency costs of the filters are acceptable.

4.1 Simplifying Assumptions

We make the same assumptions as in Section 3.2 and add two more:

1. The channel is symmetric and transceivers on both ends have the same characteristics.

2. The equalized channel is error free, a.k.a, the inputs to the DFE match the transmitted data stream.

A symmetric channel means that the channel response from left to right is the same as the channel response from right to left. Extending this framework to the non-symmetric case is straightforward. The error-free assumption is a simplifying assumption that allows us to obtain a linear link model (see Section 4.2). We will revisit these assumptions in greater detail in Section 4.4.

4.2 Eye-Mask Optimization as an LP

This section shows how to jointly synthesize optimal pre-equalization (PE), decision feedback equalization (DFE) and near-end crosstalk cancellation (NE) filters

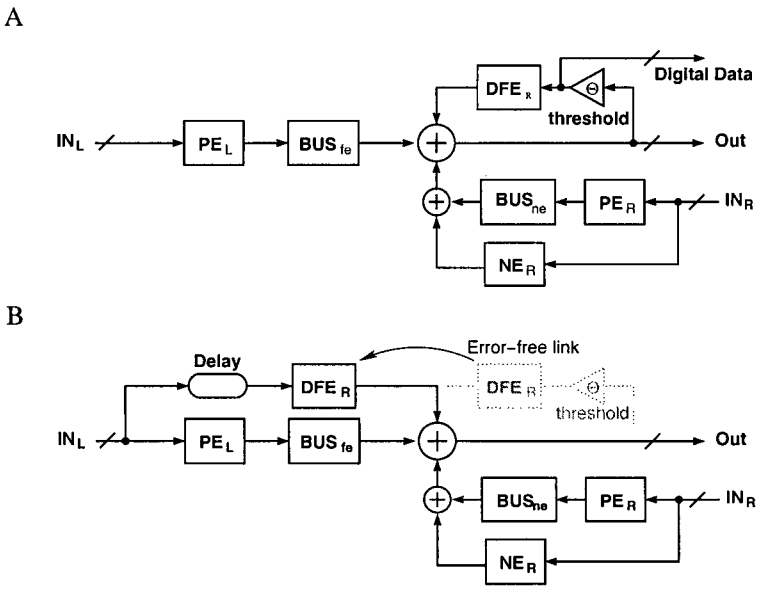


Figure 4.2: A. Block diagram for the link from left to right. B. Error-free assumption leads to a linear link model.

to maximize eye height, optimize eye masks and minimize near-end crosstalk for simultaneous bidirectional signalling over differential pairs.

As before, let r_{tap} be the number of sample times per tap time, and let r_{sym} be the number of sample times per symbol time (see Figure 3.5). Note that, for two-level signalling, the symbol rate is the same as the bit rate. In the examples in this thesis, we use $r_{\text{tap}} = 4$ and $r_{\text{sym}} = 8$ (i.e. two taps per symbol). We define $\mathbb{M}_{\text{sym}}$ as the set of integer multiples of r_{sym}.

Consider a link as shown in Figure 4.1. Each client transmits w_{link} symbols per symbol period. We call w_{link} the width of the link and say that the channel has w_{link} lines. Using differential signalling, the bus between the transceivers has $2w_{\text{link}}$ wires. In the following, we refer to each differential pair as simply a "pair"; thus, the bus has w_{link} pairs.

With the assumption that the channel is symmetric, we can restrict our attention to the channel from left to right. A block diagram for the left to right channel is shown in Figure 4.2A. In Figure 4.2, $\mathsf{Bus}_{\mathsf{fe}}$ to denote far-end impulse response of the bus: the response at Out given an impulse at In_{L}. Likewise, the near-end impulse response, $\mathsf{Bus}_{\mathsf{ne}}$ is the response at Out given an impulse at In_{R}.

Consider the signal received at Out, the input of the threshold circuit in the right transceiver. We exploit the linearity of the channel and filters to break the signal received at Out into several independent components. Let $\mathsf{Out}(j, s)$ be the value received on line j at time s. We define:

Out_{fe}: (the far end response) the data transmitted by the left-transceiver convolved with the impulse responses of the PE filter and the far-end response of the channel.

$\mathsf{Out}_{\text{dfe}}$: (the DFE filter output) The data received by the right-transceiver convolved with the impulse responses of the DFE filter.

$\mathsf{Out}_{\text{next}}$: (near end crosstalk) The data transmitted by the right-transceiver convolved with the near-end response of the channel.

Out_{ne}: (the NE filter output) The data transmitted by the right-transceiver convolved with the impulse response of the NE filter.

Thus, $\mathsf{Out} = \mathsf{Out}_{\text{fe}} + \mathsf{Out}_{\text{dfe}} + \mathsf{Out}_{\text{next}} + \mathsf{Out}_{\text{ne}}$. Note that, for a given input sequence, the received signal Out is not linear in the filter coefficients due to the feedback loop and the thresholding block (see Figure 4.2A). To obtain a linear model, we assume that the link is error free; thus, the received data stream matches the transmitted data stream. With this assumption, we conceptually break the feedback loop by using an equivalent model with a delayed version of transmitted data as the input to the DFE filter as shown in Figure 4.2B. With this modified model, the received signal Out is linear in all filter coefficients.

Let f_{pe}, f_{dfe}, and f_{ne} be column vectors of the filter coefficients for the PE, DFE, and NE filters respectively. Let $\mathsf{Out}_{\text{fe}}(j, s)$ be the component Out_{fe} for line j at time s. Consider a scenario where a value of $+1$ is output by the left-transceiver on line i for one symbol period starting at time 0 and a value of 0 is sent at all other times and on all other lines. We define a column vector $g_{\text{fe}}(i, j, s)$ such that for this scenario the response at Out on line j at time s is given by $g_{\text{fe}}(i, j, s)^T f_{\text{pe}}$. We call g_{fe} the far-end bit response of the channel. The vector g_{fe} is readily derived from bus impulse or bit responses (see Section 3.4.2). Let $v(i, s) \in \pm 1$ be the data value sent on line i at time s. By the assumption that the channel is linear, we have

$$\mathsf{Out}_{\text{fe}}(j, s) = \sum_{i=1}^{w_{\text{link}}} \sum_{k \in \mathbb{M}_{\text{sym}}} v(i, k) g_{\text{fe}}(i, j, s + k)^T f_{\text{pe}} \tag{4.1}$$

We define g_{dfe}, g_{next}, and g_{ne} in the equivalent manner. Note that g_{fe} and g_{next} depend on the response of the bus whereas g_{dfe} and g_{ne} do not. The convolutions

for calculating Out_{fe} and $\mathsf{Out}_{\text{dfe}}$ use data from the left-transceiver, whereas the convolutions for $\mathsf{Out}_{\text{next}}$ and Out_{ne} use data from the right.

Let δ_0 be the target delay for the channel. As in Section 3.4, we choose δ_0 to be slightly larger than the LC delay of the channel. Thus $v(i, s - \delta_0)$ is the desired value for $\mathsf{Out}(i, s)$. As in Section 3.4.2, we focus on the case where $v(i, s - \delta_0) = +1$, and express the response of the channel as an "undisturbed" component plus a sum of "disturbances." The undisturbed response, $u(j, s)$ is

$$u(j, s) \;=\; g_{\text{fe}}(j, j, \delta_0 + s)^T f_{\text{pe}} \tag{4.2}$$

We first consider disturbances arising from other bits sent by the left-transceiver including the contributions of the DFE filter. Choosing the signs of the other data bits to result in positive disturbances, the maximum far-end response, $\mathsf{Out}_{\text{far,max}}$, and maximum far-end disturbance, d_{fe}, are

$$\begin{aligned} \mathsf{Out}_{\text{far,max}}(j, s) &= \sum_{i=1}^{w_{\text{link}}} \sum_{k \in \mathbb{M}_{\text{sym}}} \left| g_{\text{fe}}(i, j, \delta_0 + s + k)^T f_{\text{pe}} + g_{\text{dfe}}(i, j, \delta_0 + s + k)^T f_{\text{dfe}} \right| \\ d_{\text{fe}}(j, s) &= \mathsf{Out}_{\text{far,max}}(j, s) - u(j, s) \end{aligned} \tag{4.3}$$

We now consider near-end interference. The contribution of $\mathsf{Out}_{\text{next}} + \mathsf{Out}_{\text{ne}}$ is purely a disturbance. By reasoning equivalent to the far-end case, we get

$$d_{\text{ne}}(j, s) \;=\; \sum_{i=1}^{w_{\text{link}}} \sum_{k \in \mathbb{M}_{\text{sym}}} \left| g_{\text{next}}(i, j, \delta_0 + s + k)^T f_{\text{pe}} + g_{\text{ne}}(i, j, \delta_0 + s + k)^T f_{\text{ne}} \right| \tag{4.4}$$

Here we used the assumption that the transceivers and channel are symmetric by assuming that f_{pe} is the pre-equalizer for the left transceiver in equations 4.2 and 4.3 and for the right in equation 4.4. This assumption could be removed by using separate filter coefficient vectors for the left and right transceivers.

We pessimistically assume that the left and right transceivers have independent clocks; thus, there is no fixed alignment between the period of the near-end disturbance and the received data. Instead, we assume that the worst-case near-end disturbance could happen at any time during the sampling interval. We overload d_{ne} and write $d_{\text{ne}}(j)$ for the worst case near-end disturbance on line j with

$$d_{\text{ne}}(j) \;=\; \max_{s \in [0 \ldots r_{\text{sym}}]} d_{\text{ne}}(j, s) \tag{4.5}$$

Equations 4.2, 4.3 and 4.5 yield the mathematical program for unified optimization for PE, DFE and NE filters for bidirectional signalling:

$$\begin{aligned}\min_{f_{\text{pe}}, f_{\text{dfe}}, f_{\text{ne}}} \ & \eta \text{ s.t.} \\ & \forall (s,\alpha) \in L.\ \forall j \in [1 \ldots w_{\text{link}}],\ \ 1 + d_{\text{fe}}(j,s) - u(j,s) + d_{\text{ne}}(j) \leq \alpha\eta \\ \wedge\ & \forall (s,\alpha) \in U.\ \forall j \in [1 \ldots w_{\text{link}}],\ \ d_{\text{fe}}(j,s) + u(j,s) - 1 + d_{\text{ne}}(j) \leq \alpha\eta \end{aligned} \tag{4.6}$$

We note that u, d_{fe}, and d_{ne} are all linear in the filter coefficients. Thus, this is a linear programming problem. The optimization problem for unidirectional signalling is the same with $d_{\text{ne}}(j) = 0$.

4.3 Evaluation

This section presents results using the filter synthesis approach described in the previous section for realistic channels. We evaluate the synthesis procedure by exploring trade-offs and limitations for high-speed off-chip buses.

4.3.1 Channel Models

We considered two basic channel configurations as shown in Figure 4.3: a 10 cm point-to-point interconnect between two chips on the same PC board; and an interconnect across a 50 cm backplane. For the inter-board link, we used a model provided by Teradyne for their eight-row, VHDM-HSD connector. We modelled each DAC output as linearly slewing from its old value to its new value over the tap period. This reflects the limited slew-rate of real DACs and avoids introducing unwanted high-frequency energy into the channel [23].

All PC board buses considered here use 6 mil traces with 8 mil spacing in 0.5 oz copper with ground planes on each side of the signal plane based on a design from Rambus [39]. We used the 2D field solver of HSPICE to obtain an electrical model for this bus. The bus has a 100Ω nominal differential impedance. We assumed a manufacturing tolerance of $\pm 10\%$ for the actual impedance, and extracted models with worst-case mismatches. All differential pairs are terminated with 100Ω resistors.

For our initial experiments, we used the ball grid array model from Dally and Poulton [18, p. 39]. Our initial experiments showed that the chip package was the critical bottleneck for both configurations due to the 5 nH of chip-to-package and another 5 nH of board-to-chip inductances. After consulting with designers of high-speed links in industry [29], we created a model where these inductances were reduced to 0.5 nH and reduced the package capacitances by a factor of 3.

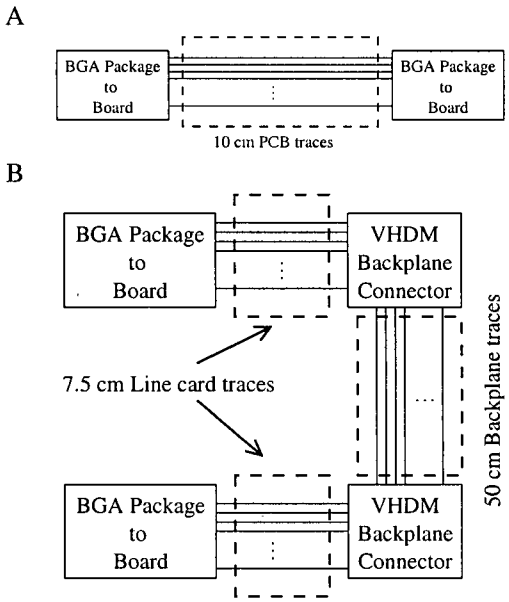


Figure 4.3: Off-chip channel models.

	Inter-board		Intra-board	
	bi	uni	bi	uni
P1, w/o filters	0.5	1.1	1.2	3.5
P1, w. filters	1.7	3.0	2.5	6.0
P2, w/o filters	0.5	1.4	2.5	10.0
P2, w. filters	4.0	5.0	12.5	> 25.0

bi = bidirectional link; uni = unidirectional link;
P1 = original, Dally and Poulton BGA model;
P2 = $L/10, C/3$ BGA model.

Table 4.1: Maximum Bit Rates (in Gb/s/pair/direction) for 50% Eye Height

While we believe that these models provide a fairly realistic model for high-speed links, it is important to note that we did not include ground and Vdd bounce, timing jitter, or PC board vias. Thus, with a real, physical link, the actual eye heights would be lower than the ones we report, and the effects of vias will be especially pronounced at high data rates. However, by using the same models for the channels with and without filters, we believe that our results realistically indicate the merits of our equalization filters and our unified optimization framework.

4.3.2 Simulation Results

Table 4.1 shows the bit rates that can be achieved for a variety of link configurations with four differential pairs. All filters were optimized using the parameterized eye mask from Figure 3.3. For all of the filters, the tap rate is twice the symbol rate. The pre-equalization filters compute their outputs for each differential pair based on the data input for that pair and for each of its immediate neighbors. We set the length of the filters to values that appeared to give good trade-offs between filter complexity and signal integrity. For inter-board links, the pre-equalizer has six taps (i.e. three symbol times), and for intra-board links it has four taps.

The decision feedback equalizer only considers the current line. Like the pre-equalizer, the DFE filter has six taps for the inter-board links and four taps for the intra-board links. We assume a latency of two symbol periods in the DFE; thus, it can only correct for ISI and reflections that occur after that delay. The DFE, together with the pre-equalizer, covers most of the bit response of the link to cancel ISI caused by high-frequency losses and reflections. We experimented with adding an extra set of taps to the DFE corresponding to the delay for reflections in the backplane, but the improvement that we saw in signal integrity was negligible. For our simulations, the DFE produces an output for every tap period – this allows

eye-height	Inter-board		Intra-board	
	bi	uni	bi	uni
50% (2-PAM)	4.0	4.0	12.5	> 25.0
83% (4-PAM)	< 1.0	3.3	2.5	15.0
93% (8-PAM)	< 1.0	< 1.0	< 1.0	8.0

Table 4.2: Maximum Bit Rates (in Gb/s/pair/direction) for Various Eye Heights

us to construct the eye-diagram for the full sample period. We note that in an actual implementation, the DFE only needs to produce an output at the sample time for each symbol.

Reflections are particularly severe for the bidirectional links, and we designed the near-end (NE) filters with multiple segments corresponding to the delays of the principal reflections. For the inter-board links, the NE filters have four segments. The first two segments have six taps and consider nearest neighbors. The last two have six taps but only consider the line itself. The NE filters for the intra-board links have two segments. Both segments have four taps and consider nearest neighbors. For the inter-board NE filters the last two segments can ignore neighboring lines because there is sufficient high-frequency attenuation in the backplane to render such coupling negligible.

In our initial designs, near-end crosstalk severely limited the performance of the bidirectional links. This is because the peak of the near-end interference can occur anywhere in the received eye. To mitigate this, we introduced an integrating receiver [59, 75] at the input of the thresholding element. We modelled it with a simple convolution over four consecutive sample points, using weights of $1/8$, $3/8$, $3/8$, and $1/8$. This allowed bandwidth improvements of roughly 10% for most of the bidirectional scenarios and a much larger improvement of 60% for the intra-board link with the reduced inductance package. In all cases, the integrator reduced the performance of the unidirectional links. Thus, we report results for bidirectional links with an integrating receiver and with a simple, thresholding receiver for the unidirectional links.

Rather than picking fixed weights for the integrator, it would be desirable to have them included in the optimization problem. We note that this is no longer a linear optimization problem because the received signal depends on the product of the pre-equalizer coefficients and the integrator weights. Optimizing the integrator weights is a topic for future research.

From Table 4.1, we see that the equalizing filters double the channel bandwidth in all cases with even greater gains when using the low inductance package. Due to

near-end crosstalk, bidirectional signalling always has a lower one-way bandwidth than unidirectional signalling. However, bidirectional signalling simultaneously sends and receives data across the same transmission channel. It effectively doubles the maximum throughput for a given pin-count when the workloads on both directions are equal. Therefore, to obtain the peak effective throughput, we need to double the numbers given in Table 4.1. Alternative to simultaneous bidirectional signalling, we can use a unidirectional link for each direction. For example, PCI-express uses two unidirectional serial links to realize bidirectional communication [50]. This provides lower latency and greater total bandwidth at the cost of greater pin and wire counts. The better BGA package provides greater bandwidth than the old BGA package, especially in the case of intra-board links. For inter-board links, with better BGA packages, the connector between the line card and the backplane becomes the bottleneck that limits the bandwidth.

Table 4.2 explores the trade-off between symbol rate and eye height. All results are with the reduced inductance package. If a channel with two-level signalling achieves an eye height of $(100 - E/(N-1))\%$, then a channel with N-level signalling and the same filters can achieve an eye height of $(100 - E)\%$. This is because we assume eye masks that constrain overshoot and undershoot symmetrically. Thus, 83% eye height for two-level signalling provides 50% for four-level, and a 93% eye height for two-level signalling provides 50% for eight-level. Because we are ignoring ground bounce, and clock jitter, these are optimistic estimates, especially for multi-level signalling. In all cases, we constrained the maximum output of the filter to be at most three times the target value. Greater signal integrity can be achieved in the unidirectional case with greater overdrive, but we regarded that the cost in power and the need for greater DAC resolution preclude such designs.

