Bridging the gap: An Optimization-based Framework for Fast, Simultaneous Circuit & System Design Space Exploration

by

Ranko Sredojević

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2008

Author ....
Department of Electrical Engineering and Computer Science
January 31, 2008

Certified by..... ....
Vladimir Stojanović
Assistant Professor
Thesis Supervisor

Accepted by..... ....
Terry P. Orlando
Professor
Chairman, Department Committee on Graduate Students

Bridging the gap: An Optimization-based Framework for Fast, Simultaneous Circuit & System Design Space Exploration

by

Ranko Sredojević

Submitted to the Department of Electrical Engineering and Computer Science
on January 31, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Design of modern mixed signal integrated circuits is becoming increasingly difficult.

Continued MOSFET scaling is approaching the global power dissipation limits while increasing transistor variability, thus requiring careful allocation of power and area resources to achieve increasingly more aggressive performance specifications.

In this tightly constrained environment traditional iterative system-to-circuit redesign loop, is becoming inefficient. With complex system architectures and circuit specifications approaching technological limits of the process employed, the designers have less room to margin for the overhead of strict system and circuit design interdependencies. Severely constrained modern mixed IC design can take many iterations to converge in such a design flow. This is an expensive and time consuming process.

The situation is particularly acute in high-speed links. As an important building block of many systems (high speed I/O, on-chip communication, ...) power efficiency and area footprint are of utmost importance. Design of these systems is challenging in both system and circuit domain. On one hand system architectures are becoming increasingly complex to provide necessary performance increase. On the other, circuit implementation of these increasingly complicated systems is difficult to achieve under tight power and area budget.

To bridge this gap between system and circuit design, we formulate a circuit-to-system optimization-driven framework. It is an equation-based description, powered by a human designer. Provided with equation-based model we use fast optimization tools to quickly scout the available design space. Presence of a designer in the flow is invaluable resource enabling significant saving by simplifying the models to capture only the relevant information and constraining the search space to areas where meaningful solutions might be expected to be found. Thus, the computational effort overhead that plagues the simulation-based design space exploration and design optimization is greatly reduced.

The flow is powered by a signomial optimization engine. The key challenge is to bring, from the modeling point of view, very different problems such as circuit design and system design into the realm of an optimization engine that can solve them jointly, thus breaking the re-design loop or at least cutting it shorter.

Relying on signomial programming is necessary in order to accurately model all the necessary phenomenons that arise in electrical circuits and at system level. For example, defining regions of operation of transistors under polarization conditions can not be modeled accurately with simpler type of equations. Similarly, calculating the effect of filtering to a signal also requires possibility to handle signomial equations. Thus, signomial programming is necessary yet not fully explored and finding suitable formulation might take some experimenting as we will see in this thesis.

Signomial programming, as a general non-convex optimization problem, is still an active research area. Most of the solutions proposed so far involve local convexification of the problem in addition to branch & bound type of search. Furthermore, most of the non-convex problems are solved for one particular system of equations, and general methodology that is reliable and efficient is not known. Thus, a big part the work to be presented in this thesis is detailing how to construct a system formulation that the optimization engine can solve efficiently and reliably. We tested different formulations and their performance measured in terms of parsing and solving speed and accuracy. From these tests we motivate and explain how a series of transformations we introduce improve our formulation and arrive to a well-behaved and reliable form.

We show how to apply our design flow in high-speed link design. By restructuring the traditional design flow we derive system and circuit abstractions. These subproblems are interfaced through a set of well defined interface variables, which enables code level separation of problem descriptions, thus building a modular and easy to read and maintain system and circuit model.

Finally we develop a set of scripts to automate formulating parametrized system level description. We explain how our transformations influence the speed of this process as well as the size of the model produced.

Thesis Supervisor: Vladimir Stojanović
Title: Assistant Professor

Acknowledgments

I would like to dedicate this thesis to everyone who passed some knowledge onto me, to all my teachers.

This work is my accomplishment as much as it is of those who supported me on my way towards it; of those who gave me the knowledge to work on it and of all who believed in me and trusted me with this task. I have been very fortunate to learn from many different people and as it puts me in debt to them I apologize if I had forgotten someone. I will mention some of them here, in the same order as they appeared in my life.

In the first place, I want to thank my family; my mother Stanica and my father Radovin who tried to teach me how to work and how to be fair and responsible and to my sister Vesna with whose help I learned how to share and love. You have given me everything, even when I would have not deserved it, and I hope you think that I was worth it.

I would like to thank to all my friends, some of which I have known for as long as I can remember, and especially two people, Marko and Miloŝ, who share most of my ideas about the world, yet they never hesitated to correct me if I were wrong. You always offered an alternative view, enough to make me find my way when I would get lost.

Big thanks go to all of my professors in Mathematical High-School in Belgrade, Serbia. Above all to my math and physics teachers, professors Dragović, Ĉukić and Milić who gave me my first lectures in science, the ones I still remember today. You were wonderful examples of what a teacher should be.

A special thank you for my professors Aleksandra "Mom" Pavasović, Jelena Popović and Slavoljub Marjanović who went out of their way to guide me and who, at times, had better vision of my path than even myself. I can only hope to be so unintrusively helpful to someone as they were to me. I had to earn this honor, though, in a wast sea of students admitted in my class. I was lucky and stubborn enough to make it through one of the toughest school 'drills' around: Department of Electronics of the Faculty of Electrical Engineering in Belgrade, Serbia. Almost three years after my graduation, I am still discovering how well they laid out fundamental concepts of science and engineering for us, and I am still to find all the doors they left unlocked.

A hearty hug to my 'second family' here in New England: Mira, Ranko, Miloŝ and Marko Stevanović. I will not forget the lessons in kindness that you are real example of.

My gratitude for improving my English from 'somewhat understandable' to 'fairly fluent', for reminding me that there are other things apart from my work, and for teaching me that loneliness is not the only path to achieving my goals, goes to miss Alexandra Muse Fallows.

I am very grateful to my research group at MIT. It has been good two and a half years, and I feel I improved in every respect; be it in understanding circuit design better with Byungsub Kim and Fred Chen or learning about communication theory concepts from Natasha Blitvic and Sanquan Song.

I was, first, introduced to optimization-based circuit design through works of Mar Hershenson, Sunderarjan Mohan and Dave Colleran, of Sabio Labs. This work would not be possible if it were not for them to bring me up to speed with optimization based circuit framework. I would also like to acknowledge Shyne Tseng and Almir Mutapcić for their support and guidance with optimization tools. This work was greatly advanced by solver tools obtained from Sabio Labs and Mosek.

Finally, I want to thank my research adviser, professor Vladimir Stojanović, who trusted me with this project. While learning about high-speed communication systems and optimization was always happening with him, I think I started learning something more important: patience and systematic approach to problems, which is important as stabilization objective model to my I-want-it-all-now-chaotic type of thinking. His advising style which would, in my opinion, best be described as a sliding-mode control gave me, at the same time, direction and some freedom. I know I have not been the easiest plant (student?) to control (deal with?), but I feel we converged to the same manifold (page?) in finite time (one more proof that it is, indeed, sliding mode control), and this work and this thesis are the results.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we discuss the need for a new, more coherent approach to design of mixed-signal and analog integrated systems, focusing in particular on a high-speed link example. We discuss similar attempts or significant previous work and point out the technologies and conclusions we rely on. Further on, we introduce an example: a high speed link design. After describing performance measures for our example design, we focus on the difficulties of the traditional design flow. Finally, we introduce the formal description and setup of our example.

1.1 Motivation

Design of modern mixed signal integrated circuits is becoming increasingly difficult, mainly due to transistor scaling and always increasingly challenging performance specifications. Continued MOSFET scaling is approaching the global power dissipation limits while increasing transistor variability, thus requiring careful allocation of power and area resources to achieve increasingly more aggressive performance specifications. Designers manage to overcome these challenges by modifying topology and better budgeting at the system level. However, the design cycle is becoming longer and more expensive [1]. One possible aid to this problem is creating a design flow that would provide tighter interaction between design hierarchy levels thus enabling faster design prototyping and design space exploration.

Currently, the mixed signal design is done in an iterative loop where system and circuit designers take turns, redesigning respective levels of design hierarchy. Usually, one of the biggest challenges is connecting the system performance and resource allocation (e.g. power/area) estimate with subsystem performance, while providing realistic estimate of achievable designs at the circuit level. Usually, the system level designers start by assuming certain performance at the physical level. With these in mind, they design system level and define the necessary sub-system performance. After this system level budgeting is done circuit designers try to implement it. This design attempt will provide system designers with new information about achievable sub-system performance, and the iteration is repeated. Such iteration is performed until the assumptions taken at the system level happen to match the achieved performance at the circuit level; at this point the design loop converged.

With modern technologies it is not easy to predict performance before the actual design phase. The difficulty is due to smaller slack on specifications designers have, more complicated transistor modeling and harder trade-offs as the technology scales and device variability increases. Thus, even relatively small changes in specifications might prove to be challenging, requiring significant effort from the circuit designer, or even prove to be infeasible. Such sensitive design environment usually leads to a bad first estimate, and the design loop we described takes several iterations to converge. With increased design time costs this is becoming expensive and one of the main drawbacks to efficiently designing complex systems [1]. Furthermore, as average system complexity and size increase, power efficiency and area are becoming as important as meeting the performance specifications.

Our aim in this thesis is to develop a design flow that brings system and circuit models jointly to a solution. We rely on system and circuit designers to provide reasonably accurate models of respective levels of design abstraction. Provided with such models, in a certain form, we plan to use optimization engine to explore the joint design space outlined by them. This approach, the system and circuit models are solved 'aware' of each other, and all inter-dependencies are guaranteed to be satisfied.

A good example to demonstrate this design methodology is a high-speed link (HSL). Basically, any data communication path in modern microprocessors, network communication equipment, etc. can be considered a HSL, [26, 18, 42]. Due to wire bandwidth limitations, HSLs are becoming more complicated systems, while on the other hand being tightly constrained by power, area and speed requirements due to high density of integration. These are very challenging and opposing system requirements and interconnects still draw significant portion of power in modern designs [27]. Initial system design-redesign phase can take many turns before an acceptable design is achieved, and this process can include even some unsuccessful prototypes, thus increasing design time and cost significantly.

1.2 Previous work and important technologies

Developing a design flow assumes developing underlying CAD infrastructure as well as demonstrating actual circuit and system models. With this in mind, the previous work can be divided into two main areas: 1. *optimization-aware modeling and optimization methods*, and 2. *circuit and system design and modeling.*

To develop an optimization-driven framework we rely on many relatively recent results in optimization technology, optimization driven circuit design and appropriate tools. The optimization technology improved significantly in last couple of decades by introduction of interior point methods for convex mid-sized optimization problems. On this theoretical background most of the new solver infrastructure was implemented, resulting in high quality solvers for certain types of optimization problems such as the geometric programming, generalized geometric programming, linear programming, etc. The main challenge from this perspective is translating our system and circuit models into appropriate form that can be efficiently solved with a standard optimization tool. For a good overview of the optimization in practical settings the reader is referred to [3].

This recent progress in the field of mathematical programming prompted a lot of work in attempts to apply these new achievements in engineering. The success rate varies, and there are, still, many types of problems that appear in engineering applications where optimization theory still cannot give satisfactory solutions in an efficient manner.

As always, with no exact solution known, we see a lot of work in heuristic convex relaxation or brute force methods. Some work that is very relevant for the material laid out here can be found in [9, 23, 22, 12, 14, 13, 11]. In these reports the authors introduce assumption that transistor model can be represented in simple form through 'monomial' functions (i.e. generalizations of the power-function in multiple dimensions $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$), and that performance measures of circuit blocks can be expressed in the generalized geometric (GP) programming form [3] starting from the transistor model parameters.

One should keep in mind that these works assume process technology nodes that were significantly easier to model than the current sub-100nm process nodes (the quadratic-law transistor model was reasonably good approximation of the measured results). Also, these works were circuit oriented without any treatment of the system level design and modeling. Furthermore, polarization of the circuits was usually omitted from the model and left to be checked once the solution is obtained. The significant contribution of these works is that they laid out the basis for the use of convex programming in circuit design [23, 28, 9, 22] and with further refinement (in terms of more accurate signomial-based models and appropriate solver extensions to handle signomial formulations like branch&bound guided convexification and GP approximation) represent the basis for our optimization-based approach. Out of these works, the GP-like optimization engine used in this work was derived.

1.2.1 High-speed link example

Serial links feature fairly complex system topologies. Usually multiple bit streams are serialized by a multiplexer, after which they are passed through pre-driver inverter chain that serves as a link between digital logic (usually utilizing minimum size transistors) and analog transmit equalization filter that also serves as the output/channel driver. The clocking of the transmitter is derived from a phase-locked loop (PLL). After the channel, the receiver signal path starts with an analog peaking amplifier/equalizer, followed by the clock-and-data recovery (CDR) loop. Finally, the de-serialization is performed, thus recovering the original bit streams.

As links do not perform any useful computation or processing but directly influence the performance of the system as the interconnect infrastructure, it is quite natural to desire highest performance for a low price in terms of area and power consumption. One interesting problem with high speed links is strong coupling between sub-systems. This is quite different from mainstream digital design where many blocks can be treated separately. This is not possible in the HSL design. As an example we can see the interaction of transmitter and receiver equalizers: they both perform almost the same task. The question that arises immediately is: which one is better? Do we need both? How should the link resources be allocated? Due to this complicated interactions of the building blocks performances designers still do not have final answers or methods to answer these types of questions in full and do it efficiently. In such conditions designers try to simplify situation by introducing certain assumptions [21]. At times, the meaning and justification for such assumptions is not clearly expressed and it can be challenging to interpret the resulting trade-offs and impact of the assumptions we introduced. Furthermore, in the traditional design flow it is hard to formally write many such constraints and get insight into system/design sensitivity to them. Our goal in this work is to provide a more structured approach to express design constraints at the system and at the circuit level, and to do so in an unified manner.

Most generally, system level designers can apply any of the techniques developed in communication theory to high speed links. As always, there are certain aspects of the high-speed link design that are very specific and we should consider them when making initial assumptions and deciding on topology and system level functionality. As an example, we can notice that designers of HSLs seldom make use of coding techniques. While this is unusual, at first, as coding can increase the effective data rate helping to achieve the channel capacity, this choice makes sense once we take into account the complexity of decoder implementations, throughput requirements in case of a HSL and the power overhead at such high data rates, [42].

In general, a good overall treatment of the high-speed link specific design and modeling issues can be found in [43]. It covers both system and circuit design, and describes the high-speed link environment and general setup. For this thesis the most important conclusions from [42] would be ones that have to do with system level design (equalization especially), the link environment, and link performance analysis to some extent. A possible extension of the work to be presented here might be incorporating timing jitter model into developed flow; for which a model is proposed in [43].

A simulation-based tool with a similar purpose to the one we are developing was described in [5], along with some modeling insights. There have been a couple of interesting simulation-based circuit optimization attempts, such as [31]. However, simulation based approach requires immense computing power in and does not scale beyond circuits with tens of transistors and simple simulation requirements [31].

1.2.2 Contributions of this thesis

To the best of our knowledge, major and original contributions of this work are

1. The first attempt to show an unified and highly structured flow for unified circuit and system design of HSLs

2. Robust system level formulation utilizing MOR-like techniques and transformations to enable efficient parsing and solving

3. HSL design space exploration and tradeoffs

    - Power consumption versus sampling phase choice
    - Power allocation between predriver, transmit and receiver equalizer
    - Interaction between transmit side and receive side equalization efforts

1.3 High-speed link design

A system level model of a high-speed link is shown in Fig. 1-1. The transmit and linear receive filters are needed as equalizers in order to compensate for the frequency-selective (predominantly low-pass) channel characteristic. The effect of PLL and CDR jitter can be included through effective voltage noise models [43].

We aim to offer an answer to very important questions when new high-speed link design is being conceived:

- how limited resources (power, area/complexity) should be allocated in the design
- what are the operating conditions of the circuits in the filter chain
- what is a reasonable (initial) design for these circuits

Our main goal is to answer these questions in relation to the ultimate performance measure for a high-speed link: the bit error rate (BER). In the light of our previous discussion our final goal is to try developing a flow that can help designing high-speed link with a certain performance (BER) for a given data rate, with as low as possible power consumption and bounded area. The design variables we are considering are: transistor sizes, biasing conditions such as polarization currents and voltage references, tap coefficients in digital equalizers and pole-zero placement of analog peaking

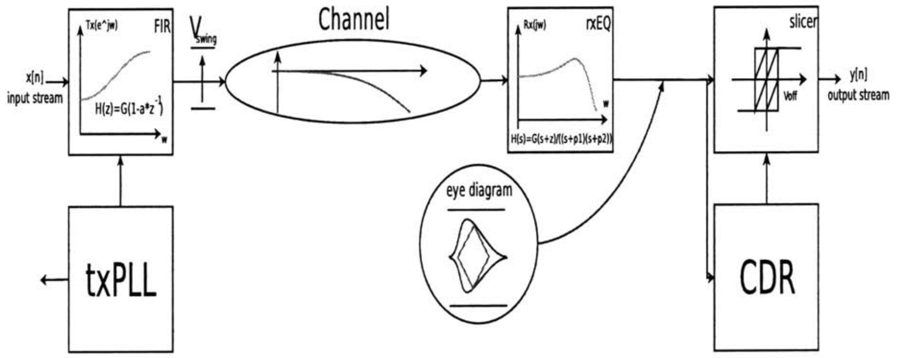


Figure 1-1: System-level view of a high-speed link

amplifiers. Thus, quasi formally we can express our design goals as the following optimization problem:

$$\begin{array}{lc}
\min & Power \\
st. & estimate(BER) < BER_{desired} \\
 & estimate(Area) < Area_{desired} \\
 & circuit\ biasing\ constraints \\
 & variables: \ Ws, \ Ls, \ currents, \ equalization\ coefficients, \ ...
\end{array} \tag{1.1}$$

For simplicity, we focus on the signal processing chain in a link. We do not model any jitter or timing inaccuracies effects introduced in the circuits. To the first order this is justified as the residual ISI is the dominant error mechanism producing order of magnitude more errors than jitter-induced errors [42].

1.4 Summary

In this chapter we introduced the necessity of jointly solving circuit ans system levels for tightly constrained designs. We introduced the general idea we will use to approach the problem of system and circuit joint co-design, and introduced the most relevant past work that we base our method on. Finally we defined a concrete system we will use as a vehicle to demonstrate the performance of the method.

Building on the previous work in the field of equation-based circuit optimization we devised a set of MOR-like transformations that enable compact and robust formulation of the system level model of a HSL. Using this formulation enables us to formulate circuit and system (joint) optimization problem that is tractable and accurate.

Chapter 2

System level design

This chapter is dedicated to the system level model and design formulation. We will see how we can estimate the BER of the system, and what are the assumptions that need to be taken in order to do so. Furthermore, we will explain the subsystem integration and interface variables we pass between layers of the system model abstraction.

From the formulation point of view we will see how certain problem-specific observations can help us reduce the problem size. This will have big impact on our ability to formulate the optimization and use this flow.

2.1 System description

Here we explain the purpose of each sub-block of the system level and their mutual interaction at the top level of the design.

A high-speed link model we will be referring to in this chapter is shown in Fig. 1-1. We will, to the first order, distinguish between two main sub-systems:

- the signal processing chain and
- the timing/clocking subsystem.

In the simplest model, these two subsystems operate in different domains: signal processing is mostly observed through operations in voltage/current domain; the clocking is an auxiliary subsystem needed for synchronization and could be modeled as an superimposed imperfect timing in the ideally synchronous core link. Alternatively, it can be modeled as some perturbation of the ideal signal processing result [41].

2.1.1 The signal processing chain

The core of the link, in our case, is the signal processing chain consisting of

- transmitter predriver
- transmit equalization FIR filter
- the channel
- linear receive equalizer (peaking amplifier)
- slicer/latch

Before we proceed we should define a couple of terms to be used.

Symbol response (SR) can be defined for linear systems. It is, basically, the convolution of the impulse response of the channel and the symbol (basis function) used for communication. [1] If bit-space sampling is used it is the representation of the channel response at a certain *sampling phase.*

Sampling phase is the relative position within the symbol interval where receiver decides on transmitted symbol. In the HSL the clock and data recovery (CDR) subsystem determines the exact location of the sampling phase.

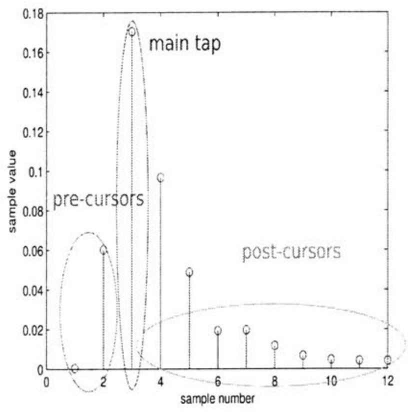
Main tap is the largest sample in the bit-space sampled SR, Fig. 2-1(a), of the system. This is the relative position where we expect forming of the appropriate sampling value. [2]

[1]It has been observed that SR cannot be defined for certain single-ended systems as their falling and rising edges are not symmetric, thus, in binary communication, symbols 0 and 1 cannot be expressed using the same waveform. However, most of modern serial links operate in differential mode and this issue does not exist.

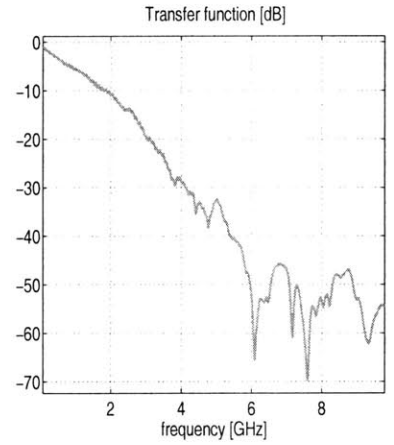
[2]The position of the main tap is always found from the SR of the system. While this is influenced by the SR of the channel, this position can and will be changed due to effects of equalization, as we can see in Fig. 2-4(a) and Fig. 2-4(b). We will discuss this in much more depth in the section on equalization.

The channel

We refer to any type of medium used for transmission of data as the channel. In our example, it is some type of electrical interconnect (i.e. off-chip PCB traces, on-chip global routing layers, ...). We show a typical impulse response in Fig. 2-1(a) and the corresponding transfer function in Fig. 2-1(b). In case of oversampling the SR is convolution of the impulse response and symbol waveform, as we have mentioned.



(a) Typical SR of a channel



(b) Typical transfer function magnitude

Figure 2-1: A 32 inch off-chip interconnect example

In this work we model the channel as an FIR filter. We chose this representation because it is easy to obtain from S-parameter measurements. It is also very convenient for automation as it is straightforward to switch between time and $\mathcal{Z}$-domain in this form. Finally, any system can be represented in this form, to any desired accuracy.

From Fig. 2-1(b) we see that for a typical interconnect channel the $-3dB$ bandwidth and the Nyquist frequency of signaling for gigabit rates are widely separated. On average, in high speed links, channel attenuation at the Nyquist frequency seems to be between $15dB$ and $30dB$ [41] which means that $-3dB$ bandwidth can be at a frequency that is more than one decade lower than Nyquist frequency. Under these circumstances it is difficult or impossible to recognize incoming bit stream at the receiver side and we are forced to use equalization in order to reconstruct the information [38, 15, 35, 41].

Equalization: bandwidth extension model

Traditionally, in communications we treat equalization as an aid in achieving less intersymbol interference by extending the bandwidth of the system [43]. The intuition behind this approach is drawn from the frequency domain analysis of linear systems: extending bandwidth enables more of the significant components of the transmitted waveform to be passed through the system, thus providing higher fidelity when comparing input and output waveforms.

Note that this technique applies equally to analog and digital communication systems. This is due to the fact that with this approach we set our goal to be preservation of the transmitted signal waveform despite the channel filtering. This approach can be formalized as Least Mean Square (LMS) parameter estimation problem [6, 39].

As HSL channels are predominantly low-pass, this bandwidth extension is achieved by introducing the high-pass filters at the transmitter or receiver side. The basic idea behind this equalization scheme is presented in Fig. 2-2. Ideally, we would invert the degradation introduced by the channel in full which is known as zero-forcing equalization (ZFE) [6], due to the fact that it removes all the ISI. The ZFE attempts to make the system response perfectly flat in the range of interest, as we explain in Fig. 2-2. This means that significant peaking might be necessary in order to counteract the attenuation at high frequencies, Fig. 2-1(b). In Fig. 2-2 we see the channel characteristic in blue. The digital transmit-side equalizer transfer function is given in red up to the Nyquist frequency, and in dashed-red above for another Nyquist range. The resulting transfer function is given in green. The resulting transfer function (the green curve) should be an approximation of the flat all-pass filter in the range of the significant spectral components of the transmitted waveform. However, this is not always possible [6].

In the presence of noise, inverting the channel would amplifying the noise at high frequencies significantly, and potentially reduce the signal-to-noise ratio (SNR) [43]. Under these circumstances we employ the Unconstrained Minimum Mean Square Error (U-MMSE) [3] [39, 24, 41]. We note that one significant difference between ZFE

[3]This algorithm is just another name for Least Mean Square fitting procedure in different context.

and U-MMSE is the objective function we are optimizing for. For ZFE algorithm the objective is minimization of the ISI. On the other hand, in U-MMSE the objective is constructing equalizer that minimizes energy of errors (in a certain period of time) in presence of noise. In order to make this model more realistic, additional constraint limiting the maximum output voltage from the transmitter (a constraint that is present in every circuit topology) can be introduced. Such optimization problems are sometimes called Constrained MMSE (C-MMSE) [8].