Table 4.2 illustrates how a designer can use our synthesis procedure to explore design trade-offs. For example, it shows that for the channels that we considered, multi-level signalling is never advantageous for a bidirectional link – the total bandwidth of the link is much less than can be achieved with simple, two-level signalling. For unidirectional links, four-level signalling is a more viable alternative. A designer would have to trade-off the advantages of a lower symbol rate against the increased complexity in the receiver circuitry. Moreover, in all cases, eight-level signalling is never advantageous compared with four-level signalling. We note that because our models do not include PC board vias, the extremely high data rates obtained for intra-board, unidirectional links should be taken with many grains of salt. Even in this case, we see the advantage of automatic filter synthesis – it allows us to quickly identify scenarios where more detailed modelling of the channel is needed.

We also tried separate synthesis of the filters. We first synthesized an opti-

mal pre-equalizer assuming no DFE or NE filters, and then synthesized the other two filters including the pre-equalizer in the channel. The unified approach performed much better in the bidirectional case when the filter output magnitude was unconstrained. This is because the pre-equalizer generated by separate synthesis would drive very high slew-rate transitions into the channel causing severe near-end crosstalk. Reducing the maximum filter output magnitude mitigates this effect. However, it also reduces the performance of the pre-equalizer. Hence manual adjustments and multiple iterations are needed to achieve good performance with separate synthesis. The unified optimization automatically finds the best balance between the filters and achieves eye-heights 5-10% greater than those for separate optimization. Thus, in addition to producing better filters, the designer can explore a simpler design space without losing optimality.

Our filter design times range from less than one minute for a unidirectional link with four differential pairs, to about two minutes for a bidirectional link with four pairs and about forty minutes for a bidirectional link with 16 pairs. These times are for a 900MHz, UltraSparc III processor. The time is roughly equally divided between setting up the linear program and solving it. As described in Chapter 5, the LP solver has been optimized for filter synthesis. The LP set-up code has not been carefully tuned and there are likely opportunities for further efficiency gains.

4.4 Generalizations

In Section 3.2 and Section 4.1, we made several assumptions to simplify the presentation. We now revisit each of these assumptions.

The response of the channel is linear. Linearity is what allows us to consider the responses from each wire and each bit-time separately. Fortunately, linear models are very accurate for off-chip interconnect. We do not see this as a major limitation.

Each differential pair is used to convey two-level signals. As noted in Section 4.3, our methods extend directly to multi-level signalling because we consider both overshoot and undershoot.

The eye mask is symmetric and the same for every differential pair. Our methods work with asymmetric eye masks and with different eye masks for each link of the channel. These changes would be reflected in the objective function. For example, with different eye masks for each link of the channel, let η_j be the eye-mask parameter for link j. A simple way to deal with different eye masks for each link of the channel is to introduce a global variable η, such that, for every link j, $\eta_j \leq \eta$. The final LP would remain the same with one additional constraint for each link of the channel.

The channel is symmetric. As noted following equation 4.4, this assumption can easily be removed. Disturbances from the two transceivers would have to be considered separately. This roughly doubles the number of variables and constraints in the LP, but the structure remains the same.

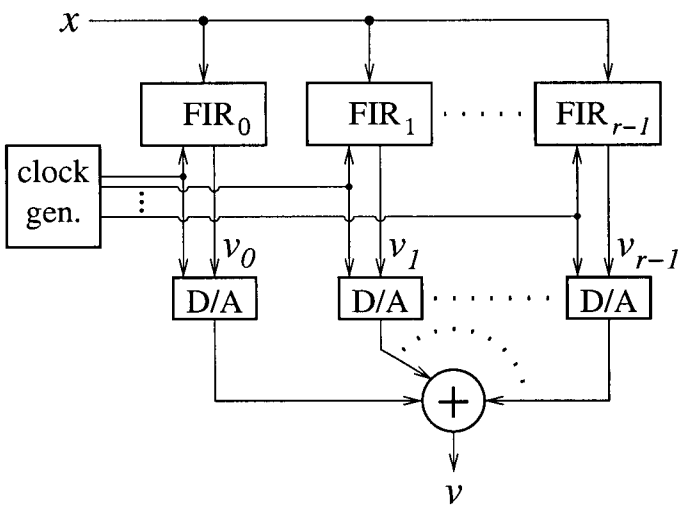
The link is error-free. In practice, errors may occur, and following an error, the DFE filter may corrupt signal integrity rather than improve it. In practice, the DFE filter output is typically small enough to prevent cascading errors from becoming a serious issue. Consider a bus with an error-rate of 10^{-15}/symbol when the data input to the DFE is correct. For the sake of an example, we will assume that the error rate increases to 10^{-6} when the DFE is processing an erroneous bit. Because the higher error rate only lasts for a few symbols, the impact on the overall error rate is negligible. More generally, our linear programming formulation provides a natural opportunity for restricting the magnitude of the DFE output to ensure that cascading errors are not a problem.

The filter synthesis problem is a linear program. As discussed in Section 3.1, our linear programming formulation corresponds directly to eye masks. While we parameterized the eye mask using a single scaling factor, η, we note that the linear programming framework provides a great degree of flexibility in formulating the objective function. Nevertheless, there are some natural aspects of filter design that do not fit into our linear programming framework. In particular, optimizing cascaded filters (such as the integrators discussed in Section 4.3) and minimizing the power output of the transmitter require quadratic formulations. We examine these issues further in Section 7.1.

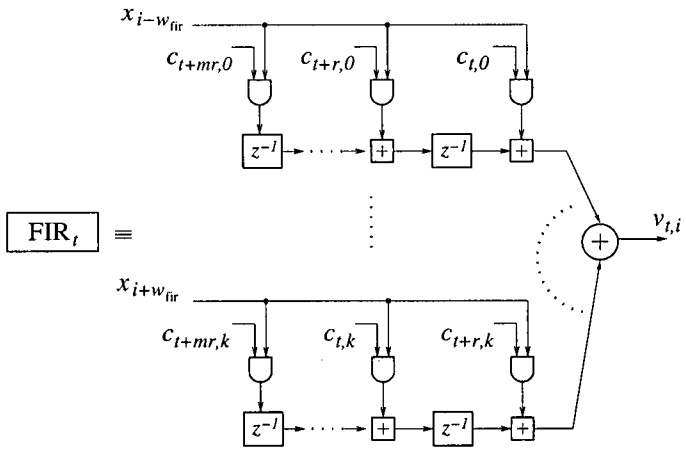
4.5 Hardware Implementation

We briefly consider the practicality of implementing the equalization filters designed in Section 4.3. Once the size and the oversample rate of the filter are decided, the filter implementation can use standard FIR designs [17]. The filter synthesis problem amounts to determining the values of the filter coefficients. We do this using the optimization methods described in the previous sections. These coefficients can then be loaded into the FIR hardware, for example by using a scan-chain.

As an example, we consider an implementation based on pipelined adder trees to compute the output for each wire. As shown in Figure 3.4, the filter can be implemented as a separate filter for each wire that receives inputs for the data to be transmitted on the wire itself and a few of its nearest neighbors. Figure 4.4 shows a hardware implementation framework suitable for the filters described in this thesis. It uses an interleaved DAC as described in [17, 71]. The clock generator



(a) FIR filter with interleaved DAC



(b) Filter for each DAC

Figure 4.4: An implementation of an equalizing filter

produces phases for enabling each DAC. A current summing circuit combines the DAC outputs to produce the filter output, v. For simplicity, we show a design where the interleaving factor for the DAC is the same as the oversampling rate of the filter, r. By using a separate filter for each DAC, the DACs are incorporated into the channel, and filter coefficients can be adjusted to compensate for variations between the DACs.

Because the filter is linear, we can compute the contributions to the output arising from each input separately. The output of a FIR filter for a single DAC channel includes a delay, z^{-t} to align its output with the clock phase of its DAC. Figure 4.4(b) shows an implementation of a FIR filter for a single DAC channel. The filter coefficients, $c_{t,j}$ correspond to the contributions of an input on wire j after a delay of t tap times:

$$c_{t,j} = \sum_{g=0}^{\min(t,r)} F(t-g,j) \tag{4.7}$$

The values of x are either 0 or 1, thus the multiplications are simple AND gates.

We now estimate the area cost and latency cost of our filters. For a filter of size $m_{\text{fir}} \times w_{\text{fir}}$, the number of filter coefficients per wire is $m_{\text{fir}}(2w_{\text{fir}}+1)$. The filters for the wires on the edges of the bus will have smaller number of filter coefficients. We will ignore the edge effects for the estimation below. If we consider an oversampling rate of r, the filter for each DAC needs to sum up $n_{\text{fir}}(2w_{\text{fir}}+1)$ filter coefficients to compute the input for the DAC, where $n_{\text{fir}} = m_{\text{fir}}/r$ is the length of the filter in bit time. This requires $r(n_{\text{fir}}(2w_{\text{fir}}+1)-1)$ full adders for each wire. For example, if we consider an oversampling rate of 2 and a filter of size 6×1, and 8-bit data paths for 8-bit DACs, this design requires 128, 1-bit adders. Each 1-bit adder can be implemented with less than 40 transistors. Thus, a 6×1 filter can be constructed with $\sim$ 5K transistors per output pad. The total area cost per output pad is less than 20K transistors for the bi-directional inter-board links presented in Section 4.3 which employ pre-equalization, decision-feedback equalization and near-end crosstalk cancellation (see Table 4.5). Notice that this is a straightforward implementation without any effort to reduce the number of transistors. With careful design, we believe the area could be reduced further. For a chip with 100-200 million transistors and a few hundred high-speed I/Os, our filters can double the output bandwidth for a few percent of the total chip area.

Furthermore, the filter's latency is very small. The design shown here requires an adder-tree of depth roughly $\log_{3/2}(4w_{\text{fir}}+2)$. With simple retiming techniques, the latency cost can be reduced to roughly $\log_{3/2}(2w_{\text{fir}}+1)$. For example, for a 6×1 filter, that is 2 to 3 adder delays. Thus, it is reasonable to estimate that the

Filter	Filter Size	# Filter Coefficients	# 8-bit Adders
PE	6×1	18	16
DFE	6×0	6	4
NE, segment 1,2	6×1	18	16
NE, segment 3,4	6×0	6	4

Table 4.3: Hardware cost of the equalization filters for the bi-directional inter-board links presented in Section 4.3.

filter adds less than 500ps to the latency of the channel for an implementation in a 0.13μ process.

4.6 Summary

Chip clock rates continue to grow at a much faster rate than improvements in off-chip interconnect. To bridge the gap, designers are using increasingly sophisticated on-chip equalization filters. To achieve maximum bandwidth, designers need to jointly synthesize these filters. In this chapter, we presented a unified approach for synthesizing optimal filters for three of the most common forms of equalization: transmitter pre-equalization, decision-feedback equalization, and near-end crosstalk cancellation.

To illustrate the use of our methods, we examined the design of bidirectional and unidirectional channels for both intra-board and cross-backplane communication. Our models included chip packaging parasitics, impedance mismatches, dielectric and skin-effect losses, and connector parasitics. The optimal filters significantly improve channel bandwidth in all cases. Moreover, automatic synthesis of optimal filters allows early identification of bandwidth bottlenecks and rapid evaluation of design trade-offs such as the use of multi-level signalling and integrating receivers.

We briefly examined the area and latency costs of the filters synthesized in this chapter. The significant bandwidth advantages, the acceptable per-pad transistor count and the low added latency demonstrate the practicality of these filters.

Chapter 5

Implementing the Optimization Algorithm

Chapter 3 and 4 formulated the l_∞ optimal equalization filter synthesis problem as a linear programming (LP) problem. In this chapter, we describe an efficient algorithm for solving the resulting LPs. The size of the LPs presents a computational challenge. The number of variables and constraints grows quadratically with the number of links in the bus. The examples presented in Section 4.3 have over 10,000 variables and over 15,000 constraints for a bidirectional bus with four differential pairs. With sixteen pairs, the LP has over 100,000 variables and 200,000 constraints. To solve these large LPs efficiently, we implemented a customized version of Mehrotra's interior-point, predictor-corrector algorithm [46, Section 14.2] using MATLAB [65].

To simplify the presentation, we present the solution for channels with preequalizers. For more general cases as presented in Chapter 4, the algorithm described in this chapter applies in a straightforward fashion. Section 5.1 describes the detailed structure of the LP and shows that the large number of LP variables is primarily due to the number of disturbance terms. Section 5.2 briefly describes Mehrotra's interior-point method. The critical step for the performance of Mehrotra's interior-point method is the solution of a linear system derived from the constraint matrix. Section 5.3 presents our efficient linear system solver that exploits the sparsity structure of our particular constraint matrix and analyzes the time complexity and space requirements of the algorithm. Section 5.4 analyzes the numerical stability of the linear system solver. Section 5.5 examines the time complexity and space requirements of the least-squares (LSQ) approach presented in Section 3.4.3.

5.1 The Linear Program

Given a bus's bit responses and an eye-mask specification, we set up the linear programming problem according to the formulation presented in the previous chapters. In addition, we note that in practice, filters have limited overdrive ability. Thus, we

augment the constraints from equation 3.27 with constraints to limit the magnitude of the filter output on each wire at each tap time. The final linear programs have the following constraints:

- **Constraints to Compute the Absolute Values of the Disturbances:**

$$|g(i,j,s)^T f| = d(i,j,s) \tag{5.1}$$

As defined in Section 3.4.1, $g(i,j,s)$ is the vector such that $g^T(i,j,s)f$ is the response on wire j at time s to a single bit input on wire i given filter f. The g vectors can be readily derived from the bus bit responses (see equation 3.23). To obtain a linear formulation, we turn the equality constraints on absolute values above into the following two linear inequality constraints:

$$\begin{array}{rcl} g(i,j,s)^T f & \leq & d(i,j,s) \\ -g(i,j,s)^T f & \leq & d(i,j,s) \end{array} \tag{5.2}$$

Note that we are effectively minimizing the maximum total disturbance (see Section 3.4.2), which is the sum of the absolute value of each individual disturbances, i.e. the l_1 norm of the disturbance vector d. At optimality, one of the two constraints from equation 5.2 for each component of d must be tight; otherwise, we would be able to further improve the objective function. Hence, with these two constraints, at the optimal vertex, the components of d are the absolute values of the disturbances.

As defined in Section 3.4.2, G_d denotes the set of $g(i,j,s)$ vectors such that the rows of G_d compute the disturbances. In matrix form, equation 5.2 becomes

$$\begin{array}{rcl} G_d f & \leq & d \\ -G_d f & \leq & d \end{array} \tag{5.3}$$

Let k_{disturb} denote the number of disturbances (i.e. the length of d) and k_{fir} denote the number of filter coefficients (i.e. the length of f); thus, G_d is $k_{\text{disturb}} \times k_{\text{fir}}$. The number of filter coefficients, k_{fir}, is roughly $w_{\text{bus}} m_{\text{fir}}(2w_{\text{fir}}+1)$. Note that, k_{disturb} grows quadratically with the width of the bus. We have one disturbance term for each pair of wires, each sample time, and each symbol time of the channel's bit response. Let k_{mask} denote the number of sample points for which the eye mask specifies constraints. The number of bit-times that disturb a bus output at any particular bit time is roughly $(n_{\text{fir}} + n_{\text{bus}})$; thus

$$k_{\text{disturb}} \approx w_{\text{bus}}^2 k_{\text{mask}}(n_{\text{fir}} + n_{\text{bus}}) \tag{5.4}$$

The number of disturbances dominates the number of LP variables. A typical example with 16 differential pairs has over $50,000$ variables and over

100,000 constraints. The number of filter coefficients is typically only several hundred.

- **Eye-Mask Constraints:**

$$\begin{array}{rcl} G_u f + W d & \leq & 1 + \alpha\eta \\ G_u f - W d & \geq & 1 - \alpha\eta \end{array} \tag{5.5}$$

The eye-mask constraints above are the matrix representation of the constraints given in the LP formulation of equation 3.27. The first and second constraints determine the maximum overshoot and undershoot respectively. The column vector α has length $k_{\text{mask}} w_{\text{bus}}$, with an element for each measurement point for each wire. In the case of nonsymmetric eye masks, the α vector for the overshoot constraints is different from the one for the undershoot constraints. Likewise, constraints can be specified with a different eye mask for each wire without changing the structure of the matrices. Here, we assume symmetric eye masks to simplify the presentation. Note that this has no impact on the complexity of the LP. G_u has a separate row to compute the undisturbed response for each measurement point of the eye mask and for each wire; thus, G_u is $k_{\text{mask}} w_{\text{bus}} \times k_{\text{fir}}$. Each row of W computes the maximum total disturbance for that wire and measurement point by summing up the contributing disturbances. W is $k_{\text{mask}} w_{\text{bus}} \times k_{\text{disturb}}$. We write k_{block} to denote the number of disturbances contributing to a single measurement point of the eye mask for a given wire. Thus, $k_{\text{block}} = k_{\text{disturb}}/(w_{\text{bus}} k_{\text{mask}}) \approx w_{\text{bus}}(n_{\text{fir}} + n_{\text{bus}})$. We group together the rows of G_d for computing disturbances on a given wire at a given measurement point. With this permutation of G_d, W is block diagonal:

$$W(i,j) = \begin{array}{ll} 1, & \text{if } (i-1)k_{\text{block}} < j \leq i k_{\text{block}} \\ 0, & \text{otherwise} \end{array} \tag{5.6}$$

In Section 5.3, we show that this block structure of W leads to a very efficient way to solve the linear program.

- **Maximum Filter Output Constraints:**

$$\begin{array}{rcl} Ff & \leq & p \\ -Ff & \leq & p \\ Mp & \leq & \rho \end{array} \tag{5.7}$$

We can compute the maximum filter outputs for every tap time and every wire the same way as we compute the maximum total disturbance: Ff computes the bit responses of the filter f, and at optimality, the elements of

the vector p are the absolute values of the filter bit responses. The length of p, k_{maxf}, is roughly $w_{\text{bus}}(2w_{\text{fir}}+1)(m_{\text{fir}}+r_{\text{bit}}/r_{\text{tap}})$. Similar to W, M computes the maximum filter outputs for each wire at each tap time by summing up the contributions from each bit time and each wire. Thus, M is $w_{\text{bus}}r_{\text{bit}}/r_{\text{tap}} \times k_{\text{maxf}}$ and has the same structure as W, but with block size $1 \times n_{\text{fir}}(2w_{\text{fir}}+1)$. Note that F and M are much smaller than G_d and W. The parameter ρ is the limit of filter overdrive ability.