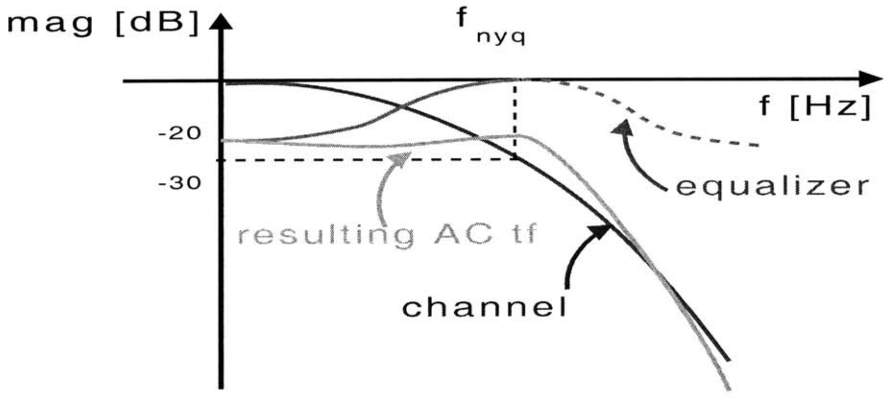


Figure 2-2: Principal idea of equalization

Finally, since in HSL environment the ISI is the dominant error generating mechanism [42], our objective can be reduced to the maximization of the eye opening. In such circumstances model becomes a Linear Programming (LP) problem [37]. This brings us to a different interpretation of the role of an equalizer in the digital communication system.

Equalization: symbol classification model

A different approach to equalization can be adopted from machine learning, machine vision and pattern classification. In this framework we construct equalizer to be linear separator of symbols in the received constellation [6, 16, 10]. The main idea of this approach can be seen in Fig. 2-3, for the case of binary signaling.

Here we can see two different sets of received symbols. In binary communication framework we can interpret each set as possible coordinates in equalizer state space

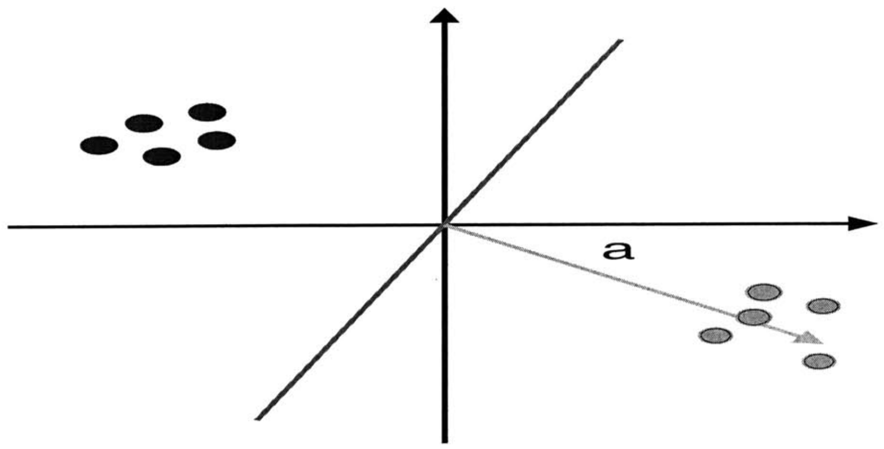


Figure 2-3: Separation of two non-intersecting sets

for different symbols being transmitted. Existence of multiple points in each set is clearly due to intersymbol interference as received values will depend on the particular sequence of symbols surrounding the symbol we try to decide on. Similar analysis is presented in [6].

In Fig. 2-3 vector **a** is, actually, vector of equalizer coefficients. We can observe that this vector has positive inner product with all the points in one of the sets, and negative inner product with points in the other set. This is equivalent to the existence of the separation hyperplane presented in the picture in red color. More on separation of sets can be found in [32, 29].

We can note that this approach does not, necessarily, try to match waveforms at the input and output of the system. Thus, this method is not suitable for analog communication systems. In these settings we are exploiting the fact that we are communicating in digital domain and the only function of equalization is to provide easy detection of the symbol which is easily associated with separation of sets. This is, exactly, the difference between the objective functions used to derive ZFE and MMSE versus eye maximizing LP program, as we have discussed in the previous section.

Transmit equalization FIR filter

This basically analog subcircuit has two main purposes:

- power amplification and impedance matching with the channel
- filtering to ameliorate severe intersymbol interference (ISI) generated in the channel

This analog front-end serves as a gateway from digital-logic's discrete time domain where we operate in terms of symbol-streams into analog domain where appropriate analog continuous-time representation of the bit sequence is transmitted over a lossy channel. It can be modeled as a very simple D/A converter.

The filtering operation is performed on a per-symbol basis. Thus, this module is a digital filter and has, to the first order, periodic magnitude of the frequency response. Since it is per-symbol filtering, the Nyquist frequencies of signaling and the Nyquist frequency of this filter côincide. Usually, the transfer function of such a filter is written as

$$txF(z) = \sum_{k=0}^{N} a_k z^{-k} \tag{2.1}$$

assuming that we only consider sampling times that are T_{bit} apart. Here we will introduce a slightly more general notation. We will assume that we have *oversampled* this system with oversampling ratio *ovsRate*. Under these circumstances, the transfer function of this filter will be

$$txF(z) = \sum_{j=0}^{N*ovsRate} a_{mod(j,\ ovsRate)}\ z^{-j} \tag{2.2}$$

where *mod*(*a*, *b*) is standard notation for reminder of integer division. The reason for this 'complication' will become clear when we talk about CDR and sampling phase, as well as when we consider problems rising in debugging and the utility of an eye diagram.

As we can see in Fig. 2-2, high-pass transfer function in transmit equalization filter is achieved by attenuating the signal at low frequencies. The source of this attenuation can be traced to the circuit implementation of the filter. As it has to be a feasible circuit, we always have some maximum output signal swing that we can achieve. For example, we can rarely achieve more than supply voltage, unless special techniques are used. More often some other constraints would limit achievable output swing. What this means at the system level is that for all the possible bit sequences the output from the transmit equalization FIR filter has to be limited. We can safely assume this limit to be unity, and scale appropriately. This means that

$$\sum_{k=1}^{N} |a_k| \leq 1 \tag{2.3}$$

This equation is usually referred to as *the peak power constraint* or *the peak swing constraint*. Usually, some of the equalizer taps are negative, so DC gain given by

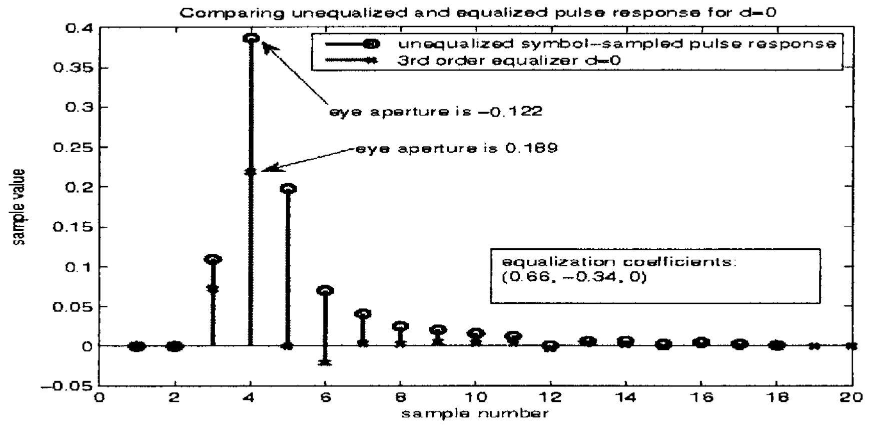
$$\sum_{k=1}^{N} a_k = txF(z = 0) \tag{2.4}$$

is usually less then 1. In order to achieve some peaking and extend the bandwidth of the system through transmit side equalization we have to sacrifice the gain of the system. This is a very important property that brings up many questions about efficiency of transmit side equalization and trade-offs with receiver side equalization.

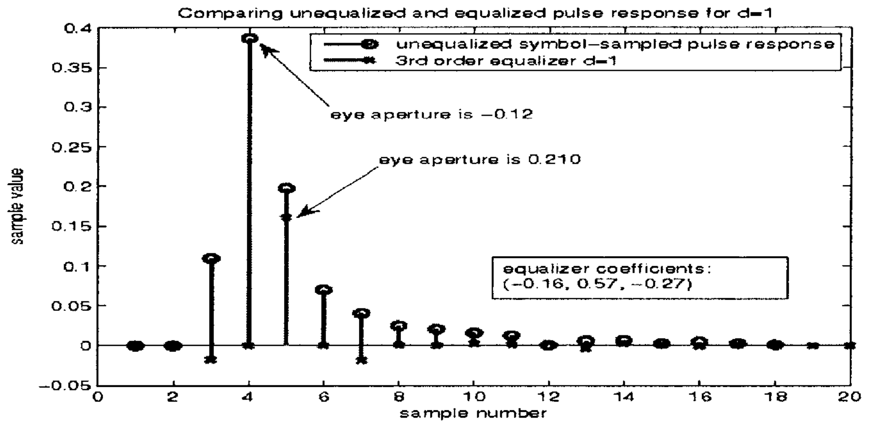
Another important aspect of the equalizer is the delay it introduces in the signal chain. In Fig. 2-4(a) and Fig. 2-4(b) we can see two examples of equalized single bit response (SR) for two transmit equalizations with the delays of $d = 0$ and $d = 1$, respectively. It is obvious that by allowing for longer delays we potentially perform a better equalization as we can take into account not only the post-cursors in the SR, but some of the pre-cursors as well, Fig. 2-1(a). The tradeoff is observed in the fact that filters with a delay move relative position of the main sample in the SR vector.

We should note that increasing delay in order to perform a better equalization is acceptable in systems which can tolerate delay but need as much bandwidth as possible. However, in systems which have very short delay and require small flight time (possibly some on-chip links), such approach might be undesirable.

As this work is a proof of concept we will mainly consider $d = 0$ transmit equalization settings, as it is easier to perform debugging and think about signals in this way. It should be straightforward to extend this to arbitrary delay by removing some



(a) Second order FIR filter with delay d = 0



(b) Second order FIR filter with delay d = 1

Figure 2-4: Delay of the transmit equalizer

of the constraints.

The optimal settings that maximize the worst case eye opening for transmit side equalization filter are known to be solution of a linear programming problem (LP), [37]. Furthermore, the peak swing constraint is straightforward to include. As the LP solver technology is well developed, it is easy to determine these optimal settings for a given channel and equalizer delay. We will talk more about worst case eye opening later in the text.

To combat reflections in the channel, the transmit equalizer output impedance is usually matched to the characteristic impedance of the channel (typically 50Ohm).

As this block also provides power amplification, the output transistors tend to be relatively large in size, and it is necessary to take into account their parasitic capacitance. Otherwise it can severely degrade the performance of the system due to impedance mismatch, which causes channel degradation through reflections.

In case of an off-chip link, maybe even more serious source of parasitic capacitance are the ESD (Electrostatic Discharge) structure and pad capacitance at the output pad. This is introduced into the formulation as some fixed capacitance at the output. To take it into account we can follow the usual design procedure and impose a constraint that pole resulting from the output impedance of the equalizer has to be significantly higher in frequency than Nyquist frequency of communication. If it turns out to be a major drawback to the design, it would make sense to model it more accurately.

Linear receive equalizer

This is an analog front end in the receiver. It has almost the same functions as the transmit equalizer:

- match the impedance of the channel and buffer the signal before going into digital logic

- provide high-frequency peaking in attempt to extend the frequency range of the channel and decrease ISI

The main idea of this type of equalization is basically the same as for the transmit-side equalization, as we have explained in Fig. 2-2. The objective is to make overall transfer function as flat as possible in the region of interest. The region of interest is, again, defined as the range of spectral frequencies where transmitted signal has significant components. This is, of course, an attempt to make undistorted signal transmission. There are, however, some differences.

Firstly, this is a continuous-time system, which means that the transfer function is not periodic. Furthermore, the load of the output node is the input into decision stage which might be less challenging to drive properly than the channel. Maybe most importantly, this block is capable of introducing gain into system transfer function.

Usually, this system is implemented as a selective amplifier with inductive peaking and transfer function

$$rxF(s) = G\frac{s}{1 + \frac{s}{Q\omega_0} + \frac{s^2}{\omega_0^2}} \tag{2.5}$$

or a system with capacitive source degeneration of a differential pair, producing similar peaking transfer characteristics:

$$rxF(s) = G\frac{1 + \frac{s}{z_1}}{(1 + \frac{s}{p_1})(1 + \frac{s}{p_2})} \tag{2.6}$$

In the first case the transfer function has complex-conjugate pair of poles producing a peaking characteristics. In the latter, the poles are always real, and we have to ensure such parameters that will produce the zero at the lower frequency than the poles, to achieve some peaking in the transfer function.

This is an analog system and to be incorporated into our formulation it has to be discretized. There are a couple of different approaches to discretization of an analog system (i.e. impulse invariance, pulse invariance, pole-zero mapping, ...) [34]. For simplicity we usually employ the Tustin's mapping [34] as it enables simple and easy to track and debug interface between physical (circuit) parameters and top-level system parameters. The main concern when discretizing an analog signal is the accuracy of the digital approximation. This is directly related to the aliasing problem [33], and relatively high oversampling should be employed to accurately capture analog system behavior [34].

We will discuss the circuit model later on in the following chapters. At that point, we will also introduce the sampling method we used, in detail and give all the appropriate explanation. At the system level we only need the discretized $\mathcal{Z}$-domain transfer function of the receive equalizer block. In our case, that means: equivalent gain and digital poles/zero locations.

Note that digital filters we derive from this circuit operate per-sample as opposed to per-symbol filtering of the transmit equalization filter. This is obvious from equation (2.2) where we can see that transmit-side per-sample behavior is obtained by upsampling per-symbol response through zero-order hold.

To really answer a question of proper, implementation-aware equalization and system level design of a HSL we have to design a joint system and circuit optimization-based framework that can take all these effects into account.

Slicer/latch

This is a decision element. At this point in the receiver the transmitted bit sequence is being reconstructed from the received waveform. This block functions as a simple, but very fast, A/D converter. The performance of the whole link can be estimated from the knowledge of the signal quality and noise measure at the input to this block.

It is often the case that slicer is being combined with FIR filtering in the feedback thus producing a nonlinear equalization structure known as *decision feedback equalizer* (DFE) [42]. This is especially the case with strong postcursor ISI where linear equalization is not powerful and efficient enough.

In our work we will not consider the DFE case. Modeling DFE at this (system) level is easy: we would just exclude certain number of postcursors from eye calculation, expecting the DFE to correct them. However, the circuit implementation might need some more attention as it is fairly complex subsystem. The goal of this thesis is to show potential of optimization-driven design flow in bringing circuit and system level design phases closer together.

Transmitter predriver

Link designers use transmitter predriver as an interface between digital logic, usually being implemented in minimum size transistors, and transmit equalizer which presents significant input capacitive load due to large transistors we are forced to use in order to deliver significant signal power into the channel.

Implementation is usually very simple: inverter chain. Main contributions of this block are dynamic power and certain delay.

2.1.2 The timing subsystem

Both the transmitter and the receiver in a synchronous link need to have timing information. In our model, Fig. 1-1, the sub-blocks providing time-keeping functionality are transmit side PLL and CDR in the receiver.

It is customary to generate clocking for the transmitter by means of a phase locked loop (PLL) [7]. Such an architectural setup is convenient as only low-frequency clocks are distributed globally. This decreases number of high frequency lines in the design thus decreasing electromagnetic interference and power consumption. It is, also, necessary if the data rate is to be programmable or the reference clock does not meet timing jitter specifications.

The CDR loop is supposed to extract appropriate timing information from the incoming signal and adjust sampling phases in the receiver to optimally sample the incoming signal, if possible. It also contains some form of PLL.

The main trade-off in a PLL block is between clock quality in terms of phase noise/jitter and power [20] and it has been shown in [9] that circuit optimization can be used to assist this complicated trade-off. A reasonable design even for these blocks is somewhat dependent on the channel properties. All timing imperfections are being transmitted through the channel as edge modulation on the bit sequence [43]. Thus the overall effect, as seen at the receive end, is dependent on both PLL circuit and the channel.

While the effects of transmit jitter are, undoubtedly, important, they can be ameliorated with the appropriate equalization as it follows from the model in [43, 42].

This part of the system and its influence will not be considered in this work. For the time being, we will assume that we can perform ideal synchronization in the system. We will, however, try to provide an environment that is general and enables further manipulation of the data structures provided to make these kinds of extensions possible.

2.2 Performance metrics: BER and worst case eye diagram

In our design objective for a HSL, we want to minimize power of the design, given performance specifications, as expressed in Formulation (1.1). Obviously, once the circuit constraints are met, system performance is expressed in terms of BER and power. Estimating power is straightforward from circuit model, once the circuit and system formulations are jointly described.

In order to estimate BER of a communication system designers usually consider signal and noise properties at the input of a decision element [43]. Thus, the slicer decides based on the value of the sample at a sampling phase. Noise present in the system is usually modeled as AWGN coming from circuits and the channel. Consecutively, system performance can be estimated once we can estimate sampled output value depending on the top level system parameters. This is achieved by constructing the eye diagram samples under worst case ISI at each sampling phase. Since ISI is dominant error mechanism [43] we use simplistic AWGN noise model at the input of the slicer with some given standard deviation.

In this subsection we explain how we construct the eye diagram. We look into influence of each sub-system to the eye diagram. Finally, we describe how we use it to find biasing conditions for circuits and how we try to calculate BER from it.

2.2.1 The eye diagram

One of the most widely used tools to visualize performance of a digital communication system is the eye diagram of the signal at the input of the sampler (decision element). *Open eye* means that there exists a moment in each symbol frame where we can unambiguously decide which symbol was transmitted.

Ideally, it is obtained by overlying all the possible transitions during one symbol time on the same timing diagram. More practically, it is obtained on the scope or in simulation by overlying one symbol time long frames of the analyzed waveform, which is usually transient response to a certain input symbol sequence.

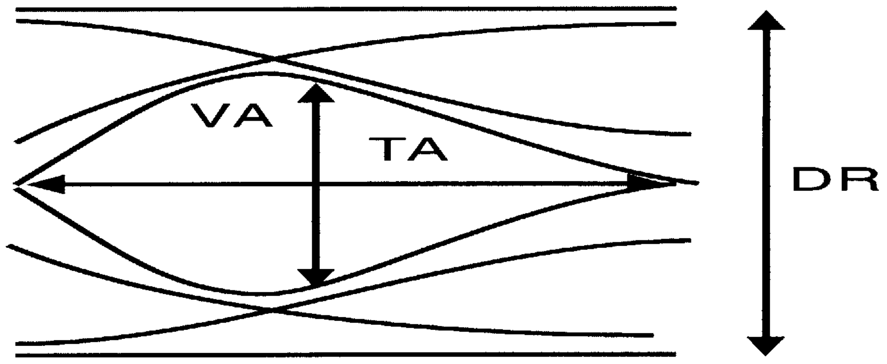


Figure 2-5: Example eye diagram

An example eye diagram is presented in Fig. 2-5. Some of the most interesting characteristics are also indicated on the picture. The most interesting properties of an eye diagram are:

- voltage aperture (VA) - defined as the sampled value at the appropriate average sampling phase
- time aperture (TA) - defined as the horizontal (time domain) opening of the eye
- dynamic range (DR) - defined as the maximal deviation of the signal from the mean value

It is usually enough to know one sample phase of the eye in order to make a decision. This means that we could, theoretically keep track of only symbol-space sampled SR of the system, and perform equalization according to the impact it makes to this particular sampling phase. Such approach would make sense from the formulation standpoint, as we have less equations to keep track of, thus saving in parsing and solving time and formulation file size. However, if we do keep track of one sampling phase only, we will not be able to provide any possibility to include the effect of the CDR to the system in the future. This is obvious if we remember that CDR operates by observing the symbol transitions (for example 0 to 1 and vice versa in binary communication) while the data recovery path observes the signal at the appropriate moment between two transitions. Other way to note this is to say that the sampling clocks for data sampling and incoming timing extractions are, usually, in quadrature.

Thus having multiple sampling phases of the eye diagram can be helpful dealing with the jitter and timing imperfections. Currently the optimization formulation we are providing needs to be given the sampling phase we expect the eye to be opened. With a little experimenting we can determine good candidates and proceed with other aspects of the design.

Finally, there is a strictly technical aspect of this choice to keep multiple phases. Having only one sampling phase makes debugging of the system level formulation much harder. In all equation-based optimization-driven design approaches one big issue arising is a debugging methodology. This is because we have only limited amount of data. As we noted before, the consequence of the presence of the designer in the loop introducing only relevant details. It greatly reduces the solution time thus improving the efficiency of this approach over simulation-based approaches. However, reducing amount of information can be a problem in debugging and testing phases, while a formulation is being developed.

Unlike with a simulation where we can observe many states of the system tracing the problem back to its cause, in optimization driven approach it is usually not possible as we have only information on sensitivities of constraints that we specified. For example, if only one phase is available and the formulation cannot find an appropriate equalization to open the eye, we cannot know if the system does not have any chance of opening the eye, or the eye cannot be opened at that particular sampling phase. Having multiple phases inside the optimization run provides valuable information as we can see what the optimization engine is seeing in the system, and observe much more through analyzing decisions made for different sampling phases.

Thus, tracking multiple phases could be beneficial and we take that into account when formulating the system level optimization model. Through series of transformations that we introduce, we manage to arrive to a simple, compact and reliable formulation, that enables quick parsing and solving and greatly reduces the performance cost associated with keeping multiple sampling phases in the formulation.

2.2.2 Worst case eye diagram

If we agree to use the worst case scenario design, the eye diagram of the communication can be simply calculated. As the usual SR of a communication channel has between 20 and 30, Fig. 2-1(a), significant samples this approach is well justified for low bit error rates (BER) of order less than $1e-15$, the usual HSL specifications. (We will verify this later and relate it to Fig. 2-7.)

In a (relatively rare) case of a channel that has many significant samples, this approach can be too conservative as the worst case sequence estimated from a very long SR is very unlikely to happen. As an example we can note that 10^{15} is approximately expressed as 2^{45}. Thus this problem should not be considered unless the impulse response has more than (approximately) 45 significant samples. In the case of a very long impulse response some approximate analysis treating certain, reasonable, number of most important samples in the SR as deterministic ISI and others as a source of noise might bring more realistic results [43].

With previous discussion in mind, to determine the worst case input sequence, we start from the SR (h_n) of the system. For the moment we will assume that the SR is bit-space sampled. At certain moment T we have, at the output of the system:

$$y_T = \sum_{k=1}^{N} h_k b_{T-k\Delta T} \tag{2.7}$$

which is just the convolution between SR of length N and the incoming bit sequence b_k. Suppose that the main tap of the system in question is at position h_m. Suppose also that we have transmitted symbol 1 (in binary case). The worst case scenario would yield sampling value of

$$s_T = h_m - \sum_{k=1,\ k\neq m}^{N} |h_k| \tag{2.8}$$

We can interpret it as follows: the symbol transmitted at time $t = T - m\Delta T$

produces the main tap at time T and it is considered to be 1 (we assumed symbol 1 is being transmitted). All the other bits that produce any significant interference (basically those bits whose SR overlaps with sampling time T) are chosen in such a way to inflict maximum destructive interference at time T. This basically means that they are chosen according to the following formula

$$b_{T-(m+j)\Delta T} = -sgn(h_{m+j}) \tag{2.9}$$

where $j = -(m-1), \ldots, N-m$ and $j \neq m$.

The same procedure we just described in case of symbol-space sampled SR can be performed in case we have an oversampled SR. The only modification is to carefully account for the interference at any point as it is still constructed from the symbol-spaced samples. For example: should we have a SR oversampled with *ovsRate* the worst case sample at any sampling phase k would be

$$s_T^{(k)} = h_k - \sum_{j=1,\ j\neq k,\ mod(j-k,ovsRate)=0}^{N} |h_j| \tag{2.10}$$

which is just a generalization of Eq. (2.8). As the waveform acquired in this process is longer than one symbol interval, it cannot be the eye diagram. An explanation is due.

In Fig. 2-6 we can see how this waveform can be used to construct the real eye diagram. We also note that for clarity only one side of the real eye is shown. The other side we can be obtained by mirroring around the axis, or equivalently taking the negative of the lower part to get the upper part of the eye. We refer to this waveform as *the unwrapped worst case eye.*

In Fig. 2-7 we compare a simulated eye diagram (in blue) with the unwrapped eye (given in red). The unwrapped eye we calculate according to Eq. (2.10), while the eye diagram was obtained from a simulation for the channel presented in Fig. 2-1(b), by transmitting a pseudo random bit sequence (PRBS).

As we can see from Fig. 2-6 and Fig. 2-7 this waveform contains all the significant information provided by the eye diagram: voltage and time aperture and the dynamic

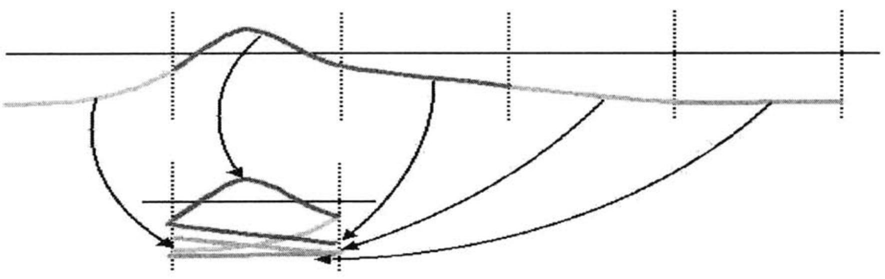


Figure 2-6: The unwrapped eye

range. It takes into account the worst possible case for any sampling phase, at any delay, in the SR. While only one part of this waveform actually corresponds to the eye opening seen on the eye diagram, the rest of it also provides useful information, as we can confirm from Fig. 2-7 and Fig. 2-5. We could construct the eye diagram with all the interesting extremal points if we slice this waveform into pieces, each of which is one bit-time long, Fig. 2-6. Exactly this waveform is used in our system instead of the actual eye diagram. The reason is simple: all the useful information contained in the eye is available here, and this waveform is easily determined without any simulation.