We combine the constraints above and write the linear program as:

$$\min_{f,d,\eta,p} \eta \text{ s.t.}$$

$$\begin{bmatrix} -I & 0 & G_d & 0 \\ 0 & -I & F & 0 \\ -I & 0 & -G_d & 0 \\ 0 & -I & -F & 0 \\ W & 0 & G_u & -\alpha \\ W & 0 & -G_u & -\alpha \\ 0 & M & 0 & 0 \end{bmatrix} * \begin{bmatrix} d \\ p \\ f \\ \eta \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ \rho \end{bmatrix} \tag{5.8}$$

The number of variables in the LP is

$$k_{\text{vars}} = k_{\text{fir}} + k_{\text{disturb}} + k_{\text{maxf}} + 1 \tag{5.9}$$

and the number of constraints is

$$k_{\text{cons}} = 2(k_{\text{maxf}} + k_{\text{disturb}} + w_{\text{bus}}k_{\text{mask}}) + w_{\text{bus}}r_{\text{bit}}/r_{\text{tap}} \tag{5.10}$$

Table 5.1 on page 69 summarizes the notation used in this chapter and also the matrix dimensions. To illustrate the relative sizes of the matrices, the third column of the table shows the values for each variable and matrix dimensions for a small example.

Typically, $n_{\text{bus}} > n_{\text{fir}}$ and $w_{\text{bus}} > w_{\text{fir}}$ in which case k_{disturb} is much greater than k_{fir} and k_{maxf}. This yields that the number of variables and the number of constraints are both $O(w_{\text{bus}}^2 n_{\text{bus}} k_{\text{mask}})$, and the total number of elements in the constraint matrix is $O(w_{\text{bus}}^4 n_{\text{bus}}^2 k_{\text{mask}}^2)$. Fortunately, the matrix is sparse: most of these elements are zero. Most of the nonzero elements of the matrix are from matrix G_d (see equation 5.8) whose rows are the g^T vectors as defined in Section 3.4.2. Note that $g(i,j,t)^T f$ denotes the bus output on wire j at time t for a single input bit on wire i at time 0. Thus, $g(i,j,t)$ only has nonzero elements at indices that correspond to the filter coefficients of wire i. Therefore, G_d has the following properties:

1. each column of G_d only has $1/w_{\text{bus}}$ of its elements nonzero.

2. the columns of G_d that correspond to filter coefficients of wire i have no overlapping nonzero elements with the columns of G_d that correspond to filter coefficients of wire k with $k \neq i$.

These sparsity properties also apply to G_u. We show in Section 5.3 that this sparsity structure allows us to greatly speed up the computation.

The matrix G_d has approximately a density of $1/w_{\text{bus}}$ and contains most of the nonzero elements of the constraint matrix. Thus, the overall density of the constraint matrix is roughly $k_{\text{fir}}/(k_{\text{disturb}} w_{\text{bus}})$. As an example, consider a twelve-tap pre-equalization filter ($m_{\text{fir}} = 12$) for a 32 bit bus ($w_{\text{bus}} = 32$), where the filter for each wire considers eight neighboring wires in each direction ($w_{\text{fir}} = 8$); the bus impulse response has a length of 4 bit times ($q_{\text{bus}} = 4r_{\text{bit}}$); there are 4 filter tap times per data bit time ($r_{\text{bit}}/r_{\text{tap}} = 4$); and the eye mask has four sample points $k_{\text{mask}} = 4$. For this example, k_{disturb}, k_{fir}, k_{maxf} are roughly 28K, 7K and 8K respectively. The linear program for such a filter has roughly 43K variables and 72K constraints. Thus, the constraint matrix has nearly 3 billion entries, of which roughly 24 million are nonzero.

5.2 Mehrotra's Interior-Point Algorithm

Primal-dual interior-point methods outperform the simplex method on many large problems and perform better than other interior-point methods. Among many general algorithmic approaches, the most effective one in practice has proven to be the primal-dual infeasible-interior-point approach, including a number of variants and enhancements such as Mehrotra's predictor-corrector technique [44]. To solve the large LPs formulated in previous chapters, we implemented Mehrotra's interior-point algorithm. This popular predictor-corrector algorithm is applied to a primal-dual formulation of the LP, and one of its key features is a clever, adaptive choice of the centering parameter [46, Section 14.2]. This section briefly describes Mehrotra's algorithm.

Consider an LP problem in standard form:

$$\min \{c^T x | Ax = b, x \geq 0\} \tag{5.11}$$

where $A \in R^{m \times n}$, which determines the sizes of other vectors involved. The dual problem for equation 5.11 is

$$\max \{b^T \lambda | A^T \lambda + z = c, z \geq 0\} \tag{5.12}$$

Primal-dual solutions of equation 5.11 and 5.12 are characterized by Karush-Kuhn-Tucker conditions [46, Section 13.1]:

$$\Psi(x, \lambda, a) = \begin{bmatrix} A^T\lambda + z - c \\ Ax - b \\ XZe \end{bmatrix} = 0 \tag{5.13}$$

where

$$X = \text{diag}(x_1, x_2, \ldots, x_n) \quad ,$$
$$Z = \text{diag}(z_1, z_2, \ldots, z_n) \quad ;$$
$$e = (1, 1, \ldots, 1)^T$$

The system of equations 5.13 can be solved by applying Newton's method and carrying out a line search to enforce the non-negativity constraints on x and z. Unfortunately, often we can only take a small step before violating the non-negativity constraints. Therefore, the pure Newton's method with line search converges very slowly in this case. Rather than solving the system of equations 5.13, primal-dual interior point methods introduce the concept of a central path. The central path is parameterized by a scalar τ, and consists of a set of points that are solutions of the following linear system for $\tau > 0$:

$$\Psi(x_\tau, \lambda_\tau, z_\tau) = \begin{bmatrix} A^T\lambda + z - c \\ Ax - b \\ XZe \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix} \tag{5.14}$$
$$(x, z) > 0$$

The role of τ is to enforce that all the complementarity products have the same values for all indices. Hence, the central path keeps iterates biased towards the interior of the nonnegative orthant $(x, z) \geq 0$. As τ approaches 0, the solution of the linear system 5.14 approaches the optimal solution (x^*, λ^*, z^*) which is the solution of the linear system 5.13. In practice, τ is defined as the product of a centering parameter σ and a complementarity gap μ, where $\sigma \in [0, 1]$ and $\mu = x^T z/n$.

Mehrotra's predictor-corrector algorithm [44, 46] implements the basic ideas described above with an additional second-order correction. It consists of three major steps.

Given an initial point (x^0, λ^0, z^0) with $(x^0, z^0) > 0$,
For $k = 0, 1, 2, \ldots$

Predictor step: At this step, Mehrotra's algorithm computes the pure Newton (affine-scaling) direction $(\Delta x^{aff}, \Delta\lambda^{aff}, \Delta s^{aff})$ by solving:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta\lambda^{aff} \\ \Delta z^{aff} \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -X^k Z^k e \end{bmatrix} \tag{5.15}$$

where $r_c = A^T\lambda^k + z^k - c$, $r_b = Ax^k - b$ are the residuals of the primal and dual feasibilities respectively.

An adaptive approach for computing the centering parameter σ: The algorithm calculates this parameter based on the complementarity gap at the current point and the gap after taking a hypothetical step in the affine scaling direction. The step size in the affine scaling direction is calculated by

$$\begin{aligned} \alpha^p_{aff} &= \min(1, \min_{i:\Delta x_i^{\text{aff}}<0} -x_i^k/\Delta x_i^{aff}) \\ \alpha^d_{aff} &= \min(1, \min_{i:\Delta s_i^{aff}<0} -z_i^k/\Delta z_i^{aff}) \\ \mu_{aff} &= (x + \alpha^p_{aff}\Delta x^{aff})^T(z + \alpha^d_{aff}\Delta z^{aff})/n \end{aligned} \tag{5.16}$$

The algorithm sets the centering parameter to $\sigma = (\mu_{aff}/\mu)^3$. Thus, the centering parameter is small when good progress can be made in the affine direction and large when the affine direction produces little improvement and more centrality is needed. In this way, the choice of σ balances the competing goals reducing μ and improving centrality.

A corrector step: At this step, Mehrotra's algorithm solves the following equations to obtain a corrected, centered step direction. It is essentially a step based on the Taylor series expansion of the complementarity equations [56]. The resulting linear system of equations is

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta\lambda^k \\ \Delta z^k \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -X^k Z^k e - \Delta X^{aff}\Delta Z^{aff} e + \sigma\mu e \end{bmatrix} \tag{5.17}$$

Finally, the algorithm sets the step size, α, by the same basic approach as presented in equation 5.16 and updates $(x^{k+1}, \lambda^{k+1}, z^{k+1}) = (x^k, \lambda^k, z^k) + \alpha(\Delta x^k, \Delta\lambda^k, \Delta z^k)$.

The special structure of the left-hand-side matrix in equations 5.15 and 5.17 enables their reduction into smaller systems with positive definite matrices [46]. For example, 5.15 can be reformulated into the following system:

$$\begin{aligned} A\Lambda^2 A^T \Delta\lambda &= -r_b + A(-Z^{-1}Xr_c + x - \sigma\mu Z^{-1}e) \\ \Delta z &= -r_c - A^T\Delta\lambda \\ \Delta x &= \text{⌙}x + \sigma\mu Z^{-1}e - Z^{-1}X\Delta z \end{aligned} \tag{5.18}$$

with $\Lambda = Z^{-1/2}X^{1/2}$ and $\Delta z = -z - X^{-1}Z\Delta x$. The first equation above is called the system of "normal equations" for the LP. Recall that X and Z are diagonal matrices; thus, the right hand side of equation 5.17 can be computed very efficiently. Forming and inverting $A\Lambda^2 A^T$ directly requires impractical amounts of time and memory for real filter design problems. The next section presents our approach to solving this system efficiently.

5.3 An Efficient Linear System Solver

The LP given in equation 5.8 is in dual form. Let A denote the constraint matrix in equation 5.8, and let x be the column vector $[d, p, f, \eta]^T$. Each Newton step of Mehrotra'a algorithm solves the linear system corresponding to the *normal equations*:

$$A^T\Lambda^2 Ax = y \tag{5.19}$$

where Λ is a diagonal matrix whose elements are updated with each iteration of the algorithm (see Section 5.2).

The size of the normal equations grows quadratically with the width of the bus (see Section 5.1). Even with standard sparse-matrix techniques, the resources required to form the normal equations and solving the system would render the l_∞ filter synthesis approach impractical. Large linear systems can be solved either by using iterative methods or by direct techniques that exploit special properties of the specific problem. We explored both approaches. Iterative methods require preconditioners, especially for ill-conditioned systems such as those that arise as interior-point methods approach optimality. We have not found a suitable preconditioner for our problem. On the other hand, we did find a very efficient way to solve the problem directly. We present this direct approach here.

- **Block Computation:**
  Due to the size of A, computing the normal equation directly from A would use an unacceptable amount of memory. We note that A is formed by distinct

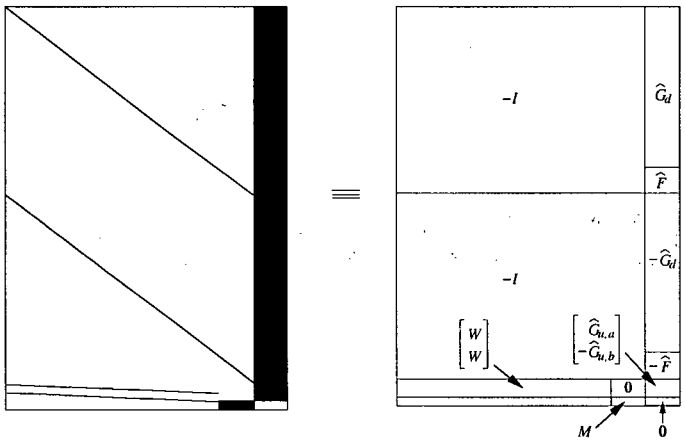


Figure 5.1: Sparsity pattern of the constraint matrix A.

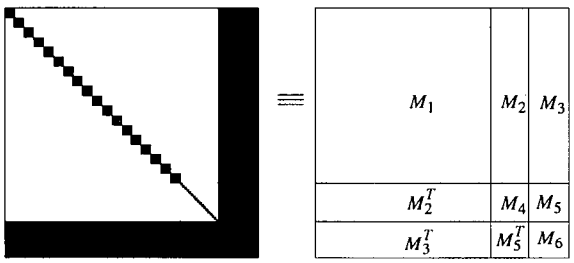


Figure 5.2: Sparsity pattern of $A^T \Lambda^2 A$.

r_{bit}	Number of sample points per bit time	8
r_{tap}	Number of sample points per tap time	4
w_{bus}	Bus width	16
q_{bus}	The length of the bus impulse response in sample time	80
w_{fir}	Filter width	1
m_{fir}	Filter length in tap time	6
k_{mask}	Number of measurement points defined in the eye mask	8
k_{fir}	Number of filter coefficients, roughly $w_{\text{bus}} m_{\text{fir}}(2w_{\text{fir}}+1)$	276
k_{disturb}	Number of variables for computing absolute values of the disturbances, roughly $w_{\text{bus}}^2 k_{\text{mask}}(n_{\text{fir}}+n_{\text{bus}})$	38784
k_{maxf}	Number of variables for computing absolute values of the filter bit responses, roughly $w_{\text{bus}}(m_{\text{fir}}+r_{\text{bit}}/r_{\text{tap}})(2w_{\text{fir}}+1)$	368
k_{block}	$k_{\text{disturb}}/(k_{\text{mask}} w_{\text{bus}})$	303
G_d	$k_{\text{disturb}} \times k_{\text{fir}}$, roughly $k_{\text{disturb}} k_{\text{fir}}/w_{\text{bus}}$ nonzeros	38784×276
G_u	$w_{\text{bus}} k_{\text{mask}} \times k_{\text{fir}}$, roughly $k_{\text{mask}} k_{\text{fir}}$ nonzeros	128×276
W	$w_{\text{bus}} k_{\text{mask}} \times k_{\text{disturb}}$, only k_{disturb} nonzeros	128×38784
F	$k_{\text{maxf}} \times k_{\text{fir}}$, roughly $k_{\text{maxf}} k_{\text{fir}}/w_{\text{bus}}$ nonzeros	368×276
M	$w_{\text{bus}} r_{\text{bit}}/r_{\text{tap}} \times k_{\text{maxf}}$, only k_{maxf} nonzeros	32×368
M_1	$k_{\text{disturb}} \times k_{\text{disturb}}$ (Block Diagonal)	38784×38784
M_2	$k_{\text{disturb}} \times k_{\text{maxf}}$ (All zero)	38784×368
M_3	$k_{\text{disturb}} \times (k_{\text{fir}}+1)$	38784×277
M_4	$k_{\text{maxf}} \times k_{\text{maxf}}$ (Block Diagonal)	368×368
M_5	$(k_{\text{maxf}}) \times (k_{\text{fir}}+1)$	368×277
M_6	$(k_{\text{fir}}+1) \times (k_{\text{fir}}+1)$	277×277

Table 5.1: Notation and matrix dimensions; the third column shows the values and matrix dimensions for a 32-wire differential bus.

blocks naturally arising from the linear program as shown in equation 5.8. Let us define the following matrices based on equation 5.8:

$$\hat{F} = [F\ 0]\ ;\ \hat{G}_d = [G_d\ 0]\ ;\ \hat{G}_{u,a} = [G_u\ -\alpha]\quad \hat{G}_{u,b} = [G_u\ +\alpha] \tag{5.20}$$

where 0 is a column vector and α is the vector specifying the eye mask that we defined previously in the formulation of the LP. The sizes of G_d, G_u, F, W and M are given in Table 5.1. We identify blocks M_1 through M_6 as shown in Figure 5.2 and defined below:

$$\begin{aligned}
M_1 &= \Lambda_1^2 + \Lambda_3^2 + W^T(\Lambda_5^2 + \Lambda_6^2)W \\
M_2 &= 0 \\
M_3 &= (\Lambda_3^2 - \Lambda_1^2)\hat{G}_d + W^T(\Lambda_5^2\hat{G}_{u,a} - \Lambda_6^2\hat{G}_{u,b}) \\
M_4 &= \Lambda_2^2 + \Lambda_4^2 + M^T\Lambda_7^2 M \\
M_5 &= (\Lambda_4^2 - \Lambda_2^2)\hat{F} \\
M_6 &= \hat{G}_d^T(\Lambda_1^2 + \Lambda_3^2)\hat{G}_d + \hat{F}^T(\Lambda_2^2 + \Lambda_4^2)\hat{F} + \hat{G}_{u,a}^T\Lambda_5^2\hat{G}_{u,a} + \hat{G}_{u,b}^T\Lambda_6^2\hat{G}_{u,b}
\end{aligned} \tag{5.21}$$

where Λ_1, Λ_2, Λ_3, Λ_4, Λ_5, Λ_6 and Λ_7 are the portions of Λ corresponding respectively to rows of $\hat{G}_d$, $\hat{F}$, $-\hat{G}_d$, $-\hat{F}$, $\hat{G}_{u,a}$, $\hat{G}_{u,b}$ and M in the constraint matrix A (see Figure 5.1).