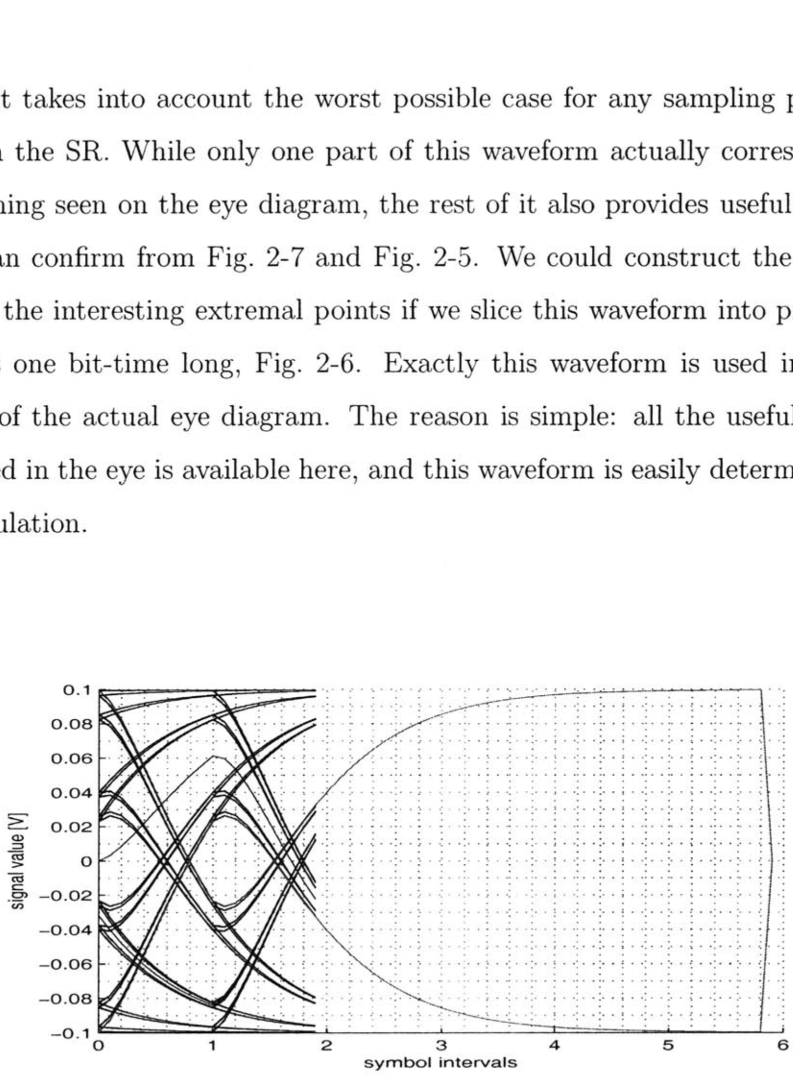


Figure 2-7: Comparing the unwrapped eye and simulated eye

2.3 System level abstraction and hierarchy

Here we explain different approaches to formulate system level model. Then we explain the problems in each of them and give a solution.

In the system level model we use parameters of the signal processing chain to estimate operating conditions of the circuits in the chain as well as the influence of filtering to the system performance. For this to be done we need to calculate the worst-case eye at the sampling phase that we believe will be chosen and ensure that the decision element can resolve with certain BER in the presence of noise. The advantages of calculating multiple possible sampling phases have been discussed before.

In order to determine the influence of each block in the signal chain on the signal waveform we abstract them into parametrized equation-based models (transfer functions). At the system level we represent each block with information needed to account for its modifications to the signaling environment. The implementation of the circuit and any internal modeling is left for the circuit module which we describe in Chapter 3.

This coding style yields clear, hierarchically organized code. It also defines a very natural interface between system and circuit level, and thus decouples the implementation modeling from performance modeling at the code level. This makes it easy to modify the formulation to account for different circuit implementation, as long as it fits well into system level abstraction, and can be properly linked to it.

Here we state the interface variables for each block at the system level, so that we can discuss problems and solutions in making the system design computationally tractable. The reasons for this particular model at the system level will be clarified in Chapter 3 when we take a look into circuit implementations.

Here we also introduce the default names used throughout this text in reference to system level abstractions of the circuit blocks.

Transmit equalizer - txEQ. This block is represented as a symbol-space sampled FIR filter . We also model the output impedance and impose a constraint that a pole introduced at the circuit output has to be significantly higher in frequency than the Nyquist frequency of signaling. This should prevent the degradation of the performance as the output transistors become large, Chapter 3. We also export signal swing from this block as well as the power estimate, Table 2.1[4].

Table 2.1: Transmit equalizer (txEQ) system level variables

variable name	*variable symbol*	*description*
FIR filter coefficients	[1 *aFR1 aFR2* ...]	*normalized* (*aFR*0 = 1) *vector of transmitter equalizer coefficients*
output swing	*swingIN*	*maximum transmitter swing*

Note that we export the absolute maximum swing that transmit equalizer can produce at its output. In order to derive the digital domain transfer function of this block for the top level we have to take into account the peak swing constraint dependency on tap coefficients as we discussed previously in Eq. (2.3) and Eq. (2.4). The simplest way to ensure that we account for the changing gain of the normalized tap coefficients is to define some equivalent 'reduced' swing and use it at top level. By putting

$$swingRED = \frac{swingIN}{\sum |aFR_k|} \tag{2.11}$$

we ensure that maximum possible swing for any input sequence in transmit equalizer is exactly *swingIN*. Thus, we can define the top level transfer function (in symbol-spaced domain) of the transmit equalizer to be

$$txEQ(z) = swingRED \sum_{j=1}^{N*ovsRate} a_{div(k,\ ovsRate)} z^{-j} \tag{2.12}$$

which we oversampled in order to obtain sample-spaced discrete representation. In

[4]The filter coefficients are normalized for simplicity and debugging purposes. This filter setting, however, is not the most general one. This comes from the fact that the first coefficient $aFR0$ is fixed and with the positive sign. Most of the modern passive channels have positive precursors. If we, for example, try to delay SR for one symbol period, as main symbol creates positive interference on the next symbol in sequence, we can conclude that $aFR0$ should be negative in order to compensate for that positive interference. This is not possible in our setup as $aFR0 = 1$. As this is a proof of concept system we did not treat this in greater depth. However, by defining $aFR0 = -1$ and repeating the procedure we could cover most of the necessary cases.

Eq. (2.12) we use $div(r, q)$ as a standard notation for integer division. As before *ovsRate* is the oversampling rate used in top level formulation.

As the solver we utilize is based on a GP engine it can handle only positive variables directly. In order to enable variables that can change their sign we define two new variables and express the desired variable as the difference of these optimization variables. This is a standard optimization trick widely used when standardized optimization formulations assume positive variables.

$$\begin{aligned} & var\ aFR1p,\ aFR1n \\ aFR1 &= aFR1p - aFR1n \end{aligned}$$

We can use this at the top level, sometimes, if we are not sure which sign a particular tap should have.

Receive equalizer - rxEQ. The representation of this block can vary slightly depending on the implementation chosen in the circuit domain. It is considered to be a combination of a sample-spaced FIR filter, accounting for the effect of the zero in the transfer function of the peaking amplifier, and a couple of IIR filters representing the effect of the poles. We also take into account the gain and the static power of that block, Table 2.2[5].

The reader might note that we have the same name for the rxEQ and txEQ power numbers. This is not a problem since, as we discussed, the code is hierarchically written and each name has its proper domain resolving any ambiguities. We will clarify in text or by the structure reference such as *rxEQ.P* or *txEQ.P* if the context is not clear.

In our formulation we assume the following transfer function of the receiver equalizer (in sample-spaced domain)

[5]Note that the names of poles/zeros are very specific in text to make following the appropriate optimization code easier. In general these names can be arbitrary and no assumption on the properties of any circuit is taken at the system level formulation. The only real requirement is that their transfer function can be obtained as a rational function in $\mathcal{Z}$-domain.

Table 2.2: Receive equalizer (rxEQ) system level variables

variable name	*variable symbol*	*description*
gain	*scale*	*equivalent Z-domain gain of the system*
zero	zRX_1	*coefficient next to the* z^{-1} *in the numerator of the transfer function*
drain pole	$dpRX_1$	*inverse of the pole formed in the drain (in digital domain)*
source pole	$spRX_1$	*inverse of the pole formed in the source (in digital domain)*

$$rxEQ(z) = scale\frac{1 - zRX1z^{-1}}{(1 - spRX1z^{-1})(1 - dpRX1z^{-1})} \tag{2.13}$$

and parameters of this transfer function are exported to the top level formulation, Table 2.2. Note that *scale* is not exactly DC gain of the circuit in discrete domain as it does not correspond to $rxEQ(z = 1)$ but $rxEQ(z = 0)$. All the necessary details to derive such form of the transfer function will be explained in Chapter 3.

We will discuss additional details of the formulation in the last chapter where we show the results. Here we will only note that in order to estimate system behavior we need to know eye diagram at the receiver equalizer's output. However, to ensure proper receiver polarization we need to estimate the dynamic range at the receiver input.

Channel. The channel is modeled as an FIR filter with known coefficients. As these coefficients are just constants obtained from s-parameter measurements and transformed into impulse response of the channel they will appear only as constant coefficients in our final transfer functions. Formally we can write it as

$$C(z) = \sum c_i z^{-i} \tag{2.14}$$

Slicer. This is the final block of the signal processing chain in our simple link model. It is currently not modeled to high accuracy. Currently we model it as a relatively simple system, Table 2.3. This is a very approximate model and modeling and characterization of the slicer block remain a part of future work.

Table 2.3: Slicer system level variables

variable name	*variable symbol*	*description*
input noise	*sigma*	*input referred noise of this block*
sensitivity	*level*	*input signal needed for resolving*

We will describe all the circuit modeling in depth in Chapter 3.

Having defined all the effects that we wish to account for at the system level, we can proceed to different types of optimization formulations that make use of the parameters discussed here in order to estimate link performance.

2.4 System level formulations

As we have discussed in previous sections, in our example formulation we model (with with certain simplifications) the general HSL top level, Fig. 1-1, as a parametrized transfer function in $\mathcal{Z}$-domain of the link. This approach yields the model presented in Fig. 2-8.

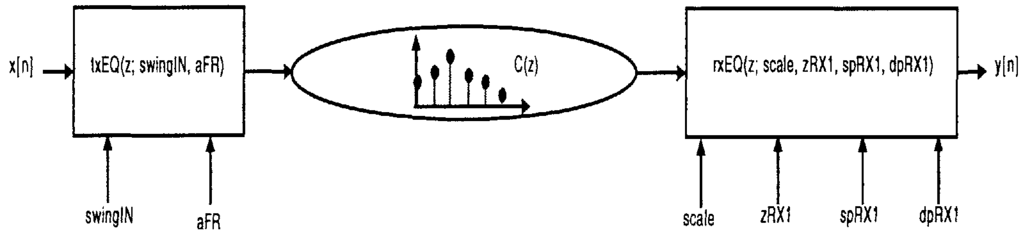


Figure 2-8: Block diagram of our top level formulation

for which we can define the overall transfer function, Eq. (2.15). This transfer function is compiled from the channel, oversampled transmit equalizer and receive equalizer transmit functions. Using the notations we introduced here we can write it down as

$$tf(z) = txEQ(z; swingIN, aFR)C(z)rxEQ(z; scale, zRX, spRX, dpRX) = \frac{p(z)}{q(z)} \tag{2.15}$$

where we recognize the fact that in discrete domain the system model we described has a rational transfer function. Furthermore, we can just ignore any leading delay, expressed as z^{-k} in $p(z)$ for some k. This will not change the results of equalization and it only expresses the fact that system has certain transport delay through the channel as well as some delay that appears in the process of discretization. Part of the delay can be taken out when the channel transfer function is defined.

Transfer function of the system $tf(z)$ is parametrized and depends on parameters of all the subsystems we described. Our goal is to determine a good set of parameters that will be feasible from the circuit point of view, while they provide good equalization settings for desired system performance. This is what optimization engine should ensure.

2.4.1 Time domain formulation

The first problem with this is that rational functions, in general, are not the easiest to deal with in an optimization engine. This comes from the fact that they can be highly non-convex, and as all the other computations consisting of divisions introduce accuracy and range problems.

One of the first attempts we made to reformulate or simplify the problem was to deal with the denominator of the transfer function by expanding it into Taylor series, thus transforming a rational function into polynomial function, like in Eq. (2.16).

$$f(z) = \frac{1}{1 - qz^{-1}} \rightarrow f(z) \approx 1 + \sum_{k=1}^{N} q^k z^{-k} \tag{2.16}$$

While this produces an optimization friendly (for the type of solver we decided to use) formulation that had shown good performance it is very limited due to its size. Namely, it is easy to see that the bandwidth of the equalizers will be around the Nyquist frequency of signaling for the system. Also, if the channel requires lots of peaking in order to equalize the impulse response of the equalizer could last for significant number of bit-periods. Furthermore, because of the discretization error due to sampling and aliasing the filters should be oversampled enough to provide accurate calculation [34, 33]. These constraints, combined, yield the requirement to do Taylor expansions with large number of terms. Once this is done, we will be convolving several long FIR filters, thus getting a symbolic formulation with very large number of terms that is very impractical in terms of file size, parsing time as well as time to generate the problem.