- **Schur Complement Formulation:**
The sparsity pattern of $A^T\Lambda^2 A$ is depicted in Figure 5.2. Solving the normal equations directly is time consuming. For example, the bottom rows are updated at each iteration of a Cholesky decomposition [67, p. 174].The update fills in the bottom rows of $A^T\Lambda^2 A$, destroys the sparsity of those blocks and results in a slow, memory intensive computation. Other direct techniques suffer similar inefficiencies. The block structure of the matrix lends itself naturally to a block elimination procedure. In particular, since $\begin{pmatrix} M_1 & M_2 \\ M_2^T & M_4 \end{pmatrix}$ is block diagonal (because $M_2 = 0$), it makes sense to form the Schur complement [22, p. 101] of $A^T\Lambda^2 A$ with respect to this 2×2 block matrix. This yields

$$\begin{aligned}
Sx_3 &= y_3 - (M_3^T M_1^{-1} y_1 + M_5^T M_4^{-1} y_2) \\
x_1 &= M_1^{-1} y_1 - M_1^{-1} M_3 x_3 \\
x_2 &= M_4^{-1} y_2 - M_4^{-1} M_5 x_3
\end{aligned} \tag{5.22}$$

where

$$S = M_6 - (M_3^T M_1^{-1} M_3 + M_5^T M_4^{-1} M_5) \tag{5.23}$$

and $x = [x_1; x_2; x_3]$, $y = [y_1; y_2; y_3]$. The lengths of x_i and y_i are determined by the matrices M_1 to M_6.

The Schur complement S is $(k_{\text{fir}} + 1) \times (k_{\text{fir}} + 1)$ and is much smaller than the original system of normal equations. We note that calculating right hand

sides of equation 5.22 can be computed easily as they only involve a small number of matrix-vector multiplications and vector additions provided that we have the inverses of M_1 and M_4. Thus, the time complexity is determined by the effort required to invert these two matrices and form the Schur complement, S. M_4 is much smaller than M_1, and the two matrices have similar structure. Thus, we focus our attention on inverting M_1 and computing the matrix S. Once S is formed, it is straightforward to solve the corresponding system using Cholesky decomposition.

- **Efficient Inversion of M_1:**
  Computing the Schur complement, S, involves inverting M_1 and M_4. Both matrices are block diagonal with similar properties (see equation 5.21). While M_4 is of size $k_{\text{maxf}} \times k_{\text{maxf}}$ and thus small and easy to invert, M_1 is $k_{\text{disturb}} \times k_{\text{disturb}}$. As k_{disturb} grows quadratically with the width of the bus, a direct inversion is computationally costly.

  From equation 5.21, we note that M_1 is the sum of $W^T(\Lambda_5^2 + \Lambda_6^2)W$ and a diagonal matrix $\Lambda_1^2 + \Lambda_3^2$. Let

$$\begin{aligned} \Lambda_{13}^2 &= \Lambda_1^2 + \Lambda_3^2 \\ \Lambda_{56}^2 &= \Lambda_5^2 + \Lambda_6^2 \end{aligned} \tag{5.24}$$

  It follows that $M_1 = \Lambda_{13}^2 + W^T\Lambda_{56}^2 W$. From equation 5.6, W is a $k_{\text{mask}} w_{\text{bus}} \times k_{\text{disturb}}$ matrix composed of non-overlapping, stripes of ones as shown below:

$$W = \begin{bmatrix} 1\cdots1 & & & & \mathbf{0} \\ & 1\cdots1 & & & \\ & & 1\cdots1 & & \\ & & & \ddots & \\ \mathbf{0} & & & & 1\cdots1 \end{bmatrix} \tag{5.25}$$

  Each stripe is of length $k_{\text{block}} = k_{\text{disturb}}/k_{\text{mask}}w_{\text{bus}}$. From the structure of W, it is easy to see that $W^T\Lambda_{56}^2 W$ forms a block diagonal matrix, with $w_{\text{bus}}k_{\text{mask}}$ blocks of size k_{block}. Moreover, all the elements in the i^{th} sub-block have the same value, namely $\Lambda_{56}^2(i,i)$. Thus, each block of M_1 is a rank-1 update of a diagonal matrix. Because Λ_{56}^2 is a diagonal matrix, we can write $V = W^T\Lambda_{56}$ and obtain

$$M_1 = \Lambda_{13}^2 + VV^T$$

  This allows us to apply the Sherman-Morrison-Woodbury formula [25] directly to obtain a formula for M_1^{-1} that involves merely diagonal matrix inversion and low-rank updates.

$$\begin{aligned} M_1^{-1} &= \Lambda_{13}^{-2} - \Lambda_{13}^{-2}V(I + V^T\Lambda_{13}^{-2}V)^{-1}V^T\Lambda_{13}^{-2} \\ &= \Lambda_{13}^{-2} - \Lambda_{13}^{-2}VBV^T\Lambda_{13}^{-2} \end{aligned} \tag{5.26}$$

where $B = (I + V^T\Lambda_{13}^{-2}V)^{-1}$ is a $k_{\text{mask}}w_{\text{bus}} \times k_{\text{mask}}w_{\text{bus}}$ matrix. Let λ_{13} be the vector consisting of the diagonal elements of Λ_{13}^2. Note that $V^T\Lambda_{13}^{-2}V = \Lambda_{56}W\Lambda_{13}^{-2}W^T\Lambda_{56}$. From the structure of W (see equation 5.25), it is easy to see that $W\Lambda_{13}^{-2}W^T$ forms a diagonal $k_{\text{mask}}w_{\text{bus}} \times k_{\text{mask}}w_{\text{bus}}$ matrix whose i^{th} diagonal element equals $\sum_{j=1}^{k_{\text{block}}} \lambda_{13}(i * k_{\text{block}} + j)$. Accordingly, B is a diagonal matrix and

$$B(i,i) = \left(1 + \Lambda_{56}(i,i)\sum_{j=1}^{k_{\text{block}}} \lambda_{13}(i * k_{\text{block}} + j)\right)^{-1} \tag{5.27}$$

and thus, the computation of B thus involves $O(w_{\text{bus}}k_{\text{mask}}k_{\text{block}})$ operations which is $O(k_{\text{disturb}})$.

Rather than forming M_1^{-1} explicitly, we compute B and use formula 5.26 each place where M_1^{-1} is needed in the Schur complement computation as described below.

- **Forming the Schur Complement:**
We now look at the pieces of the Schur complement from equation 5.22 and show how each can be computed efficiently using the formulation for M_1^{-1} described above. The two pieces that dominate the time are the computation for M_6 and $M_3^TM_1^{-1}M_3$. We combine these two together to eliminate redundant computations. Using the formula for M_1^{-1} from equation 5.26 we get:

$$\begin{aligned} & M_6 - M_3^TM_1^{-1}M_3 \\ &= M_6 - \left[M_3^T\Lambda_{13}^{-2}M_3 - (V^T\Lambda_{13}^{-2}M_3)^TB(V^T\Lambda_{13}^{-2}M_3)\right] \end{aligned} \tag{5.28}$$

We calculate $M_6 - M_3^T\Lambda_{13}^{-2}M_3$ and $(V^T\Lambda_{13}^{-2}M_3)^TB(V^T\Lambda_{13}^{-2}M_3)$ separately. Next, we expand M_3 and M_6 according to their definitions in equation 5.21 to get

$$M_6 - M_3^T\Lambda_{13}^{-2}M_3 = S_1 - (S_2 + S_2^T) \tag{5.29}$$

where

$$\begin{aligned} S_1 &= \hat{G}_d^T(\Lambda_{13}^2 - (\Lambda_3^2 - \Lambda_1^2)^2\Lambda_{13}^{-2})\hat{G}_d \\ &\quad + \hat{F}^T(\Lambda_2^2 + \Lambda_4^2)\hat{F} \\ &\quad + \hat{G}_{u,a}^T\Lambda_5^2\hat{G}_{u,a} + \hat{G}_{u,b}^T\Lambda_6^2\hat{G}_{u,b} \\ &\quad + (\Lambda_5^2\hat{G}_{u,a}^T - \Lambda_6^2\hat{G}_{u,b}^T)W\Lambda_{13}^{-2}W^T(\Lambda_5^2\hat{G}_{u,a} - \Lambda_6^2\hat{G}_{u,b}) \end{aligned} \tag{5.30}$$

$$S_2 = (\Lambda_5^2\hat{G}_{u,a}^T - \Lambda_6^2\hat{G}_{u,b}^T)W\Lambda_{13}^{-2}(\Lambda_3^2 - \Lambda_1^2)\hat{G}_d$$

Note that due to the sparsity pattern of W (see equation 5.25), $W\Lambda_{13}^{-2}W^T$ is diagonal of size $k_{\text{mask}}w_{\text{bus}} \times k_{\text{mask}}w_{\text{bus}}$; it takes k_{block} calculations to compute each element. Accordingly, it takes $k_{\text{block}}k_{\text{mask}}w_{\text{bus}} = k_{\text{disturb}}$ operations to compute $W\Lambda_{13}^{-2}W^T$. $\hat{G}_d$ is much larger than $\hat{F}$, $\hat{G}_{u,a}$ or $\hat{G}_{u,b}$ (see Table 5.1 and equation 5.20). Thus, the most computationally intensive component of this step is to compute $\hat{G}_d^T(\Lambda_{13}^2 - (\Lambda_3^2 - \Lambda_1^2)^2\Lambda_{13}^{-2})\hat{G}_d$. The Λ's are diagonal, so calculating $\Lambda_{13}^2 - (\Lambda_3^2 - \Lambda_1^2)^2\Lambda_{13}^{-2}$ only takes $O(k_{\text{disturb}})$ operations. As we described in Section 5.1, each column of $\hat{G}_d$ only has $1/w_{\text{bus}}$ of its elements nonzero. Moreover, each column only has overlapping nonzeros with approximately $k_{\text{fir}}/w_{\text{bus}}$ columns. Hence, for each row of the final product of $\hat{G}_d^T(\Lambda_{13}^2 - (\Lambda_3^2 - \Lambda_1^2)\Lambda_{13}^{-2})\hat{G}_d$, there are only $k_{\text{fir}}/w_{\text{bus}}$ nonzeros and each nonzero entry takes $k_{\text{disturb}}/w_{\text{bus}}$ operations to compute. Thus, this step takes $O(k_{\text{fir}}^2 k_{\text{disturb}}/w_{\text{bus}}^2)$ operations.

Now we look at the computation of $(V^T\Lambda_{13}^{-2}M_3)^T B(V^T\Lambda_{13}^{-2}M_3)$. We substitute for M_3 using equation 5.21 and get

$$V^T\Lambda_{13}^{-2}M_3 = (V^T\Lambda_{13}^{-2}(\Lambda_3^2 - \Lambda_1^2))\hat{G}_d + (V^T\Lambda_{13}^{-2}W^T)(\Lambda_5^2\hat{G}_{u,a} - \Lambda_6^2\hat{G}_{u,b}) \tag{5.31}$$

Because $\hat{G}_d$ is much larger than either $\hat{G}_{u,a}$ or $\hat{G}_{u,b}$, the time for this step is dominated by the time to compute $(V^T\Lambda_{13}^{-2}(\Lambda_3^2 - \Lambda_1^2))\hat{G}_d$. Note that $V = W^T\Lambda_{56}$ and Λ_{56} is diagonal; thus, V^T has the same structure as W shown in equation 5.25. V has exactly k_{disturb} nonzero elements. Multiplication of V^T with a diagonal matrix only takes k_{disturb} operations. Likewise, multiplying V^T by any column of $\hat{G}_d$ takes at most k_{disturb} operations. $\hat{G}_d$ has $k_{\text{fir}} + 1$ columns, thus, the multiplication of $V^T\Lambda_{13}^{-2}(\Lambda_3^2 - \Lambda_1^2)$ with $\hat{G}_d$ takes $O(k_{\text{disturb}}k_{\text{fir}})$ operations. Note that, here for simplicity, we ignored the sparsity of $\hat{G}_d$. Hence, this is an upper bound for the number of operations for calculating $V^T\Lambda_{13}^{-2}M_3$.

Because $V^T\Lambda_{13}^{-2}M_3$ is $k_{\text{mask}}w_{\text{bus}} \times (k_{\text{fir}} + 1)$ and B is a diagonal matrix, $(V^T\Lambda_{13}^{-2}M_3)^T B(V^T\Lambda_{13}^{-2}M_3)$ can be calculated with $O(k_{\text{fir}}^2 k_{\text{mask}}w_{\text{bus}})$ operations. Combining these observations, we note that typically $k_{\text{disturb}} >$

$k_{\text{fir}}k_{\text{mask}}w_{\text{bus}}$ and conclude $(V^T\Lambda_{13}^{-2}M_3)^T B(V^T\Lambda_{13}^{-2}M_3)$ can be completed with $O(k_{\text{disturb}}k_{\text{fir}})$ operations.

The time critical steps for forming the Schur complement have been described above. Summarizing, we get:

Calculating B: this can be done with $O(k_{\text{disturb}})$ operations. The number of disturbances k_{disturb} is approximately $w_{\text{bus}}^2 k_{\text{mask}}(n_{\text{bus}} + n_{\text{fir}})$. In typical filter designs, $n_{\text{bus}} > n_{\text{fir}}$. Thus, in terms of the design parameters, this step can be done with $O(w_{\text{bus}}^2 k_{\text{mask}} n_{\text{bus}})$ operations.

Calculating $M_6 - M_3^T\Lambda_{13}^{-2}M_3$: this requires $O(k_{\text{fir}}^2 k_{\text{disturb}}/w_{\text{bus}}^2)$ operations. The total number of filter coefficients k_{fir} is approximately $w_{\text{bus}}m_{\text{fir}}(2w_{\text{fir}} + 1)$. Thus, this step can be done with $O(w_{\text{bus}}^2 k_{\text{mask}} n_{\text{bus}} m_{\text{fir}}^2 w_{\text{fir}}^2)$ operations.

Calculating $(V^T\Lambda_{13}^{-2}M_3)^T B(V^T\Lambda_{13}^{-2}M_3)$: this can be done with $O(k_{\text{disturb}}k_{\text{fir}})$ operations, which is $O(w_{\text{bus}}^3 k_{\text{mask}} n_{\text{bus}} m_{\text{fir}} w_{\text{fir}})$.

In practice, filter design parameters such as m_{fir} and w_{fir} are typically small because of area and latency costs as well as power consumption of the filters. Thus, for wide buses, the total time for constructing the Schur complement is $O(k_{\text{disturb}}k_{\text{fir}})$, which is $O(w_{\text{bus}}^3 k_{\text{mask}} n_{\text{bus}} m_{\text{fir}} w_{\text{fir}})$. The Schur complement system can be solved in $O(k_{\text{fir}}^3) = O(w_{\text{bus}}^3 m_{\text{fir}}^3 w_{\text{fir}}^3)$ operations using Cholesky decomposition. Depending on the design parameters, either the Shur or Cholesky step can be the critical step. For small filters, $m_{\text{fir}}^2 w_{\text{fir}}^2$ is small relative to $k_{\text{mask}}n_{\text{bus}}$ and constructing the Schur complement dominates. For large filters, Cholesky decomposition dominates. In total, the linear system solver runs in $O(k_{\text{disturb}}k_{\text{fir}} + k_{\text{fir}}^3)$ time, which is $O(w_{\text{bus}}^3 m_{\text{fir}} w_{\text{fir}}(k_{\text{mask}}n_{\text{bus}} + m_{\text{fir}}^2 w_{\text{fir}}^2))$. Although the running time is cubic in w_{bus}, we note that in practice the width of parallel buses is limited by timing issues as discussed in Section 2.1. Moreover, note that $O(k_{\text{fir}}^3)$ is the amount of work required to solve a general linear system with k_{fir} independent variables, i.e. a variable for each filter coefficient.

We ran a few examples for buses of various sizes. Table 5.2A shows the the size of the LPs, the number of iterations and time per iteration for buses of various sizes with a 4×4 pre-equalizer. When the width of the bus doubles, the size of the LP goes up by a factor of 4. The number of LP iterations stays roughly constant. As the width of the bus doubles, the time per iteration also goes up by approximately a factor of 4, better than the factor of 8 predicted by the asymptotic analysis above. We note that the examples given in Table 5.2 are small in terms of w_{bus} and may not fully reflect the asymptotic regime. Moreover, the run-times using MATLAB can only serve as a rough indication to performance, since the routines are not

uniformly optimized. To show this, we list the time breakdown for the critical steps of the linear system solver in Table 5.2B.

First, note that the asymptotic behavior of each critical step validates the asymptotic analysis. For example, the time for calculating $M_6 - M_3^T \Lambda_{13}^{-2} M_3$ grows quadratically with the width of the bus, as predicted by the analysis. The time for calculating $V^T \Lambda_{13}^{-2} M_3$ grows by approximately a factor of 6, slightly better than the factor of 8 predicted by the analysis. Note that the asymptotic analysis for this step is an upper bound since we ignored the sparsity of $\hat{G}_d$ when we estimated the number of operations for calculating $V^T \Lambda_{13}^{-2} M_3$.

Second, as we noted above, the routines are not uniformly optimized. For example, in the asymptotic analysis, the Cholesky factorization is one of the critical operations. However, in our MATLAB implementation, the Cholesky factorization is fairly fast and calculating $M_6 - M_3^T \Lambda_{13}^{-2} M_3$ and $(V^T \Lambda_{13}^{-2} M_3)^T B (V^T \Lambda_{13}^{-2} M_3)$ determines the actual running time. We note that the chol() function in MATLAB for full matrices is highly optimized while we implemented the other steps in MATLAB in a straightforward fashion. Moreover, there are other issues such as cache and translation look-aside buffer (TLB) misses, paging etc. that will affect the measured run-times especially for computations that involve large matrices. Clearly there are many opportunities for further optimization of our implementation. This is an area for future work.

We now consider the memory requirements of this algorithm. In the steps presented above, we carefully avoid producing large dense intermediate matrices. For example, M_3 is large and dense. In step 3, instead of forming M_3, we compute $V^T \Lambda_{13}^{-2} M_3$ directly from the submatrices of the constraint matrix A. The largest matrix with the greatest number of nonzeros in the algorithm is $\hat{G}_d$. The number of nonzeros in $\hat{G}_d$ is approximately $2k_{\text{fir}} k_{\text{disturb}} / w_{\text{bus}}$ (see Section 5.1). Thus, the amount of memory required by this algorithm is $O(k_{\text{fir}} k_{\text{disturb}} / w_{\text{bus}})$, which is $O(w_{\text{bus}}^2 k_{\text{mask}} n_{\text{bus}} m_{\text{fir}} w_{\text{fir}})$.

5.4 Numerical Stability

We briefly address the issue of the numerical stability of our approach. In general, applying the Sherman-Morrison-Woodbury formula may be numerically unstable [26, 72]. However, in our case the risk is minimal. We assess the stability of the approach by a combination of theoretical analysis and empirical measurements as follows.

We invert M_1 with the Sherman-Morrison-Woodbury formula 5.26:

$$\begin{aligned} M_1^{-1} &= (\Lambda_{13}^2 + VV^T)^{-1} \\ &= \Lambda_{13}^{-2} - \Lambda_{13}^{-2} V (I + V^T \Lambda_{13}^{-2} V)^{-1} V^T \Lambda_{13}^{-2} \end{aligned} \tag{5.32}$$

A.

w_{bus}	# var.	# cons.	# LP iter.	time/iter. (s)	time/iter/cons. (ms)
4	2393	4752	18	0.98	0.214
8	9529	18976	20	1.74	0.092
16	38009	75840	24	7.20	0.095
32	141801	303232	17	30.41	0.100
64	606713	1212672	13	137.84	0.114

B.

w_{bus}	B	$M_6 - M_3^T \Lambda_{13}^{-2} M_3$	$(V^T \Lambda_{13}^{-2} M_3)^T B (V^T \Lambda_{13}^{-2} M_3)$	Cholesky
4	<0.01	0.04	0.01	<0.01
8	0.01	0.12	0.05	<0.01
16	0.07	0.55	0.34	0.02
32	0.29	2.49	2.40	0.08
64	1.21	9.28	14.11	0.56

Table 5.2: Example filter designs for buses of various sizes with 4×4 pre-equalizers, $n_{\text{bus}} = 10$, $r_{\text{bit}} = 8$, $r_{\text{tap}} = 2$, and $k_{\text{mask}} = 8$. Panel B shows the time breakdown for critical steps of the linear system solver for a single linear system solve. All numbers reported are in seconds. These times are for a 900MHz, UltraSparc III Processor.

recalling that each of the Λ matrices is diagonal with $\Lambda_{13}^2 = \Lambda_1^2 + \Lambda_3^2$, $\Lambda_{56}^2 = \Lambda_5^2 + \Lambda_6^2$, and $V = W^T\Lambda_{56}$.

First of all, note that the inversion is done for a symmetric low rank correction, VV^T, of a positive diagonal matrix, Λ_{13}^2, whose entries are bounded away from zero and do not exhibit fast growth throughout the iteration (figure 5.3 A and B illustrate this for a particular example).

Secondly, $V^T\Lambda_{13}^{-2}V$ is diagonal with all nonzero elements positive. Hence, $(I+V^T\Lambda_{13}^{-2}V)$ is diagonal and its eigenvalues are bounded away from zero. Therefore, the inversion of $(I + V^T\Lambda_{13}^{-2}V)$ is easy (see equation 5.27) and does not introduce ill-conditioning.

Moreover, the correction, VV^T, has rank $k_{\text{mask}}w_{\text{bus}}$ and is block diagonal; thus it is sufficient to consider each of the $k_{\text{mask}}w_{\text{bus}}$ blocks separately. Let $M_{1,i}$ denote the i^{th} diagonal block of M_1. Let $\Lambda_{13,i}^2$ be the portion of Λ_{13}^2 corresponding to $M_{1,i}$. With W as shown in equation 5.25, we have

$$M_{1,i} = \Lambda_{13,i}^2 + \Lambda_{56}^2(i,i)ee^T \tag{5.33}$$

where e is a column vector of all ones and $\Lambda_{56}^2(i,i)$ denotes the i^{th} diagonal element of Λ_{56}^2. Hence $M_{1,i}$ is a positive rank-1 update of a positive diagonal matrix. Let $\zeta_{1,i}, \zeta_{2,i}, \ldots, \zeta_{n,i}$ (in decreasing order) denote the eigenvalues of $M_{1,i}$. Let $\xi_{1,i}, \xi_{2,i}, \ldots, \xi_{n,i}$ (in decreasing order as well) denote the eigenvalues of $\Lambda_{13,i}^2$. Thus, $\xi_{1,i}, \ldots, \xi_{n,i}$ are just a reordering of the diagonal elements of $\Lambda_{13,i}^2$ and are all positive as well. According to the interlacing eigenvalue theorem [70, pp. 103-104][22, Theorem 8.1.8], the eigenvalues of the updated matrix interlace with the eigenvalues of the original matrix:

$$\zeta_{1,i} \geq \xi_{1,i} \geq \zeta_{2,i} \geq \xi_{2,i} \geq \cdots \geq \zeta_{n,i} \geq \xi_{n,i} \tag{5.34}$$

with

$$\sum_{j=1}^{n}(\zeta_{j,i} - \xi_{j,i}) = k_{\text{block}}\Lambda_{56}^2(i,i) \tag{5.35}$$

where k_{block} is the length of e, i.e. the size of each block. Thus, $\zeta_{1,i}$ is at most $\xi_{1,i} + k_{\text{block}}\Lambda_{56}^2(i,i)$ and $\zeta_{n,i}$ is at least $\xi_{n,i}$. For typical filter synthesis problems, k_{block} is on the order of a few hundred. The well-behaved nature of Λ_{13}^2 (Figure 5.3A and B) and Λ_{56}^2 (Figure 5.3C) leads us to believe that the numerical inversion involved in computing the Schur complement following the procedure we have laid out is not bound to introduce ill-conditioning beyond that already present in the normal equations $A^T\Lambda^2A$.

To test this conjecture, we observed the extremal values of Λ_{13}^2 while solving a typical filter synthesis problem, in particular, the synthesis of a pre-equalization

filter for a bus with 16 differential pairs. Panels A and B of Figure 5.3 depict the minimal and maximal values of Λ_{13}^2 throughout the iteration. The graph shows that these value are moderate (both in terms of being bounded away from zero and having a moderate upper bound) even as the solution is close to convergence. Panel C of Figure 5.3 shows that the maximal value of Λ_{56}^2 is moderate throughout the iteration. Thus, the inversion of M_1 with the Sherman-Morrison-Woodbury formula does not introduce serious ill-conditioning.

Furthermore, the residuals for the Schur complement solver throughout the iteration were consistently close to machine precision. Since the relative error is bounded (for any linear system) by the norm of the relative residual multiplied by the condition number of the matrix, we computed the condition numbers of the Schur complement matrix throughout the iteration (see Figure 5.3C). From the fifth iteration, the condition number grows steadily; this is as expected because interior point methods have an inherent ill-conditioning due to driving the barrier parameter to zero in order to penalize constraint violations. Using techniques such as the Sherman-Morrison-Woodbury formula and the Schur complement do not eliminate this underlying ill-conditioning. Thus, the condition number of the Schur complement grows as the iterations progress just as that for the normal equations. However our approach does not appear to add any further ill-conditioning and the Schur complement matrix stays very well conditioned even in the final iterations. While an empirical study such as this cannot *prove* the stability of the algorithm for all cases, the observation that the matrices remain very well-behaved strongly supports our belief that our approach is quite stable in practice.

5.5 Time and Space Requirements for the LSQ Approach

In order to compare our l_∞ methods with traditional l_2 techniques we implemented least-squares (LSQ) optimization for pre-equalizer synthesis as described in Section 3.4.3. We briefly address the time and space requirements for the LSQ approach in this section.

Let G_{lsq} denote $\begin{bmatrix} \Upsilon_u G_u \\ \Upsilon_d G_d \end{bmatrix}$ in the LSQ formulation (see equation 3.29). The number of variables in the LSQ problem is the number of filter coefficients, k_{fir}. G_{lsq} has $w_{\text{bus}}^2(n_{\text{fir}} + n_{\text{bus}})k_\Upsilon$ rows, where k_Υ is the number of non-zeros in one period of the Υ function. Typically, $n_{\text{bus}} > n_{\text{fir}}$ and $w_{\text{bus}} > w_{\text{fir}}$. Hence, G_{lsq} has many more rows than columns. Moreover, as discussed in Section 5.1, note that the g vectors have the following properties:

1. $g(i,j,t)$ has no overlapping nonzero elements with $g(k,j,t)$ where $k \neq i$.
2. $g(i,j,t)$ only has a fraction of $1/w_{\text{bus}}$ elements nonzero.

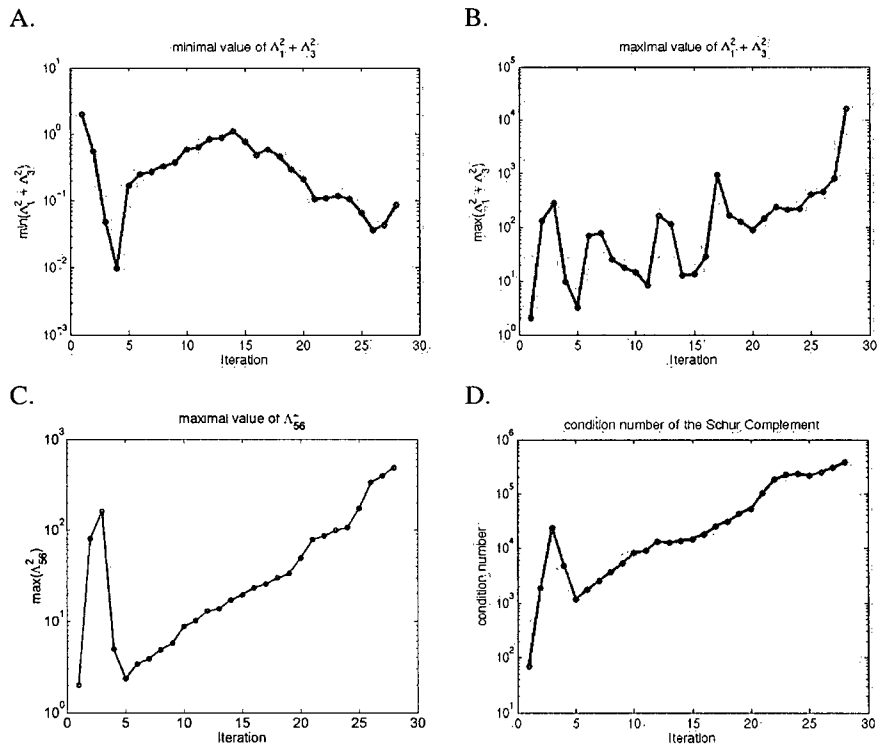


Figure 5.3: A. Minimal values of Λ_{13}^2, B. Maximal values of Λ_{13}^2, C. Maximal values of Λ_{56}^2, D. Condition number of the Schur complement throughout the iteration.

The first observation means that G_{lsq} can be permuted into a block diagonal matrix with w_{bus} blocks. Each block is a $(w_{\text{bus}}(n_{\text{fir}} + n_{\text{bus}})k_\Upsilon) \times (k_{\text{fir}}/w_{\text{bus}})$ matrix. Because G_{lsq} is block diagonal, the LSQ problem given in equation 3.29 is equivalent to w_{bus} independent subproblems.

As described in Section 3.4.3, we implemented the least-squares optimization using MATLAB's built-in mldivide function. For sparse, over-determined problems, mldivide uses Householder reflections to compute a Q-less QR factorization of G_{lsq} [65]. This factorization allows the solution of sparse least-squares problems in two steps: a QR factorization and a triangular linear system solve [67, p. 83]. For a LSQ problem of size $m \times n$, the QR factorization based on Householder reflections can be done in $O(mn^2)$ operations [67, p. 75]. As we noted above, we are effectively solving w_{bus} independent subproblems of size $(w_{\text{bus}}(n_{\text{fir}} + n_{\text{bus}})k_\Upsilon) \times (k_{\text{fir}}/w_{\text{bus}})$. Therefore, in total, the QR factorization step takes $k_{\text{fir}}^2(n_{\text{fir}} + n_{\text{bus}})k_\Upsilon$ operations, which is $O(w_{\text{bus}}^2 m_{\text{fir}}^2 w_{\text{fir}}^2(n_{\text{fir}} + n_{\text{bus}})k_\Upsilon)$. The triangular linear system solve takes $O(k_{\text{fir}}^2)$ operations, which is $O(w_{\text{bus}}^2 m_{\text{fir}}^2 w_{\text{fir}}^2)$. Thus, mldivide solves the LSQ problem given by equation 3.29 in $O(w_{\text{bus}}^2 m_{\text{fir}}^2 w_{\text{fir}}^2(n_{\text{fir}} + n_{\text{bus}})k_\Upsilon)$ operations.

Moreover, because the subproblems are independent, the Householder reflections don't introduce fill-in. Therefore, the space requirement for setting up and solving the LSQ problem is $O(w_{\text{bus}}^2 w_{\text{fir}} m_{\text{fir}}(n_{\text{bus}} + n_{\text{fir}})k_\Upsilon)$, which is the space required by the matrix G_{lsq}.

The mldivide routine in MATLAB is highly optimized. In practice, for the examples presented in this thesis, the LSQ problems are solved in a few seconds when running on a 900MHz, UltraSparc III processor.

In the LP formulation, we set up constraints to constrain the overdrive power of the filter. However, in the LSQ formulation, such constraints cannot be set up directly. To constrain the power of the filter designed, linear constraints can be set up to bound the maximum magnitude of each individual filter coefficients. In those cases, MATLAB LSQ solver lsqlin is used to solve the LSQ problem with linear constraints. This significantly slows down the optimization procedure. Moreover, this tends to lower the eye height significantly while no significant decrease in the maximum filter output. Thus, we did not report results with bounds on the filter coefficients in Chapter 3.

5.6 Summary

The linear programs for l_∞ optimal filter synthesis grow quadratically in size with the number of links in the bus. To make the l_∞ synthesis approach practical, we implemented Mehrotra's interior-point algorithm with an efficient linear system solver that exploits the problem-specific properties of our particular constraint ma-

trix. The special properties of the normal equations allow us to apply the Sherman-Morrison-Woodbury formula and obtain an efficient linear system solver. The linear system solver runs in cubic time in the width of the bus and makes the LP approach practical. We note that it is slower than the LSQ approach which runs in quadratic time in the width of the bus. However, in practice, the width of a parallel bus is limited (see Section 2.1). Moreover, compared with the traditional LSQ approach, the LP approach achieves better equalizer performance while offering a more flexible framework. The LP approach allows direct control of critical parameters such as the overdrive ability of the filters and maximum overshoot at the receiver. We also presented analysis and numerical evidence that the linear system solver is numerically stable. These make the LP approach a practical and superior alternative to the traditional LSQ approach.

Chapter 6

A Proof-of-Concept Test Bed

Equalization filters are one part of a high-speed link. The complete design must also take into account the design of high-speed DACs and ADCs, low jitter PLLs and DLLs, state-of-the-art packaging, detailed electrical modelling of the interconnect, etc. Complete designs require custom-chip design, many designers, expensive equipment, and design cycles of a year or more. While such efforts are necessary to bring interconnect solutions to the point that they can be utilized by system architects, only a few research groups can undertake such resource-intensive efforts.

Smaller groups like us often focus on particular interconnect challenges by using simulation-based studies. Simulations provide a practical way to examine issues in cutting edge interconnect. However, simulations inevitably rely on simplified models for the bus and connectors, as well as for clock jitter and other phenomena that degrade signal integrity. For example, accurate electrical models for buses and connectors are difficult to obtain without measurement as they require 3D solutions to Maxwell's equations. Numerical solutions for these systems are intractable for realistic interconnect. Thus, we need a physical implementation while avoiding massive design efforts and expensive test equipment. We achieve this by targeting much lower bit rates and deliberately designing PC board buses to exacerbate signal integrity issues at these lower frequencies. This "scale-model" approach has numerous advantages: it is inexpensive to implement; the design cycle is dramatically shortened; and we can easily alter the test bed to examine the impact of varying individual components of the link. Unlike simulation alone, the physical implementation forces us to address a wide range of issues arising in an actual design.

Section 6.1 describes how our test bed uses commodity PC graphics cards to provide analog channels operating at up to 300M samples/sec. Section 6.2 demonstrates the effectiveness of our test bed by using it to evaluate filters designed. These experiments show the impact of timing jitter and inaccuracies in channel estimation. The test bed also confirms that our filters offer dramatic improvements in signal integrity. Its low cost and simplicity should allow our test bed design to be easily used and modified for a wide range of research in signal integrity and mixed-signal design.

6.1 The Test Bed

We designed our test bed around inexpensive VGA graphics cards. Each graphics port provides three analog outputs for the red, green and blue color channels. Using dual-port, PCI graphics cards and a typical PC with five available PCI slots, we can implement a test bed with up to 30 analog channels. To use graphics cards as transmitters in a synchronous transmission system, we must synchronize the video outputs so that they have identical pixel rates and so that frames and scanlines are aligned. Section 6.1.2 presents our solution to synchronizing the graphics cards. Typical graphics cards support pixel rates of up to 250MHz-400MHz. To compensate for high-frequency losses, our equalizing filters have sample rates that are a small multiple of the data rate. For example, using four samples per data bit limits the data rate of our test bed to 60 $\sim$ 100Mbits/sec. Thus, the buses in our test bed must have crosstalk and other losses comparable to those seen at multi-Gbs rates while operating at a much lower speed. Section 6.1.3 describes the issues involved in designing a sufficiently bad PC board bus. Finally, once the test bed is implemented, we must measure its electrical parameters to validate simulation models and provide a basis for filter synthesis. Section 6.1.4 describes how we perform these measurements using our test bed.

6.1.1 System Overview

Figure 6.1 shows our test bed. It consists of three major components: a host PC with multiple PCI graphics cards, a PCB with a bus under test and impedance matching networks, and an oscilloscope as a receiver. The bus under test is driven by the video outputs of the graphics cards. We used ATI Radeon 7000 VE graphics cards at a cost of about $US 60 each. Each card provides separate VGA and DVI graphics ports providing a total of six analog channels per card. A run of seven boards cost $US 350, mainly because of the area needed for a long bus. Not counting the oscilloscope and signal generator which are available in most electronic labs, our test bed can be built for less than $US 2000.

After characterizing the bus, we use our filter synthesis methods to derive the filter coefficients and determine their worst-case input sequences. We then compute the filter outputs and store them as pixel values that are transferred to the frame buffers of the graphics cards. As result, the RGB signals from the cards drive each wire of the bus with the values from the corresponding filter. At the receiver end, the oscilloscope records the bus outputs which we then download to the host PC for analysis.