Just as an example we can quickly calculate:

- SR (channel) samples: 20 (bit-spaced)
- IIR filters: 2 filters, 1st order each
- IIR expansion terms: 15 (which gives accuracy of 3% for typical pole locations)
- oversampling: 10 (relatively low)

for which we obtain $(20 \times 10) \times (15 \times 10)^2 = 4,500,000$ terms. Of course, all these intermediate results would fit in overall response that is $(20 \times 10) + (15 \times 10) + (15 \times 10) = 500$ samples long, where each sample consists of many terms. Furthermore, our link model should have at least 2 IIR filters and 2 FIR filters making this calculation even more pessimistic.

This, indeed, is not the right way to handle these problems. As we are aiming to provide an efficient and fast tool for design space exploration we need to be able to generate system level description quickly. This is particularly important in the debugging phase when new circuit modules are being linked in the system level as long delays in obtaining the results make it hard to track changes as the designer is forced to run multiple attempts simultaneously in order to save time.

2.4.2 Frequency domain approaches

As usual the frequency domain provides more efficient way to handle system level design and manage smaller amount of data, which is one of the main reasons for a number of research efforts in model order reduction (MOR) [4, 40]. Here we describe a couple of different approaches that we have attempted and how we tried to make them fit into solver engine better.

In order to motivate these changes we will discuss what we have learned from the first attempts with time domain expansions.

For this whole flow to run as smoothly and efficiently as possible, we need to pay attention to each step in the process. The formulation has to be easy to generate, so that the system-level formulation generator (written in Python) would not be the bottleneck. The formulation should not have large expressions with many terms as we discovered that parsing time is strongly dependent on this parameter. Parser is also very sensitive on existence of rational functions as different types of transformations are being applied during the pre-solving procedure [30]. Finally, solution time is strongly dependent on the number of variables in expressions. It is a consequence of GP engine being guided to a solution using branch & bound search [2]. In each step a system of equations is being solved, and the size of the system depends on number of different variables appearing in appropriate expressions being treated. If this number is large, the sparsity patterns could be lost during the solution phase, thus stalling and even running the solver out of memory for moderately sized problems.

Direct IDFT approach

One of the first things attempted when working in frequency domain was to calculate transfer function of the signal chain and come up with the expression for each sample through Inverse Discrete Fourier Transform (IDFT).

This approach, however, has bottlenecks in system level formulation generator and parser. To see this we can look into expressions we obtain with this method. We start with the transfer function which is a rational function

$$tf(z) = \frac{p(z)}{q(z)} \tag{2.17}$$

which we sample at a certain number of points on the unity circle in the complex plane [33]. Let us denote these points as $z_0,\ z_1,\ \ldots$. This way we obtain a discretization of the transfer function that enables calculating certain number of time domain samples, Eq. (2.18).

$$\begin{bmatrix} Y_0 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} z_0^0 & z_0^1 & z_0^2 & \cdots & z_0^n \\ \vdots & & & & \\ z_n^0 & z_n^1 & z_n^2 & \cdots & z_n^n \end{bmatrix}^{-1} \begin{bmatrix} \frac{p(z_0)}{q(z_0)} \\ \vdots \\ \frac{p(z_n)}{q(z_n)} \end{bmatrix} \tag{2.18}$$

While this approach is easy to implement, the resulting expressions for each of the output samples, $Y_0,\ Y_1,\ \ldots$ are in the form of a sum of rational functions. This is not a very good formulation as it suffers from accuracy and scaling problems due to divisions for such z_i which are closer to π on the unity circle, as they correspond to higher frequencies in analog domain. Furthermore, a sum of rational functions with many different denominators would take long time to rationalize. Once in the rationalized form it would probably be a good formulation but it is either the top level formulation generator or the parser that would have to perform this work, and from our experiments it is not a very efficient way to formulate the problem.

In order to estimate the worst case voltage margin at a certain phase, according to Eq. (2.8) and Eq. (2.10), we need to know absolute values of the interfering samples. This can be done in couple of ways [2].

We can use a standard trick used in linear programming [3] by overestimating the interference and leaving to the optimization engine to make them tight. This would be accomplished by introducing constraints

$$-absY_n \leq Y_n \leq absY_n, \tag{2.19}$$

where $absY_n \geq 0$, and rewriting the Eq. (2.8) with

$$s_T = Y_m - \sum_{k=1,\ k\neq m}^{N} absY_n \tag{2.20}$$

where Y_k are samples of the system SR, thus taking into account both channel and equalization effects.

This technique works only if we impose lower bound constraint on a certain sample s_T which will always be the case as we will always require eye open enough for the slicer to decide on the value.

One should also keep in mind that this approach overestimates interference from each interfering sample. The trick is in the local optimality of the solution. Namely, when solver engine is searching for the solution, if the absolute value inequalities for the interference samples of the desired sampling phase are not active (meaning satisfied as equalities) it is obvious that we could change absolute value estimates by lowering them, see Eq. (2.19). This would be observed as an increase of the appropriate sample of the unwrapped eye, Eq. (2.10). Thus we have increased the eye opening, without inflicting any penalty in terms of optimization objective: system power. This shows that in any local optimum, absolute values of the interfering samples have to be estimated accurately and appropriate inequalities in Eq. (2.19) satisfied as equality from one side, depending on the sign of Y_n.

Fitting IDFT approach

As we mentioned in previous section, sum of rational functions did not have predictable and robust performance. This is, speculatively, due to the parser handling and decomposing of the expressions of this type [30]. If this is correct assumption, it would be split into many smaller expressions and each different expression would be renamed to a different variable. This would introduce large number of artificial variables. This action would, even though the original expressions have very small number of actual optimization variables, create expressions with very large number of auxiliary variables in each. This situation was already discussed and we noted that it could lead to 'fill-in' of sparse matrices during solving and run the process out of memory.

We needed a formulation that is simple enough to bypass the parser which operates on assumptions which were made for circuit formulations and expectations, but are not true, in general, at the top level. Some problem specific properties and structure had to be exploited.

While in the general case this type of expression can be very complicated, we know the genesis of this particular one. It is the IDFT of a certain transfer function. This special structure enables some pre-processing to be done 'by hand' thus simplifying the input formulation.

We start by writing a generalization of Eq. (2.18). To do this we move our derivation completely into $\mathcal{Z}$-domain. Assuming we have samples of the SR, Y_k we can write

$$\sum_{k=1}^{\infty} Y_k z^{-k} = \frac{p(z)}{q(z)} \tag{2.21}$$

which is a relation very often treated in model order reduction [19, 4, 40] in attempts to find a compact representation, transfer function, from a given time domain response.

In our work we will go in the opposite direction. We start with parametrized transfer function (parameters being poles and zeros of the filters in the chain) and we want to come up with sample values. In order to avoid problems coming from the existence of rational functions in the formulation, we will simply multiply both sides with the denominator of the transfer function

$$q(z) \sum_{k=1}^{\infty} Y_k z^{-k} = p(z) \tag{2.22}$$

By truncating the infinite length of Y_k after certain number of samples, and sampling this equation in certain number of equally spaced points on unity circle we get system of equations

$$q(z_j) \sum_{k=1}^{N} Y_{k*ovsRate} z_j^{-k \times ovsRate} = p(z_j) \qquad \text{where} \qquad z_j = z_{offset} + j\pi/N \tag{2.23}$$

This is, basically, a fitting problem where we try to fit variables Y_k to approximate the transfer function. Note that we fit in the range $[0,\ \pi]$. This is correct because all the parameters in the equation are real numbers. This formulation avoids the problem with rational functions. However, it still has large number of variables (especially Y_k) in each expression. This is still troublesome for the solver and the solution time is very poor.

By sampling in $\mathcal{Z}$-domain we effectively resampled the SR to some new oversampling rate *ovsRate*. Furthermore, the overall transfer function includes the influence of the channel. As we already noted channel is a low-pass filter with very low bandwidth compared to the bandwidth of other filters. This means that we can safely lower the oversampling rate without loosing any information or getting any significant aliasing. By decreasing the number of phases we decrease the number of variables in each equation, thus making the fitting problem easier. The question that remains is how do we actually extract all the samples we need if we are to keep number of Y_n variables low?

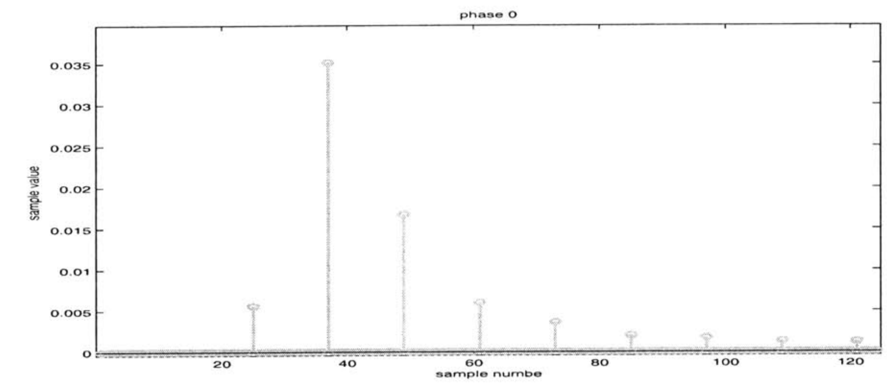
This is relatively easy to do by mimicking the modified $\mathcal{Z}$-transform. We extract one set of phases and then repeat the process by changing z_{offset} in the range $[0, \pi/N]$, Figures 2-9(a), 2-9(b) and 2-9(c). Thus, we always extract N phases. Suppose that we want to extract $K \times N$ phases. With this method we have a problem that has K small problems in parallel thus being $O(K \times N^2)$ while if we had used one full-sized problem we would have had $O((K \times N)^2)$ size of the problem.

This formulation is a good choice for system level description. It is easy and fast to generate, and parser and solver perform reasonably well with it. We note that formulation is based on signomial functions without any divisions. Also number of terms in each expression is significantly less than in the case of direct IDFT described previously. There is, however, room for further improvement, as we describe in the next section.

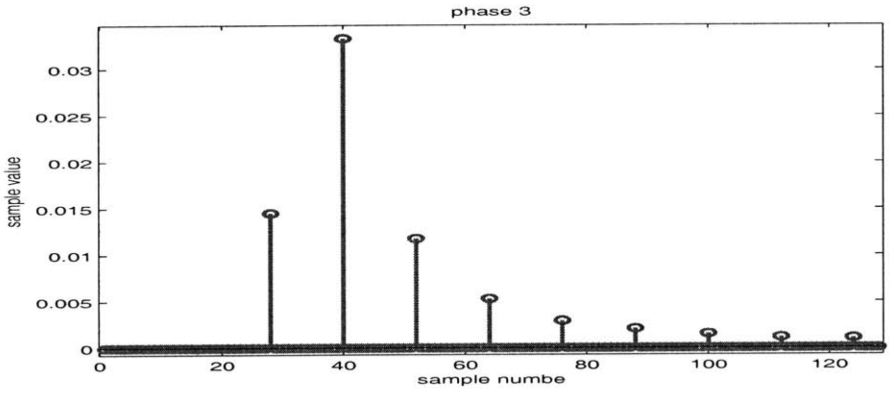
Modified IDFT approach

As we previously discussed, lowering the number of optimizer variables in expressions has very positive impact on performance. We should strive to do that if possible. Furthermore, previously described formulation does need certain tweaking of the solver accuracy parameters to perform well. In order to simplify even more we start by looking into Eq. (2.18) and Eq. (2.22).

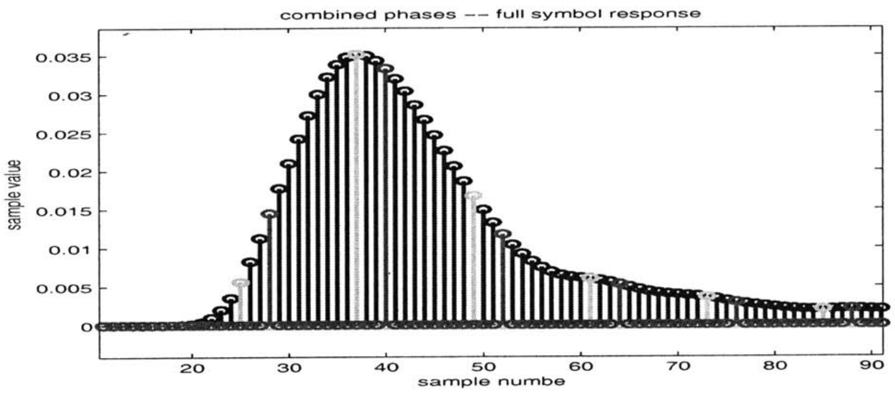
Note that if term $q(z)$ was not present at the left-hand side of Eq. (2.22) we would be able to transform that equation by means of sampling on unity circle we already described, into equation similar to the Eq. (2.18) but without rational functions on



(a) $z_{offset} = 0$



(b) $z_{offset} = 3$



(c) Combining all extracted offsets we reconstruct the whole SR

Figure 2-9: Phase sweeping process for extracting multiple sampling phases of the SR

the right-hand side. In order to accomplish this we can perform convolution of $q(z)$ and $\sum Y_k z^{-k}$ and obtain equation of the form

$$\sum_{k=1}^{\infty} Q_k z^{-k} = p(z) \tag{2.24}$$

If we approximate by truncating to N samples and do the frequency domain sampling we will be able to write an equation very similar to Eq. (2.18)

$$\begin{bmatrix} Q_0 \\ \vdots \\ Q_n \end{bmatrix} = \begin{bmatrix} z_0^0 & z_0^1 & z_0^2 & \cdots & z_0^n \\ \vdots & & & & \\ z_n^0 & z_n^1 & z_n^2 & \cdots & z_n^n \end{bmatrix} \begin{bmatrix} p(z_0) \\ \vdots \\ p(z_n) \end{bmatrix} \tag{2.25}$$

At this point we can substitute every Q_k with the appropriate expression, thus getting the formulation that explicitly defines each Y_k as a function of numerical constants and only the absolutely necessary variables defining filtering poles and zeroes. If we assume our denominator to be

$$q(z) = \sum_{j=1}^{r} q_j z^{-j} \tag{2.26}$$

and further assume $q_0 = 1$ for normalization we can derive a linear matrix equality defining all the output samples recursively, which combined with Eq. (2.25)

$$\begin{gathered} Y_0 = \sum_{k=0}^{n} z_0^k p(z_k) \\ Y_1 + q_1 Y_0 = \sum_{k=0}^{n} z_1^k p(z_k) \\ \cdots \\ Y_r + q_1 Y_{r-1} + \sum_{j=2}^{r} q_j Y_{r-j} = \sum_{k=0}^{n} z_r^k p(z_k) \\ \cdots \\ Y_n + q_1 Y_{n-1} + \sum_{j=2}^{r} q_j Y_{n-j} = \sum_{k=0}^{n} z_n^k p(z_k) \end{gathered} \tag{2.27}$$

Thus, we have compacted our formulation to a very simple form. Only the subsystem variables are present in each expression. Furthermore, the interaction between output variables Y_k could be broken through back-substitution as IIR filter introduces recursion between our Y_k variables, Eq. (2.35).

We should mention one interesting and intuitive interpretation of the algebra we just carried out. The original equation

$$Y(z) = \frac{p(z)}{q(z)} \tag{2.28}$$

is written in in the form of

$$q(z)Y(z) = Q(z) = p(z) \tag{2.29}$$

where we have observed that due to the structure (polynomial) of $q(z)$ we can achieve a form in which $Q(z)$ and $Y(z)$ have the same form

$$Y(z) = \sum Y_k z^{-k} \tag{2.30}$$

$$Q(z) = \sum Q_j z^{-j} \tag{2.31}$$

From this form of the left-hand side, as we did in direct IDFT, it is easy to obtain expression of each of the samples of the left-hand side as some functions of the right-hand side parameters.

Now we should remember that $p(z)$ is actually product of the channel transfer function $C(z)$ with all the FIR filter transfer functions of the system. It is some auxiliary signal that is not necessarily observed in the system, unless all the FIR filters physically come first in the signal chain, then followed by IIR filters at the output of the signal chain.

On the other hand $Q(z)$ is the output $Y(z)$ waveform deconvolved with all the IIR filters of the system, or equivalently convolved with their inverse FIR filters. We will always be able to perform this procedure in a numerical algorithm as all the filters in the signal chain of a HSL are stable. Results of these two operations, $Q(z)$ and $p(z)$, have to be equal, by definition.

This is a very simple procedure, but might require some further explaining. Now we give simple example noting how each of the described formulations would be obtained.

Example

Suppose we have a system

$$\sum_{k=1}^{\infty} Y_k z^{-k} = \frac{p(z)}{1 - \alpha z^{-1}} \tag{2.32}$$

and we approximate it by only first four terms in Y_k we obtain

$$Y_0 + Y_1 z^{-1} + Y_2 z^{-2} + Y_3 z^{-3} = \frac{p(z)}{1 - \alpha z^{-1}} \tag{2.33}$$

At this point we would be able to perform sampling and formulate our first type of formulation (direct IDFT approach), explained in Section 2.4.2. If we would transform into form without IIR filters and rational functions we would get

$$(1 - \alpha z^{-1})(Y_0 + Y_1 z^{-1} + Y_2 z^{-2} + Y_3 z^{-3}) = p(z) \tag{2.34}$$

at which point we could apply our second formulation in frequency domain (fitting IDFT approach) described in Section 2.4.2. However, instead of doing that, we can perform convolution (multiplication) on the left-hand side and manual IDFT thus obtaining the Eq. (2.25) form with additional relations

$$Q_0 = Y_0 \tag{2.35}$$

$$Q_1 = Y_1 - \alpha Y_0 \tag{2.36}$$

$$Q_2 = Y_2 - \alpha Y_1 \tag{2.37}$$

$$Q_3 = Y_3 - \alpha Y_2 \tag{2.38}$$

$$Q_4 = -\alpha Y_3 \tag{2.39}$$

Note that this system of equations is in the lower diagonal form. This is very fortunate as in this case the Cholesky decomposition often used in optimization algorithms [3] coincides with the actual matrix and does not need to be separately calculated as it would have been (inside solver) for all other formulations.

2.5 Summary

We started this chapter by presenting a simplified serial link system level model we use in our optimization programs. Following is a discussion of the influence of each block and subsystem in the proposed model. We explained the function of signal processing chain and different approaches to equalization. We also gave pointers to the appropriate literature we believe could be useful in extending our current work.

Furthermore, we explained the eye diagram and the role it has in estimating the performance of a given communication system. The new 'unwrapped worst case eye' concept is introduced, explained and the use of it was justified by comparing it to the actual eye diagram. We saw that it contains all the necessary information we could extract from the eye diagram. It is straightforward to calculate and it compares well with the estimates eye diagram would give for most of the channels.

At the end we present different techniques we tried to enable us to calculate the unwrapped eye. We begin with time domain expansion of the overall system transfer function. Although, straightforward and simple to explain it suffers from the exponential increase in size as new filters in the chain are added. In order to reduce the amount of data we need to process when formulating the top level optimization constraints we switch to frequency domain approach. Within this framework we define three different versions: direct IDFT, fitting IDFT and modified IDFT.

These three techniques present the logical and chronological flow of our work. The direct IDFT which is nothing more than simple Inverse Discrete Fourier Transform. While straightforward to generate this approach experiences many difficulties in parsing and solving stage due to sums of rational functions which can be badly scaled.

To ameliorate this scaling problem we decided to treat IIR filters as FIR filters operating on the output waveform. Following this approach we define fitting IDFT where optimization engine performs IDFT during solution phase thus discovering the time domain waveform as it approaches the solution. As our solver is GP-based certain simplifications were necessary in order to improve the speed of the solution.

Finally, by simplifying the fitting IDFT through an observation that IDFT could be performed before formulating problem by introducing new, recursively interdependent, variables into formulations we derive the modified IDFT formulation. It is simple and fast to generate. Furthermore, the resulting formulation is in very convenient form of a lower triangular matrix.

Each technique was described in enough detail for it to be performed. We describe problems and advantages of each of the approaches, based on our experience when attempting to automate its generation and to use it in the optimization. This set of transformations and the resulting top level optimization formulations represent one of the main contributions of this thesis.

Chapter 3

Circuit level optimization model

In this chapter we explain how to link circuit models to the top level formulation we described in Chapter 2. The main purpose of the circuit formulations is to provide an accurate estimate of circuit power performance in the system. We also aim to provide a good starting point for circuit level design. This is accomplished by ensuring that circuit model is properly biased under signaling conditions that arise in a certain system level setup.

As we mentioned in Chapter 1, at the level we are using, optimization variables are transistor sizes and polarizations (Ws, Ls, currents). Using these optimization variables more high level parameters of transistor small signal (g_m, g_{ds}, C_{gs}, ...) and large signal (V_{th}, V_{dsat}, ...) model are expressed through signomial fitting. Thus, formulating the equation-based circuit model is more sophisticated version of the traditional "back-of-the-envelope" calculations that engineers use to gain insight into circuit functionality. We illustrate this process on a HSL example.

3.1 Transmit equalizer

In this section we deal with transmit equalizer circuit block.

An example of the functional sub-division of this block is presented in Figure 3-1. Incoming bit-stream is first passed through series of delay elements. This makes multiple consecutive bits available at the same time. As the delay element is a digital

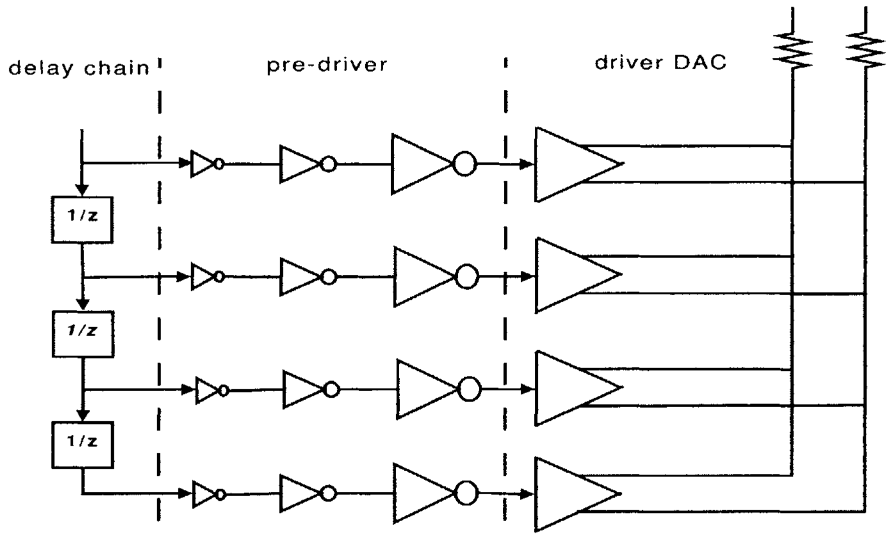


Figure 3-1: Transmit equalizer block diagram

circuit, following stage is a pre-driver that should amplify the signal and provide decoupling between digital logic, usually implemented with very small transistor sizes, and analog front end. Finally, the bit values are appropriately weighted through analog drivers and final analog value is constructed at the output. This final stage can be analyzed as a very simple DAC.

The summation is achieved on output resistance of the driver DAC which is usually matched to the characteristic impedance of the channel, and thus fixed [43].

Transmitter pre-driver

Our current model does not include the delay chain. The pre-driver model can (with certain simplifications) be expressed in GP form [3], which is compatible with our solver of choice. For simplicity, and due to the fact that we are focusing this work on the design flow and methodology we do not provide a detailed model for the transmit pre-driver. We do, however, provide a simple behavioral model. We estimate the dynamic power of the inverter chain as being proportional to the total input capacitance of the channel driver. In this way we constrain the size of the transistors in the channel driver by introducing a direct trade-off with power consumption.

Our main focus here will be the output DAC structure. In order to link this module into the overall formulation we have to make sure that this block exposes interface variables defined in Table 2.1 to the system level formulation.

The model we will develop is based on tap-sharing implementation of the transmit equalizer [41, 46, 43].

This implementation assumes that the transmit equalizer consists of a certain number of identical output drivers which should be connected to the delay chain through multiplexers in order to encode the tap coefficients. With this implementation it is simple to account for the output impedance of the driver as it is fixed regardless of the tap coefficients. However, it introduces some overhead in terms of clocking power due to more complex clock distribution and switching network.

To simplify the formulation even further we define this equalizer block as a hierarchy of the output driver switch and then we combine a number of these switches into the equalizer.

We show the basic building block, a differential switch, in our model in Figure 3-2.