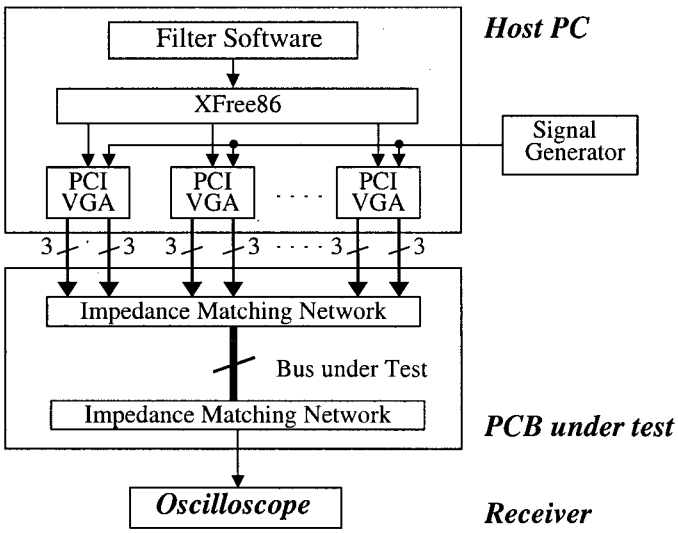


Figure 6.1: The test bed.

6.1.2 Synchronizing Multiple Graphics Cards

In our test bed, we match the pixel clocks of the graphics cards by modifying the graphics cards to operate from a common clock. We then use a software approach to align the frames and scanlines from the multiple graphics ports.

The Radeon 7000 derives its pixel clocks from a 27MHz crystal oscillator. We added connectors that allow us to override the oscillator with the signal from an external signal generator, an Agilent 33250A. The cards track the signal generator from 27MHz to 29MHz. Adding the connector for an external clock is the only physical change that we made to the graphics cards.

With the pixel clocks matched, we need to align the frames and scanlines to ensure sufficient scanline overlap to test our filters. This is called "genlock." While graphics cards with hardware support for genlock are available, they are specialty devices with prices in the thousands of dollars. Using a large number of these expensive cards would increase the cost of our test bed by an order of magnitude. Instead, we adapted a software approach to genlock originally described in [4]. The main idea is to temporarily alter the length of the horizontal and vertical retrace times of each graphics port to bring all of the ports into alignment. The solution in [4] used software observation of the retrace intervals to achieve frame level synchronization across multiple PC's running real-time Linux. Using the same

general approach on a single PC running RedHat Linux and XFree86, we use an oscilloscope to make precise measurements of sync pulse alignments and achieve synchronization to within a few pixel times.

VGA standards [1] specify the registers that control the video signal generation. To access these registers, we modified the graphics card driver in XFree86 [2] and extended the VIDMODE feature of XFree86 so that synchronization can be performed at the application level. With this method we achieve synchronization to within 15 pixels. In particular, this software approach is limited because the graphics card that we used restricts the number of pixels in a scanline to be a multiple of eight; it does not provide software observation of the horizontal synchronization events; and changes made to the timing registers appear to cause some small random perturbation in horizontal sync timing. For our application, the residual offset is not a problem and can be accommodated by shifting the location of the filter outputs in the frame buffer. Using the 1600×1200, video mode (the longest supported scanline for the DVI channel) with a filter oversampling rate of four pixels per bit provides test sequences of up to 396 bits per scanline which is more than adequate for our purposes.

6.1.3 A Scale Model for High-speed Buses

The Radeon 7000 graphics cards provide 300MHz RAMDACs. With four filter samples per data bit, this sets the upper limit for the data rate of our test bed to be 75MHz. To obtain a signaling environment at these bit rates similar to that of a multi-Gbs bus, the bus geometry must promote crosstalk. To that end, we fabricated a bus that is 1 meter long with 6 mil (0.15mm) wide traces with 6 mil spacing running 93 mils (2.36mm) over the ground plane with a standard, FR-4 dielectric and 0.5oz (0.017mm) copper thickness. The large separation to the ground plane increases the inductive coupling to produce significant crosstalk at relatively low data rates. Using the HSPICE 2D field solver, we extracted a model for the bus and simulations showed that without equalization it should have a closed data eye at 70 Mbits/sec. The board is 15in $\times$ 10in (38cm $\times$ 25cm).

Figure 6.2 shows the resistive impedance matching network that we use. Achieving sufficient crosstalk for our experiments necessitated a bus geometry with a relatively high characteristic impedance, about 100Ω. The RGB outputs from the graphics card are 75Ω, and the input to the oscilloscope (an HP 54522A) is 50Ω. To address these impedance mismatches, we considered transformers and resistive networks. To use transformers, we would have to restrict our data sequences to minimize the low-frequency content of the transmitted signals. On the other hand, insertion loss is the main disadvantage of using a resistive network. We chose the resistive network approach for its ease of implementation and the flexibility of be-

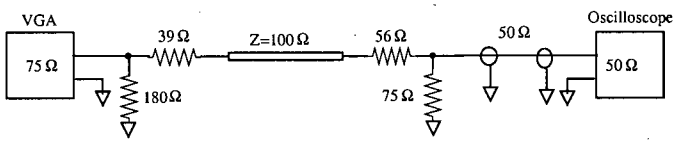


Figure 6.2: Impedance matching networks.

ing able to change the termination simply by changing resistor values. The voltage output from this network has about 40% of the magnitude of an implementation with ideal transformers.

Even with impedance matching networks, there are many discontinuities in the signal path. These include the VGA and DVI connectors, DVI-VGA adapters, header-pin connectors, and some non-impedance controlled segments on the board and between the VGA connectors and the header pins. As discussed in Chapter 2, connectors and chip packages for multi-Gbs links have many non-idealities as well.

6.1.4 Bus Characterization

A challenge in applying equalizing filters in high-speed I/O links is channel estimation. In current practice, filter coefficients for single-line pre-emphasis are manually adjusted through either trial-and-error or channel analysis [76]. For bus characterization, a time-domain reflectometer (TDR) or a vector network analyzer (VNA) can be used. Our test lab has a 2-port VNA. To use a 2-port VNA for a multi-port system, a matched terminator is required to obtain accurate measurements. Furthermore, a w-bit bus is a $2w$-port network and requires $\binom{2w}{2}$ measurement configurations. For the nine-wire bus described in Section 6.2, this requires 153 test configurations with manual changes for each. Furthermore, VNA measurements would not include circuitry and interconnect on the graphics cards such as the DACs, graphic chip packaging, and graphics card copper traces.

Instead of using the VNA, we perform channel estimation *in situ*. Our optimization methods are formulated with respect to the response on each bus output to a pulse whose width is one data bit time on each bus input. For our test bed, we program the graphics cards to produce such pulses and measure the responses directly using the oscilloscope. This method takes the entire channel including the DACs, connectors, resistive networks, and bus into account. For a w-bit bus, we only need w^2 configurations. This number can be further reduced to w by sending out bits consecutively on each wire. With a multi-channel oscilloscope, the number of configurations is only $w/(c-1)$, where c is the number of oscilloscope

ports. That is only 9 configurations for our nine wire bus with a two-channel oscilloscope. Unlike a VNA which uses a tuned front end, our approach is highly sensitive to noise. As discussed in Section 6.2.1, we average the response to 100 measurements per configuration to obtain satisfactory accuracy. For real links, simple on-chip oscilloscope circuits [28] could be used to perform channel estimation *in situ*. Due to the simplicity of the on-chip oscilloscope circuits, they are less accurate than the oscilloscope used in the test labs. One might need to average many runs to get acceptable accuracy.

6.2 Results

Using our test bed, we have performed some experiments with our filters. Figure 6.3 shows the complete test flow. For a given filter structure and the set of bit responses obtained with the *in situ* bus characterization method described above, we derive a set of filter coefficients and its corresponding worst-case input sequences with the filter synthesis approach described in previous chapters. We then compute the filter outputs and render them to the graphics cards. We drive the bus with the RGB outputs of the graphics cards and measure the bus output signals with the oscilloscope. We then upload the samples from the oscilloscope through a GPIB connection back to the host PC, post-process the samples and generate the worst-case eye diagram for the channel.

This section describes our *in situ* measurements of bit responses, measurements of filter effectiveness, and the impact of timing jitter. Pragmatism motivated us to choose a nine-wire bus for these experiments. We had two graphics cards which offer 12 channels in total. One graphics port failed and our supplier didn't have more cards available. Fortunately, nine wires were wide enough to show large-scale crosstalk and sufficient to demonstrate our filters.

6.2.1 Bus Characterization

Figure 6.4A shows a complete bit response on the middle wire of the bus, wire 4, given a single, bit-wide input on each wire consecutively. The response to the pulse on the wire itself (Figure 6.4B) has a peak at 128 $\sim$ 148mv, with wires driven directly by the VGA port having a higher peak than those driven through DVI$\rightarrow$VGA adapters. This is apparently due to high-frequency losses of the DVI$\rightarrow$VGA adapter. All wires of the bus, except for the middle three wires are driven by DVI ports via DVI-VGA adapters. This is an example of how our *in situ* channel estimation approach incorporates the entire link. Figure 6.4C shows the maximum crosstalk on wire 0 from the other wires. Note that coupling from wire

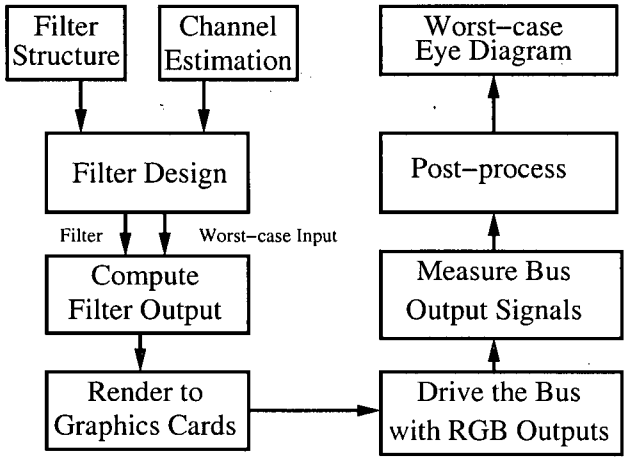


Figure 6.3: Test flow.

2 is stronger than that from wire 1; otherwise, crosstalk decreases with distance as expected. This suggests that for this relatively tight bus configuration, crosstalk cancellation for nearest neighbor might not be sufficient.

We observed a 5mv peak-to-peak noise floor. With coupling terms from 2mv to 15mv, we averaged the values from 100 measurements to estimate a bus model. We automated these measurements using GPIB programming of the oscilloscope. From the standard deviation of these measurements, we estimate that our bus model is accurate to within $\pm 3\%$. Using the average from 50 traces degrades the accuracy for filter synthesis to $\pm 8\%$ with little impact on end-to-end signal integrity. This suggests that simple, on-chip circuitry [28] for channel estimation should suffice when implementing our filters for multi-Gbs links.

6.2.2 Filter Performance

As defined previously in Section 3.4.2, we write that a filter is $m \times k$ to indicate that the filter has m taps and considers k neighbors to each side. We test each filter using its worst-case input sequence (see Section 3.5) and random variations. Since we only have $\sim$ 400 bits worth of useful data per scan-line (see Section 6.1), for the eye diagrams shown in Figure 6.5, we used the worst-case input sequences for the middle wires, which have the most severe crosstalk. Given the worst-case input sequence, the corresponding filter output is then calculated, stored and then rendered to corresponding displays. Because of the limited overdrive ability of the

RAMDACs, the filter outputs are scaled to $[0, 255]$ in pixel values, which results in different high and low values for filters with different overdrive.

Figure 6.5 shows eye diagrams measured from the bus with various filters. As predicted by the HSPICE field solver and simulation, this bus has a closed eye at 70Mbit/s, Figure 6.5A. For single-line pre-emphasis, neither the LP nor the LSQ synthesized filters open the eye. As the LP method optimizes for the worst-case, it can make no progress and produces meaningless filter coefficients. The LSQ method, on the other hand, optimizes for the average case and produces a filter that provides some improvement in signal integrity. Thus, we show a single-line pre-emphasis filter designed by the LSQ method in Figure 6.5B. While the LSQ method cannot open the eye with a 12×1 filter, the LP method achieves 14% eye height and 15% eye width in simulation that neglected jitter. However, due to jitter and voltage noise in the physical system, we do not observe this small eye opening using the test bed, Figure 6.5C. By taking more wires into account, the 12×3 filters designed by the LP method produces a good eye opening, Figure 6.5D. Figure 6.5E shows that an excellent eye is produced by a full-width filter using the LP method, while Figure 6.5F shows that the corresponding LSQ filter is not nearly as effective. This shows the advantage of the LP method that maximizes eye masks directly. These observations are consistent with simulation results reported in previous chapters and confirm the correctness and practicality of our filter design approach.

6.2.3 Jitter

High-speed I/O bandwidth should scale with technology as long as the timing uncertainties can be made to scale at the same rate [76]. Clock jitter and channel interference are the dominant causes of timing uncertainty. Although equalizing filters can greatly reduce channel interference, clock jitter can significantly degrade channel performance [8].

We face two timing issues in this test bed: subpixel misalignment and graphics card PLL jitter. The subpixel misalignment between video ports is the residual misalignment after shifting the location of filter outputs in the frame buffer (see Section 6.1.2). We measure this subpixel misalignment and incorporate it into the channel model. Although pixel clock jitter is not specified by graphics card manufacturers, it is significant. For the cards we used, the jitter appears to be random with a standard deviation of approximately 200ps. The excellent eye-diagrams shown in Figure 6.5 suggest that although we synthesized the equalizing filters assuming perfect timing, the filter design approach tolerates moderate amounts of jitter.

With 200ps rms random jitter, to ensure a bit error rate (BER) of 10^{-12}, the system should tolerate 2.8ns peak-to-peak jitter. With randomly generated extreme-

case jitter of either $+1.4$ns or -1.4ns, we simulated the transmission system with the 12×9 equalizing filter designed by the LP method. The open eye shown in Figure 6.5G suggests that the system provides 10^{-12} BER and further confirms that the filter design approach has some jitter tolerance. However, compared with Figure 6.5D, Figure 6.5G shows reduction in both eye height and eye width. This suggests the importance of taking jitter into account while designing equalizing filters in order to guarantee a specified BER. Incorporating jitter explicitly in our filter synthesis procedure is a topic for future research. Once we develop a jitter-aware synthesis procedure, we should be able to use our test bed to validate its effectiveness.

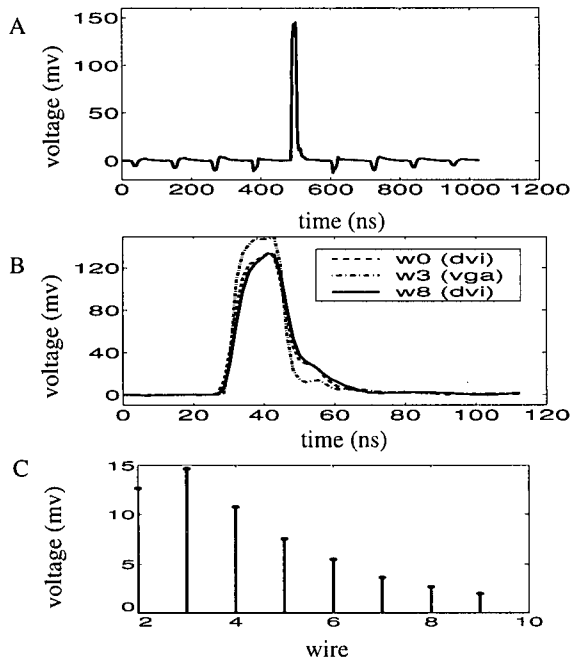
6.3 Summary

This chapter presented a simple, low-cost, signal integrity test bed for PCB buses. We used commodity graphics cards to provide a large number of analog channels at rates of 300-400M samples/sec. By designing buses with exaggerated crosstalk, we modelled channels similar to those found in multi-Gbs links at a much lower rate. Because our approach uses commodity components and does not require custom-chip fabrication, we can perform experiments with an order-of-magnitude or more reduction in cost, time, and effort compared with full-speed links. This allows us to quickly evaluate novel signaling approaches.

The low-speed scale model does not account for every aspect of a high-speed bus. For example, high-frequency losses are much more severe at multi-GHz rates where dielectric loss and skin effect are both more severe. Although we saw significant jitter in our test bed, the impact of jitter at multi-GHz rates is likely to be even greater. Similar remarks can be said for ground bounce, power supply noise, etc. To complete the evaluation of our filters for multi-GHz rates, it would be necessary to design the high-speed chips and boards, and integrate them with the appropriate packages and connectors. We could then measure the bit-error rate under suitably adverse operating conditions and determine the performance gain achieved by our equalizers. As described at the beginning of this chapter, such an evaluation is a large effort, beyond the scope of a single, Ph.D. dissertation.

We demonstrated our test bed by using it to validate our synthesis procedures for crosstalk cancelling equalization filters. Measurements from the test bed show that we can get results that are comparable to those predicted by simulation. We also observed the degradations caused by non-idealities such as timing jitter and imperfect connectors. Although a full-speed implementation is still necessary before application, our test bed allows simple and fast proof-of-concept validation and demonstration of new signalling methods as we develop them. Our test bed

promises the identification of opportunities and potential problems early in the design cycle complementing simulation and full-speed implementations to lower the total cost and time required to implement novel high-speed buses.

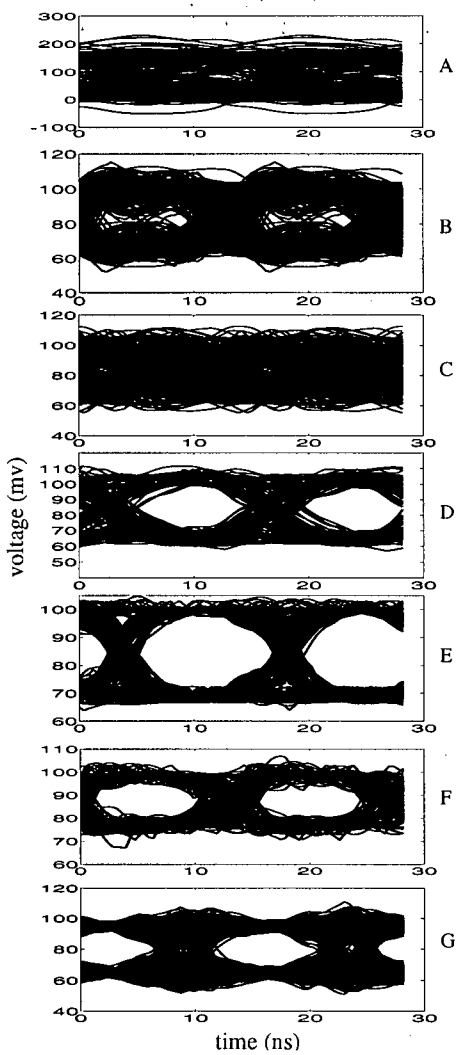


A: the cascaded waveform of the responses on wire 4 given a single, bit-wide pulse on each wire of the bus;

B: the responses on wire 0, 3 and 8 given a single, bit-wide pulse on wire 0, 3 and 8 respectively;

C: the maximum magnitude on wire 0 of crosstalk from each other wire.

Figure 6.4: Bus bit responses.



A: without filter;

B: with independent pre-emphasis on each wire, LSQ synthesized filters;

C: with 12×1 LP synthesized filter for nearest-neighbor crosstalk cancellation;

D: with 12×3 LP synthesized equalizing filter;

E: with 12×9 LP synthesized equalizing filter;

F: with 12×9 LSQ synthesized equalizing filter;

G: simulation for 12×9 LP filter with extreme-case jitter.

Figure 6.5: Eye diagrams at 70Mbit/sec/wire.

Chapter 7

Conclusion and Future Work

In this thesis, I have presented a comprehensive framework for equalization filter synthesis for high-speed digital buses with linear models. These methods allow direct optimization of eye masks, support arbitrary, linear models for buses and provide joint optimization of pre-equalizers, decision-feedback equalizers, and near-end crosstalk cancelling filters. This approach is based on the l_∞ optimization for worst-case response using linear programming. I first described the use of l_∞ methods for high-speed links in [54]. In that work, I only considered pre-equalization filters, the objective functions corresponded to eye height rather than the more general eye masks supported here, and I used a simplistic model for the bus where all wires were assumed to have the same characteristics. The present research removes all of these restrictions, provides an efficient solver for the linear programs arising from the filter synthesis problems, and validates the novel filters with a low-cost, low-speed test bed.

I first introduced parameterized eye masks which allow the direct optimization of eye masks. Using these masks, I formulated the resulting l_∞ optimal pre-equalization filter synthesis problem as an instance of linear programming. For the single-ended bus that I considered (see Section 3.5.1), filters designed by this method increased bus bandwidths by more than a factor of seven compared with a bus with no filters and more than a factor of three compared with a bus with only single-line pre-emphasis and nearest neighbor crosstalk cancellation. Moreover, parameterization of the eye mask gives the designer great flexibility for specifying trade-offs between eye height, eye width, and other details of the eye shape. Critical parameters such as filter overdrive ability and overshoot can be naturally incorporated as linear constraints. This is in contrast with the commonly used l_2 approaches for filter design where error is minimized only in the average sense and these other parameters can only be specified indirectly. Accordingly, filters designed with this method significantly outperform LSQ designed filters in terms of eye height, overshoot, and other eye-mask parameters.

I then presented a unified approach for synthesizing optimal filters for the three most common forms of equalization: transmitter pre-equalization, decision-feedback equalization, and near-end crosstalk cancellation. I showed that the resulting optimization problem remains linear under reasonable assumptions and can

be solved using linear programming. To illustrate the use of these methods, I examined the design of intra-board and cross-backplane links. The link models included chip-packaging parasitics, impedance mismatches, dielectric and skin-effect losses, and connector parasitics. I evaluated different design trade-offs such as the use of simultaneous bidirectional signalling and multi-level signalling. The unified approach allows early identification of bandwidth bottlenecks and rapid evaluation of design trade-offs. I briefly discussed the area cost and latency cost of the equalizing filters for a straightforward implementation based on pipelined adder trees. I noted that these costs are acceptable and certainly are justified by their performance advantages for bandwidth critical applications. Power consumption is obviously an important issue that merits further study.

The resulting linear programs are large. The number of variables and constraints is dominated by the number of disturbances k_{disturb} which grows quadratically with the width of the bus. To solve these large LPs, I implemented Mehrotra's interior-point method with sparse matrix techniques. Exploiting the specific structure of the constraint matrix arising in this approach, I developed an efficient linear system solver that runs in $O(k_{\text{disturb}}k_{\text{fir}} + k_{\text{fir}}^3)$ time. Moreover, in practice, the number of LP iterations stays roughly constant, around twenty for the examples presented in this thesis. This makes the linear programming approach a competitive alternative to least-squares based methods.

To validate the approach with real hardware, I presented a simple, low-cost, signal-integrity test bed for PCB buses. I used commodity graphics cards to provide a large number of analog channels at rates of 300-400M samples/sec. By designing buses with bad signal integrity at such low speeds and improving their performance with the equalization filters designed, I validated the equalization filter synthesis procedures. Measurements from the test bed are comparable to those predicted by simulation in the presence of real world effects such timing jitter and power noise.

In summary, this thesis combines modelling, optimization and prototyping to demonstrate that linear programming provides a practical, effective and flexible framework for designing equalization filters that greatly increase the bandwidth of high-speed off-chip buses. Furthermore, Moore's law [45] favors this approach: logic circuits will continue to get cheaper and faster, increasing the demand for off-chip bandwidth while reducing the size and power consumption of the filters. Packaging and PC board technology progresses at a much slower pace. Thus, I expect sophisticated equalization filters to become prevalent with technology scaling. The first applications are likely to be large SMP multi-processors [12] and high-speed network switches [43, 69] where bandwidth is critical and backplane/centerplane resources are critical. With further technology scaling, and such filters will migrate downward into a wider range of products as off-chip bandwidth becomes critical for more and more applications.

7.1 Future Work

This thesis has demonstrated the effectiveness of MIMO equalization filters for high-speed off-chip buses. This section considers some natural extensions of this work as well as other possible applications of equalization techniques.

- **Reducing the Hardware Complexity**
  The current work assumes that every tap of a filter considers the same number of neighboring wires; in other words, the filters have a rectangular shape. The crosstalk between wires far apart is typically small and short-lived, thus requiring less equalization. Non-rectangular filters that vary the number of taps for each pair of wires according to he strength of the coupling could reduce the hardware cost of the equalization filters with negligible impact on performance. A straightforward approach is to first design rectangular filters and then remove the filter coefficients that are smaller than some threshold. With the shape of the filter determined, a new LP can be set up to derive a new set of optimal equalization filter coefficients. More sophisticated methods could use heuristic search techniques such as simulated annealing [38] to find the optimal filter shape. The filter synthesis methods presented in this thesis could be used for each configuration considered.

  The bus models used here include coupling terms for every pair of wires in the bus. As noted above, crosstalk between widely separated wires is typically small. Ignoring coupling between widely separated wires can reduce the size of the LPs, thereby reducing the time and memory required by this approach and improving its practicality.

- **Adaptive l_∞ Filter Design**
  As discussed in Chapter 2, there are three common methods for setting equalization filter coefficients: "lookup table and forget", "adapt once and forget" and "continuous adaptation". The latter two methods take manufacturing variations into account and hence offer significantly better performance than the first method which characterizes channels in the lab and derives a set of equalization filter coefficients later applied to all channels [73]. The approach presented in this thesis falls into the second category, "adapt once and forget". The channels are characterized *in situ* on power-up (see Chapter 6) and a set of l_∞ optimal filter coefficients are derived. Channel characteristics drift over time due to environmental variations such as temperature and humidity changes [73]. Continuous adaptation algorithms, such as sign-sign LMS [63, 73] takes these variations into account.

A natural extension of this work is to develop an adaptive scheme for l_∞ filter design. Because the optimization problem is convex, a simple hill-climbing strategy could be employed to improve the objective function given the old solution. Moreover, environmental variations change at a time-scale much slower than the targeted bit rates (multi-GHz), which means that re-calibration can use a slower, off-line procedure. The changes in channel characteristics can be viewed as perturbations to the original problem. Although the optimal vertex of the original problem is likely not optimal for the perturbed problem, it is likely to be close. With a solution close to optimal, it might be possible to use variations of the Simplex method [46] which might only take a few simple steps to converge to the true optimal solution.

- **Receive Equalization**
  I presented a unified optimization framework for pre-equalization, near-end crosstalk cancellation and decision-feedback equalization. As noted in Chapter 2, some designs also employ receive equalization. Due to the complexity of implementing a receive equalizer at multi-GHz frequencies, pre-equalization is more popular in high-speed links. However, receive equalization may be simpler when adaptive methods are used. In particular, adaptive pre-equalization requires information to be transferred back to the transmitter end in order to adjust the filter coefficients. For example, Rambus reported a common-mode signaling system that creates a back-channel communication path over differential high-speed links for an adaptive differential high-speed link transceiver cell [27]. In contrast, channel characterization is readily available for receive equalization.

  The methods presented in this thesis can be directly applied to receive equalization. For example, if receive equalization is used instead of pre-equalization in the bidirectional link considered in Chapter 4, the optimization problem remains linear, and the LP formulation is the same with some straightforward modifications to the derivation of the G matrix. In this case, equation 3.16 becomes

$$\mathsf{Out} = \mathsf{F}^{\times n}\,\mathsf{B}^{\times n}\,\mathsf{In} \tag{7.1}$$

  where $n = q_{\text{bus}} + q_{\text{fir}} + q_{\text{In}} - 2$. As in Chapter 4, the matrices would need to be manipulated to treat the filter coefficients as the input to the system.

  If a system contains both pre-equalization and receive equalization, the optimization problem becomes quadratic and the linear programming approach developed in this work is no longer applicable.

- **Coded Signalling with Equalization**
  The idea behind a coding scheme is that a few input patterns may be responsible for the most serious crosstalk and ISI. One can design codes that avoid these worst-case patterns. Because such codes do not use all possible combinations of levels on the wires, the bits sent per symbol is reduced; however, the improvement in signal integrity can enable a higher symbol rate and thus a higher overall bandwidth than the same channel without coding. As a simple example, differential signalling can be understood as a coding technique where two wires are used to transmit a single bit. This provides current balancing and rejection of common-mode interference that in practice can greatly improve signal integrity. Equalization technique can be used with coded signalling to further improve signal integrity. In fact, I showed how differential signalling can be included in my filter synthesis framework in Chapter 4. Other schemes, for example, sending 3 signals on 4 wires while keeping current balance in the 4-wire bundle, might improve the overall bandwidth and efficiency compared with differential signalling and single-ended signalling [51]. Equalization filter design for channels that employ coded signalling techniques is a topic for future work. Note that the low-speed test bed can again be readily used to do proof-of-concept validation of such new techniques as we develop them.

- **Bit-Error-Rate Minimization**
  The performance of high-speed links is degraded by noise. Sources of noise in digital systems include power supply noise, electro-magnetic interference (EMI), intersymbol interference (ISI), crosstalk, ground bounce, process variation, thermal noise, and shot noise. Noise can be modelled by its effects on the signal level (voltage noise) or event times (jitter). While some of these disturbances such as ISI or ground bounce are caused by the digital behavior of the system and can be modelled deterministically, most of the noise sources listed above are best modelled as unbounded, random processes. When these noise sources are considered, it is no longer possible to guarantee error-free performance. Instead, one must settle for probabilistic measures, the most common of which is the bit-error rate (BER). High-bandwidth buses are normally specified with a maximum acceptable BER, with typical targets of with typical targets of 10^{-15} to 10^{-20} or lower.

  As described in Section 2.3, the problem of designing equalizing filters to minimize mean-square error in the presence of random noise has been extensively studied [15, 17, 30, 31, 63, 71]. However, this does not necessarily minimize bit-error rate. The l_∞ approach optimizes for the best worst-case performance in the case of no random noise. Although this would intuitively

result in low BER in the presence of random noise, it might not lead to the lowest BER depending on the properties of the actual noise.

To calculate the BER exactly requires the knowledge of the statistics of the data transmitted over the channel and of the noise. In practice, such detailed statistics are rarely available. Because the worst-case data pattern is typically determined by a few wires and a few consecutive symbols, it can be closely approximated by random data every few hundred or few thousand bits. Thus, the worst-case data pattern can be used to bound signal integrity and seek to minimize the worst-case BER. Although this may overestimate the BER by a factor of a hundred or a thousand, this is on the same order as other approximations that are necessary with the statistical data that is actually available. A topic for future research is to extend our l_∞ methods that model the thresholding behavior of digital links to include random noise.

- **Applications of Equalization Techniques**
  This work focused on the application of equalization techniques for off-chip buses to mitigate intersymbol interference, crosstalk and other linear distortions of the channel. Similar ideas can be applied to on-chip interconnects and optical interconnects.

  Intersymbol interference and crosstalk are present in on-chip buses and limit their performance. On-chip buses are RC transmission lines producing serious intersymbol interference at high data rates. Unlike off-chip links, the availability of active circuitry on chip means that designers can insert repeaters to improve wire delay and sharpen the rise time to some extent at the cost of additional power dissipation and chip area. Another method is to pre-emphasize the high-frequency components of the signal and effectively equalize the low-pass nature of the RC transmission line [18, p. 361].

  Crosstalk arises in on-chip interconnect primarily from capacitive coupling. In deep-submicron processes, wires are fabricated to be tall compared with their width and spacing. While this reduces wire-resistance, it also increases capacitive coupling between adjacent wires. The resulting crosstalk appears as a data-dependent delay of the interconnect. For example, if two adjacent wires switch in the same direction, the effective coupling capacitance is zero; thus the effective total capacitance decreases and so does the delay. If they switch in the opposite direction, the effective coupling capacitance is doubled; hence the effective total capacitance increases and so does the delay. Techniques such as crosstalk avoidance codes (CAC) [20, 68] reduce delay variations by eliminating worst-case coupling transitions in a bus at the cost of additional wires.

Recently, Sridhara *et. al.* combined decision-feedback equalization for ISI and CAC to improve on-chip communication speeds [61]. They used a simple variable threshold inverter as receiver and modified its switching threshold as a function of the past output of the bus. The equalization techniques studied in the current work can be directly applied to on-chip interconnect to mitigate the impact of crosstalk and intersymbol interference. Moreover, to further improve bandwidth, the current approach could be extended to design equalizing filters for coding schemes such as CAC.

Intersymbol interference and crosstalk are also present in optical interconnects. For example, insufficient channel spacing produces crosstalk between adjacent dense wavelength-division multiplexing (DWDM) channels [42]. Fibre nonlinearities cause wavelength interaction. Moreover, optical interconnects are also dispersive which results in intersymbol interference. Unlike off-chip interconnects, many phenomenon in optical interconnects are highly nonlinear [42]. Nonlinear equalization filters are likely to be necessary for optical interconnects.

- **Nonlinear Optimization**
  The methods developed in this approach are applicable to any system with a linear channel model and objective function. In some applications, linear models might be inappropriate. For example, if a link employs cascaded filters, i.e. receive equalization and pre-equalization at the same time, then the channel is linear in the filter coefficients of either equalizer, but not both. In this case, the problem is quadratic in the filter coefficients. The integrating receivers considered in Chapter 4 are a special case of this. Because we assumed a fixed coefficients for the integrator, we were able to include the integrators in the channel model. To jointly optimize cascaded filters would require optimization techniques more general than the linear programming approaches that we have used in this thesis. This remains a topic for future work.

  One of the most important design factors currently is power. The equalizers are constantly active and consume power during link operations. Low-power circuit design techniques can be used to reduce the amount of power consumption by the equalizers [37]. In this research, I indirectly constrain the peak power of the equalizers by limiting the over-drive ability of the equalizer. Instead of optimizing signal integrity, one can seek to minimize power under some signal integrity constraints. The problem then becomes a quadratic program and is a topic for future work.

In this thesis, I have presented a comprehensive framework for equalization filter synthesis for high-speed digital buses with linear models. I have shown how eye-mask optimization and joint optimization of pre-equalizers, decision-feedback equalizers, and near-end crosstalk cancelling filters can be formulated as linear programming problems. I presented an efficient linear program solver for these filter synthesis problems, and demonstrated the efficacy of our filters using a scale-model test bed based on low-cost, commodity graphics cards. Filters synthesized for l_∞ optimality by our methods consistently outperform their counterparts optimized by the traditional, l_2 methods.

In this final chapter, I briefly examined possible extensions of my approach to other equalization problems including adaptive filter design, bit-error-rate minimization in the presence of random noise, power minimization, and filter design for non-linear (e.g. optical) interconnect. For many of these problems, the models or objective functions are intrinsically non-linear and would therefore require non-linear optimization methods. Because of the thresholding operations that are fundamental in digital links, l_∞ criteria will continue to play an important role and I see promising opportunities to extend the methods developed in this thesis to these broader contexts.

Bibliography

[1] Hardware Level VGA and SVGA Video Programming Information Page. http://web.inter.nl.net/hcc/S.Weijgers/FreeVGA/home.htm.

[2] The XFree86 Project. http://www.xfree86.org.

[3] International technology roadmap for semiconductors. http://public.itrs.net/Files/2003ITRS/Test2003.pdf, 2003.

[4] J. Allard, V. Gouranton, G. Lamarque, E. Melin, and B. Raffin. Softgenlock: Active stereo and genlock for PC cluster. In *Proceedings of the Joint IPT/EGVE'03 Workshop*, Zurich, Switzerland, May 2003.

[5] M. Austin. Decision feedback equalization for digital communication over dispersive channels. *MIT Research Laboratory of Electronics Technical Report 437*, 1967.

[6] Avant! Software. *Star-Hspice Manual, Release 2001.4*, 2001.

[7] M. Azimi, F. Briggs, M. Cekleov, M. Khare, A. Kumar, and L.P. Looi. Scalability port: A coherent interface for shared memory multiprocessors. In *10th Symposium on High Performance Interconnects Hot Interconnects*, 2002.

[8] G. Balamurugan and N. Shanbhag. Modeling and mitigation of jitter in multi-Gbps source-synchronous I/O links. *IEEE Journal of Solid-State Circuits*, pages 2121–2130, 2003.

[9] J.R. Barry, E.A. Lee, and D.G. Messerschmitt. *Digital Communication*. Kluwer Academic Publishers, 2004.

[10] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error - correcting coding and decoding: Turbo-codes. In *Proceedings of IEEE International Communications Conference*, pages 1064–1070, 1993.

[11] J. Chang. How to validate BLVDS SER/DES signal integrity using an eye mask. Technical Report AN200359, National Semiconductor Corporation, March 2002.

[12] A. Charlesworth. The SUN fireplane interconnect. *IEEE Micro*, pages 36–45, 2002.

[13] J.O. Coleman and D.W. Lytle. A linear-programming design language and its application to data-transmission filters. In *Proceedings of IEEE Pacific Rim Conference on Communication, Computers and Signal Processing*, pages 778–782, May 1991.

[14] J.O. Coleman and D.W. Lytle. Linear-programming techniques for control of intersymbol interference with hybrid FIR/analog pulse shaping. In *Proceedings of International Conference on Communications*, pages 329.2.1–329.2.6, 1992.

[15] P.M. Crespo and M.L. Honig. Pole-zero decision feedback equalization with a rapidly converging adaptive IIR algorithm. *IEEE Journal of Selected Areas in Communications*, 9:817–829, 1991.

[16] V. Cuppu and B. Jacob. Concurrency, latency, or system overhead: Which has the largest impact on uniprocessor DRAM-system performance? In *28th International Symposium on Computer Architecture*, 2001.

[17] W.J. Dally and J.W. Poulton. Transmitter equalization for 4-GBPs signaling. *IEEE Micro*, 1:48–56, 1997.

[18] W.J. Dally and J.W. Poulton. *Digital Systems Engineering*. Cambridge University Press, 1998.

[19] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[20] C. Duan, A. Tirumala, and S.P. Khatri. Analysis and avoidance of crosstalk in on-chip buses. In *Proceedings of Hot Interconnects*, pages 133–138, 2001.

[21] A. Fiedler, R. Mactaggart, J. Welch, and S. Krishnan. A 1.0625Gbps transceiver with 2x-oversampling and transmit signal pre-emphasis. In *Proc. of ISSCC97*, pages 238–239, 1997.

[22] G.H. Golub and C.F. Van Loan. *Matrix Computations, 3rd ed.* Johns Hopkins, 1996.

[23] E. Haq, J. Slager, J. Pecoraro, J.D. Johnson, et al. JAZio signal switching technology: a low-cost digital I/O for high-speed applications. *IEEE Micro*, 1:72–81, 2001.

[24] M. Haycock and R. Mooney. A 2.5Gb/s bidirectional signalling technology. In *Hot Interconnects Symposium V*, 1997.

[25] H.V. Henderson and S.R. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23:53–60, 1981.

[26] N.J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.

[27] A. Ho, V. Stojanovic, F. Chen, et al. Common-mode backchannel signaling system for differential high-speed links. Technical report, Rambus, Inc., June 2004.

[28] R. Ho, B. Amrutur, K. Mai, B. Wilburn, T. Mori, and M. Horowitz. Applications of on-chip samplers for test and measurement of integrated circuits. *IEEE Symposium on VLSI Circuits*, pages 138–139, June 1998.

[29] Ron Ho. Personal communication, 2004.

[30] M.L. Honig, P. Crespo, and K. Steiglitz. Suppression of near- and far-end crosstalk by linear pre- and post-filtering. *IEEE Journal of Selected Areas in Communications*, 10:614–629, 1992.

[31] M.L. Honig, K. Steiglitz, and B. Gopinath. Multichannel signal processing for data communications in the presence of crosstalk. *IEEE Transactions on Communications*, 38:551–558, 1990.

[32] M. Horowitz, C. Ken, and S. Sidiropoulos. High speed electrical signalling: Overview and limitations. *IEEE Micro*, 18:12–24, 1998.

[33] Y. Hur, M. Maeng, C. Chun, F. Bien, H. Kim, S. Chandramouli, E. Gebara, and J. Laskar. Equalization and near-end crosstalk (NEXT) noise cancellation for 20-Gb/s 4-PAM backplane serial I/O interconnections. *IEEE Trans. Microwave Theory and Techniques*, pages 246–255, January 2005.

[34] R.K. Cavin III, C.H. Ray, and V.T. Rhyne. The design of optimal convolutional filters via linear programming. *IEEE Transaction on Geoscience and Electronics*, GE-7:142–145, 1969.

[35] Intel Corporation, Santa Clara, CA. *Accelerated Graphics Port Interface Specification, Revision 2.0*, 1998.

[36] L. Jackson. *Digital Filters and Signal Processing*. Kluwer Academic Publishers, 1996.

[37] J. E. Jaussi, G. Balamurugan, D. R. Johnson, B.K. Casper, A. Martin, J.T. Kennedy, N. Shanbhag, and R. Mooney. An 8 Gb/s source-synchronous I/O link with adaptive receiver equalization, offset cancellation and clock deskew. *IEEE Journal Solid-State Circuits*, pages 80–88, January 2005.

[38] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[39] R. Kollipara, G.J. Yeh, et al. Design, modeling and characterization of high speed backplane interconnects. In *High-Performance System Design Conference*, 2003.

[40] B. Landman and R. Russo. On a pin versus block relationship for partitions of logic graphs. *IEEE Transactions on Computers*, C-20:1469–1479, 1971.

[41] Y.C. Lim. Efficient special purpose linear programming for FIR filter design. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-31,4:963–968, 1983.

[42] R. Llorente, R. Clavero, F. Ramos, and J. Marti. Linear and nonlinear crosstalk evaluation in DWDM networks using optical fourier transformers. *EURASIP Journal on Applied Signal Processing*, 10:1593–1602, 2005.

[43] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz. The tiny tera: A packet switch core. *IEEE Micro*, pages 26–33, 1997.

[44] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2:575–601, 1992.

[45] G. Moore. Cramming more components onto integrated circuits. *Electronics*, 38, 1965.

[46] J. Nocedal and S. Wright. *Numerical Optimization*, pages 395–417. Springer Series in Operations Research, Springer Press, 1999.

[47] H. Osaka, T. Komatsu, S. Hatano, and T. Wada. High-speed, high-bandwidth DRAM memory bus with crosstalk transfer logic (XTL) interface. In *19th Symposium on High Performance Interconnects Hot Interconnects*, 2001.

[48] J. Patenaude. High-speed backplanes pose new challenges to comms designers. Technical report, Rambus, 2004.

[49] PCI Special Interest Group, Oregon. *PCI Local Bus Specification, Revision 2.2*, 1998.

[50] PCI Special Interest Group, Oregon. *PCI Express Specification, 1.1*, 2005.

[51] J.W. Poulton, S. Tell, and R. Palmer. Methods and systems for transmitting and receiving differential signals over a plurality of conductors. *Patent No. 6,556,628*, April 2003.

[52] L.R. Rabiner. Linear program design of finite impulse response (FIR) digital filters. *IEEE Transactions on Audio and Electroacoutics*, AU-20,4:280–288, 1972.

[53] RapidIO Trade Association. *RapidIO Interconnect Specification, 1.1*, 2002.

[54] J. Ren. *Equalizing Filter Design for Cross-talk Cancellation*. M.Sc. thesis, University of British Columbia, August 2002.

[55] J. Ren and M. Greenstreet. Synthesizing optimal filters for crosstalk-cancellation for high-speed buses. In *40th DAC*, 2003.

[56] V. Rico-Ramirez and A.W. Westerberg. Interior point methods on the solution of conditional models. Technical report, Carnegie Mellon University, 1997.

[57] H. Samueli. Linear programming design of digital data transmission filters with arbitrary magnitude specifications. In *Proceedings of International Conference on Communications*, pages 30.6.1–30.6.5, June 1988.

[58] C.E. Shannon. *The Mathematical Theory of Information*. Urbana, IL: University of Illinois Press, 1998.

[59] S. Sidiropoulos and M. Horowitz. A 700-Mb/s/pin CMOS signaling interface using current integrating receivers. *IEEE Journal of Solid State Circuits*, 32(5):681–690, May 1997.

[60] Y. Sohn, S. Bae, H. Park, C. Kim, and S. Cho. A 2.2Gbps CMOS look-ahead DFE receiver for multidrop channel with pin-to-pin time skew compensation. In *IEEE Custom Integrated Circuits Conference*, September 2003.

[61] S.R. Sridhara, N.R. Shanbhag, and G. Balamurugan. Joint equalization and coding for on-chip bus communication. In *Proceedings of the Sixth International Symposium on Quality Electronic Design*, pages 642–647, 2005.

[62] V. Stojanovic, G. Ginis, and M.A. Horowitz. Transmit pre-emphasis for high-speed time-division-multiplexed serial-link transceiver. *IEEE Transactions on Communications*, 38:551–558, 2001.

[63] V. Stojanovic, A. Ho, B. Garlepp, F. Chen, and J. Wei. Autonomous dual-mode (PAM2/4) serial link tranceiver with adaptive equalization and data recovery. *IEEE Journal of Solid-State Circuits*, pages 1012–1025, April 2005.

[64] S.K. Tewksbury. *Microelectronic Systems Interconnections*. IEEE Press, 1995.

[65] The Mathworks Inc. http://www.mathworks.com.

[66] The Rambus Inc. http://www.rambus.com.

[67] L.N. Trefethen and III D. Bau. *Numerical Linear Algebra.* SIAM, 1997.

[68] B. Victor and K. Keutzer. Bus encoding to prevent crosstalk delay. In *Proceedings of ICCAD*, pages 57–63, 2001.

[69] U. Wallenhorst. High density interconnection-the way forward. *Report on Components and Manufacturing Business Briefing*, September 2002.

[70] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[71] C.K. Yang, V. Stojanovic, S. Modjtahedi, M.A. Horowitz, and W.F. Ellersick. A serial-link transceiver based on 8-GSamples/s A/D and D/A converters in 0.25-μm CMOS. In *IEEE International Conference on Communications*, pages 1934–1939, 2002.

[72] E. L. Yip. A note on the stability of solving a rank-p modification of a linear system by the sherman-morrison-woodbury formula. *SIAM J. Sci. Statis. Comput.*, 7(3):507–513, 1986.

[73] J. Zerbe, Q. Lin, C. Werner, V. Stojanovic, A. Ho, and R. Kollipara. Comparison of adaptive and non-adaptive equalization techniques in high performance backplanes over temperature, humidity, and impedance variations. In *Design Conference*, Feburary 2005.

[74] J. Zerbe, C. Werner, R. Kollipara, and V. Stojanovic. A flexible serial link for 5-10Gb/s in realistic backplane environments. In *Design Conference*, Feburary 2004.

[75] J. L. Zerbe, R. S. Chau, C. W. Werner, W. F. Stonecypher, H. J. Liaw, G. J. Yeh, T. P. Thrush, S. C. Best, and K. S. Donnelly. A 2Gb/s/pin 4-PAM parallel bus interface with transmit crosstalk cancellation, equalization and integrating receivers. In *IEEE International Solid State Circuits Conference*, pages 430–432, 2001.

[76] J. L. Zerbe, C. Werner, V. Stojanovic, F. Chen, J. Wei, G. Tsang, D. Kim, W. Stonecypher, A. Ho, T. Thrush, R. Kollipara, M. Horowitz, and K. Donnely. Equalization and clock recovery for a 2.5 - 10Gb/s 2-PAM/4-PAM backplane tranceiver cell. *IEEE Journal Solid-State Circuits*, pages 2121–2130, December 2003.

Appendix A

Terminology

A communication **link** typically consists of a **transmitter**, the communication medium (called a **channel**), and a **receiver**. The transmitter takes the digital data and converts it to analog waveforms on the channel. The channel is the communication medium between the transmitter and the receiver, and can have physical realizations such as free space for wireless communication, optic fibres for optical communication and printed circuit board (PCB) traces, coaxial cables or twisted-pair wires for off-chip electrical communication etc. When a channel is implemented using electrical technologies, the implementation is often referred to as **interconnect**. Likewise, an optical implementation of a channel can be referred to as **optical interconnect**. In the backplane environment shown in Figure 1.1, the channel is implemented by on-chip wires, bonding wires and pins of the chip package, backplane connectors, PCB traces and vias etc.

This thesis addresses the synthesis of optimal filters for high-speed buses. A **bus** is an implementation of a channel consisting of multiple wires (and the associated pins, vias, etc.). We typically view the data transmitted across the bus as partitioned among the wires. We refer to each wire or cluster of wires as **line**, and each line conveys a subset of the data transmitted on the channel. For example, a bus may be organized where each line consists of a pair of wires where the voltage (or current) applied to one is the negation of the voltage applied to the other. This is called **differential** signalling because the receiver is sensitive to the *difference* of the signals on the two wires rather than to their absolute values. Another common alternative is **single-ended** signalling where each line is implemented with a single wire, and the receiver detects symbols (e.g. bits) by observing the level of the voltage on that wire.

Appendix B

Glossary

BER	Bit error rate
BGA	Ball grid array
DFE	Decision feedback equalization
DVI	Digital video input
FEXT	Far-end crosstalk
FIR	Finite impulse response
GPIB	IEEE 488 General Purpose Interface Bus
HSPICE	Avant!'s version of SPICE, the industry-standard circuit simulator.
ISI	Intersymbol interference
LP	Linear programming
LSQ	Least-squares
LVDS	Low-voltage differential signalling
l_1,l_2,l_k,l_∞	Let v be a vector of length n, $\| v \|_k$ is defined as $\sqrt[k]{\sum_{i=0}^{n-1} \|v(i)\|^k}$. In particular, $\| v \|_1$ is $\sum_{i=0}^{n-1} \|v(i)\|$; $\| v \|_2$ is $\sqrt{\sum_{i=0}^{n-1} (v(i))^2}$; $\| v \|_\infty$ is $\max\{\|v(0)\|, \|v(1)\|, \ldots, \|v(n-1)\|\}$.
MIMO	Multi-input multi-output
MMSE	Minimum mean square error
NE	Near-end equalizer
NEXT	Near-end crosstalk
PAM	Pulse amplitude modulation
PCB	Printed circuit board
PE	Pre-equalizer
PLL	Phase-lock loop
TDR	Time-domain reflectometer
VGA	Video graphics array
VNA	Vector network analyzer

Appendix C

List of Variables

We summarize the list of variables that we use throughout the thesis. We omit the variables that are only used locally.

B	$\mathsf{B} \in \mathbb{R}^{q_{\text{bus}} w_{\text{bus}} \times w_{\text{bus}}}$ is the bus impulse response.	(p. 33)
$d_{\Sigma}(i,s)$	the worst-case disturbance on wire i at sample time s.	(p. 37)
$d_{\text{fe}}(i,s)$	the worst-case far-end disturbance on wire i at sample time s.	(p. 49)
$d_{\text{ne}}(i,s)$	The worst-case near-end disturbance on wire i at sample time s.	(p. 49)
extend_0	extend_0 is a linear operator. for $v \in \mathbb{R}^{n_1}$ and $n \geq n_1$, $\mathsf{extend}_0(n_1, n)v$ pads v with zeros to produce a vector of size n	(p. 29)
F	$\mathsf{F} \in \mathsf{R}^{m_{\text{fir}} w_{\text{bus}} \times w_{\text{bus}}}$ is the filter impulse response, i.e. the filter coefficient matrix.	(p. 33)
f	the filter coefficient vector.	(p. 34)
G	G is a function of the bus impulse response and the input. $\mathsf{G} \cdot f$ gives the response of the channel consisting of the bus and filter f for a given input sequence.	(p. 35)
G_d	the set of $g(i,j,s)^T$ vectors that do not belong to G_u. $G_d f$ computes the disturbances.	(p. 37)
G_u	the set of $g(i,j,s)^T$ vectors with $i = j$ and $\delta_0 \leq s < \delta_0 + r_{\text{bit}}$. $G_u f$ computes the undisturbed responses.	(p. 37)

$g(i,j,s)$	A vector such that $g(i,j,s)^T f$ is the response on wire j at time s to a single bit input on wire i given filter f.	(p. 36)
In	$\mathsf{In} \in \mathbb{R}^{q_{\mathsf{In}} w_{\mathsf{bus}}}$ is the input vector to the channel.	(p. 33)
k_{block}	The size of the diagonal blocks in the normal equations, $k_{\mathsf{block}} = k_{\mathsf{disturb}}/(k_{\mathsf{mask}} w_{\mathsf{bus}})$.	(p. 62)
k_{disturb}	Number of disturbances, roughly $w_{\mathsf{bus}}^2 k_{\mathsf{mask}}(n_{\mathsf{fir}} + n_{\mathsf{bus}})$.	(p. 61)
k_{fir}	Total number of filter coefficients, roughly $w_{\mathsf{bus}} m_{\mathsf{fir}}(2w_{\mathsf{fir}} + 1)$.	(p. 61)
k_{mask}	Number of measurement points defined in the eye mask.	(p. 61)
k_{maxf}	The number of variables introduced to calculate the maximum filter output, roughly $w_{\mathsf{bus}}(2w_{\mathsf{fir}}+1)$.	(p. 63)
L	Lower bound constraints for eye masks. L consists of a set of (s, η_s) pairs such that the maximum undershoot at sample time s is η_s.	(p. 26)
$\mathbb{M}_{\mathsf{bit}}$	The set of integers that are multiples of r_{bit}.	(p. 32)
$\mathbb{M}_{\mathsf{sym}}$	The set of integers that are multiples of r_{sym}.	(p. 47)
m_{fir}	The length of the filter in tap times.	(p. 32)
n_{bus}	The length of the bus impulse response in bit times.	(p. 32)
n_{fir}	The length of the filter in bit times.	(p. 32)
n_{In}	The length of the input sequence in bit times.	(p. 32)
Out	The output vector.	(p. 34)
q_{bus}	The length of the bus impulse response in sample times.	(p. 33)
q_{fir}	The length of the filter in sample times.	(p. 32)

q_{In}	The length of the input sequence in sample times.	(p. 32)
r_{tap}	The number of sample times per tap.	(p. 32)
r_{bit}	The number of sample times per bit.	(p. 32)
r_{sym}	The number of sample times per symbol.	(p. 47)
U	Upper bound constraints for eye masks. U consists of a set of (s, α_s) pairs such that the maximum overshoot at sample time s is η_s.	(p. 26)
$u(i, s)$	The undisturbed output on wire i at time s given a single bit input on wire i at time 0.	(p. 36)
w_{bus}	Width of the bus.	(p. 33)
w_{fir}	Width of the filter, defined as the number of neighbors considered in each side of the input wire. The number of inputs to the filter for any given wire is $2w_{\text{fir}} + 1$.	(p. 31)
w_{link}	Number of links in the bus.	(p. 47)
α	A vector that specifies the shape of a parameterized eye mask. α_s means the maximum overshoot/undershoot at sample time s is $\alpha_s \eta$.	(p. 38)
δ_0	The estimated delay of the bus.	(p. 32)
η	The eye-mask optimization parameter, together with α to specify the undershoot and overshoot constraints of a parameterized eye mask.	(p. 24)