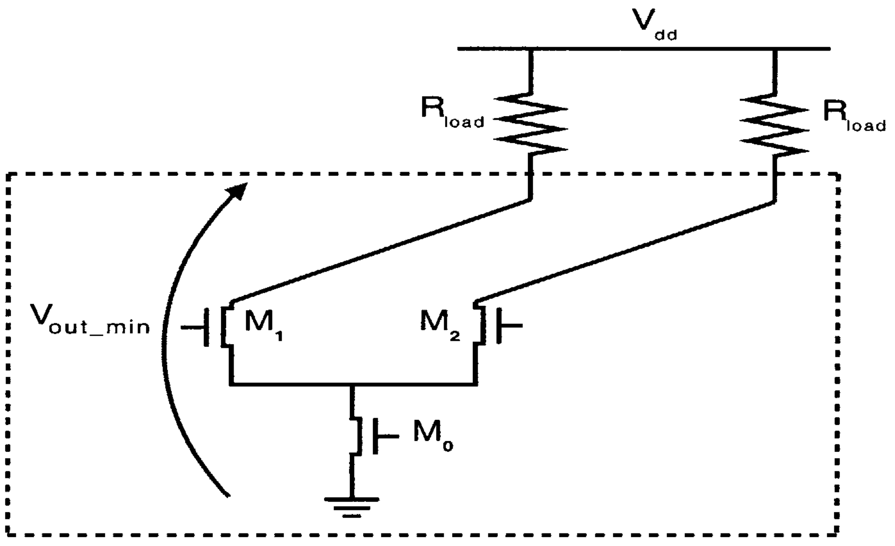


Figure 3-2: Differential switch – basic building block for tap-sharing txEQ implementation

At this level of hierarchy, we instantiate optimization-compliant transistor models. As opposed to the previous GP compliant modeling work [9, 23] switching to signomial formulations enables the inclusion of KCL and KVL into formulation as well. This provides a better estimate of the operating conditions of transistors and improves accuracy. It is especially important in deep sub-micron technologies where square law, used in previous works, as well as the monomial template models, used in works such as [25], fail in expressing transistor characteristics accurately enough over reasonably large set of operating conditions.

We will assume that, in normal operation, circuit in Figure 3-2 behaves as a switch. This observation will enter the equation for minimum output voltage under which all the transistors M_0, M_1 and M_2 operate in saturation to prevent slow recovery in case one of the output transistors would enter the triode region of operation. Under switch operation assumption we will write equations in one of two states of the switch, when the tail device (transistor M_0) current is steered into one of the output branches (i.e. transistor M_1). We will also assume that pre-driver supply voltage is equal to V_{dd} and that this is the input switching voltage for the DAC driver.

To ensure operation in saturation, we export to the higher hierarchical level (the txEQ level) this minimum voltage, referred to the ground node as indicated in Figure 3-2. With formerly described assumptions we can write

$$\begin{gathered}
M_0.ids = M_1.ids \\
M_0.vds \geq M_0.vdsat \\
M_1.vds \geq M_1.vdsat \\
V_{dd} = M_0.vds + M_1.vgs \\
M_0.vdsat \leq V_{dd} - M_1.vgs \\
V_{dd} - M_1.vgs + M_1.vdsat \leq outVmin \\
M_0.vds + M_1.vds \leq outVmin
\end{gathered} \tag{3.1}$$

At the equalizer level we can instantiate the switch as a sub-circuit. Since all the switches are the same, assuming we have L of them we can write

$$
\begin{aligned}
swingIN &= R_{load}L(switch.M_0.ids)\\
switch.V_{out\_min} &\le V_{dd} - swingIN
\end{aligned}
$$

$$
\begin{aligned}
powerS &= L(switch.M_0.ids)V_{dd}\\
C_{in} &= 2L\, switch.M_1.G.cap\\
powerD &= \alpha f_{Nyquist} C_{in} V_{dd}^2\\
powerTOT &= powerD + powerS
\end{aligned}
\tag{3.2}
$$

$$
V_{cm\_out} = V_{dd} - R_{load}L(switch.M_0.ids)/2 = V_{dd} - swing/2
$$

where we assume a switching factor α ($\alpha = 0.5$ due to differential operation of the pre-driver). In order to find the average value of the output signal we assumed that the incoming bit sequence is random, so the average is midway between V_{dd} and $V_{dd} - swingIN$. Note that we use a generic name *switch* to denote the switch sub-circuit. The name in the actual code might be different.

The reader should bear in mind that we already have discussed the difference between absolute maximum swing at the output of the receiver equalizer *swingIN* and the swing we report at the system level *swingRED*. This difference, as described in Chapter 2 is a simple consequence of transfer function normalization for the filter. Note also that filter coefficients (aFR_n) do not figure anywhere in our simple model. This is the consequence of the tap-sharing architecture [46].

Equations (3.2) express all the values required at the top level, Table 2.1, except for the IIR filter arising from the output impedance at the transmitter. In our simple circuit model we decided not to model the influence of this filter at the top level. Instead we impose such constraints to ensure that the pole arising from this impedance in the transfer function is far beyond the region of frequencies of interest

$$Cout = L(switch.M_1.D.cap + spec.padCap)$$
$$Gout = 2/spec.R_{specific} + Lswitch.M1.gds(1 + (switch.M1.gm)(switch.M0.gds))$$
$$Gout/Cout \geq 2(2\pi f_{bit}) \tag{3.3}$$

thus preventing significant deterioration of the signal at the interface to the channel. We can note that this procedure is a 'good design practice' as well, as it goes along the same lines with impedance matching constraint for the channel driving. In Eq. (3.3) we denote with $M_1.D.cap$ total capacitance at the drain of transistor and $spec.R_{specific}$ the pre-specified characteristic resistance of the channel. Finally we put constraint that will drive the pole created at the transmitter output to, at least, twice the symbol frequency.

3.2 Receive equalizer

Our current circuit model assumes capacitive-degenerated differential pair implementation for the receive equalizer, Figure 3-3.

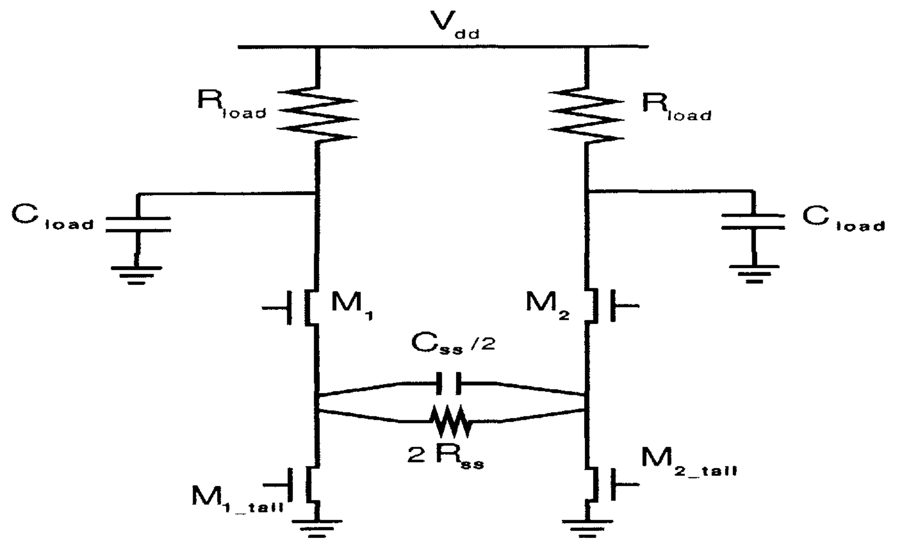


Figure 3-3: Transmit equalizer block diagram

Again, as we have done with transmit equalizer block, we will refer the reader to Chapter 2. This time we will be dealing with the values listed in Table 2.2.

Since the circuit is fully symmetric we can apply the bisection theorem [17]. For this analysis we will first compute the equivalent impedance seen in the source and drain of one of the input elements (i.e. M_1).

$$\begin{aligned} C_D &= C_{load} + M_1.cdb \\ G_D &= 1/R_{load} \\ C_S &= C_{ss} + M_{1\_tail}.cdb \\ G_S &= 1/R_{ss} + M_{1\_tail}.gds \\ g_m &= M_1.gm \end{aligned} \tag{3.4}$$

With this notation in mind we can proceed to model the amplifier transfer function. We assume the amplifier to be fairly linear in the operating region. We can treat it as an equivalent current source with source degeneration driving an equivalent drain load.

The drain load, with lumped model consisting of C_D and R_D in Eq.(3.4) acts as a frequency dependent impedance with value

$$Z_{drain} = \frac{R_D}{1 + sR_DC_D} \tag{3.5}$$

which is the parallel connection of equivalent capacitance and resistance in the small signal model.

This load is driven from a current source consisting of a transistor with source-degeneration, which transfer function we can model as a frequency dependent transconductance

$$Y_{source} = \frac{g_m}{1 + g_mR_S} \frac{1 + sR_SC_S}{1 + s\frac{R_SC_S}{1+g_mR_S}} \tag{3.6}$$

Combining these two, we can write the overall voltage transfer function of the linear receiver equalizer

$$\frac{v_{out}}{v_{in}}(s) = \frac{g_m R_D}{1 + g_m R_S} \frac{1 + sR_S C_S}{(1 + s\frac{R_S C_S}{1+g_m R_S})(1 + sR_D C_D)} \tag{3.7}$$

In this model we can make a couple of observations. The gain is the same as the gain of the source-degenerated common-source amplifier [17]. To account for frequency dependency we introduce one zero and two poles. One of the poles comes from the drain impedance, while the other pole and zero come from the equivalent impedance in the source of transistor M_1. We can also see that the zero is always at a lower frequency than the pole generated by the source-network, due to the degeneration term $1 + g_m R_S$. Finally, in order to ensure proper function of this block, as described in Chapter 2 we should make sure that the first pole also comes after the zero thus achieving peaking and equalization.

This requirement is, actually, already present in the system level formulation as a constraint for certain, minimal, eye opening that has to be achieved at a desired sampling phase. We do not need to add any system specific constraints in this circuit formulation as long as we provide information defined in Table 2.2.

Firstly, we will look into purely circuit side of this block. Here we will ensure proper biasing and certain operating range.

Since receiver equalizer is an analog circuit, and we want to be able to control the nonlinear distortion it introduces into the signal chain, we will constrain the input dynamic range of the circuit to keep it in acceptable operating conditions.

In order to limit the nonlinear distortion, we assume that only a certain percentage of the tail (bias) current can be steered into one of the input transistors. To do this, while taking advantage of the signomial quality models we instantiate three instances of the same input transistor (i.e. M_1) and assume different biasing currents, Eq. (3.8),

$$\begin{gathered} I_{HI} = 1.8 M_{1\_tail}.ids \\ I_{LO} = 0.2 M_{1\_tail}.ids \\ M_1\_hi.ids \leq I_{HI} \\ M_1\_lo.ids = 2M_1.ids - M_1\_hi.ids \end{gathered} \tag{3.8}$$

where we used instances M_1_hi and M_1_lo (in the actual code these are called M_3 and M_5) to represent input transistor (M_1) in different operating conditions when signal at its gate achieves maximum and minimum value. We assumed that we can, at most, switch between 20% and 80% of the total polarization current, which is $M_{1\_tail}.ids + M_{2\_tail}.ids = 2M_{1\_tail}.ids = 2M_1.ids$. This level was somewhat arbitrary, and ideally we would determine it from the maximal compression level acceptable for the operation of the amplifier. From this we obtain the necessarry information to formulate constraints for proper biasing, which assumes saturation region of operation for all transistors in the amplifier.

$$\begin{gathered}
M_{1\_tail}.vdsat + M_1\_hi.vgs \leq Vin\_min \\
M_{1\_tail}.vdsat + M_1\_lo.vgs \leq Vin\_max \\
\\
swing = M_1\_hiR_D \\
V_{dd} - swing + M_1\_hi.vdsat + M_1\_hi.vgs \geq Vin\_max \\
M_1\_hi.vgs - M_1\_lo.vgs > Vin\_max - Vin\_min \\
\\
Vin\_max > Vin\_cm > Vin\_min \\
\\
statP = 2V_{dd}M_{1\_tail}.ids = 2V_{dd}M_1.ids
\end{gathered} \tag{3.9}$$

At first glance, the first two relations in Eq. (3.9) might seem redundant. Basically, we do not expect problems with M_{1_tail}, Fig. 3-3, entering triode region when input signal is at its highest at the gate of M_1. However, in sub-micron technologies the trashed voltage is not a monotonic function [44]. Also, possible discontinuities in the model might interfere with proper calculation. Furthermore, we also constrain the dynamic range of the input voltage into the circuit to be less than difference of gate-source voltages of M_1_hi and M_1_lo with $M_1\_hi.vgs - M_1\_lo.vgs > Vin\_max - Vin\_min$. This ensures that the circuit will always be in the desired linearity region.

Since the optimization engine does not have a notion of 'reasonable design' and many implicit assumptions that human designers make are not introduced (in order to widen possible search space and make it easier to find a solution) we choose to stay on the safe side and introduce additional constraints that would help debugging, if need be. It is safe to assume that optimization engine will try to exploit any inaccuracy in model or constraint system and finding problems in optimization-based circuit design can be a challenging task as the amount of information from an infeasible result is very limited, and certainly much less than from the full-blown simulation.

Finally, we have to see how we can include filter model abstraction for this block into the system level model. The model we derived, Eq. (3.7), is an $\mathcal{S}$-domain abstraction of the analog behavior of the equalizer. As we are working in the sampled domain at the system level, we have to map this analog filter into digital filter.

There are multiple approaches to discretization of analog filters [34, 33], but there is no widely accepted method to implement this approximation. Here we will advocate pole-zero mapping [34], as it is very simple, or the bilinear transformation [33]. Furthermore, as the oversampling ratio of the processing compared to the Nyquist frequency of the system increases, the difference (and thus error of an approximation) between various approximations decreases. Since our system level formulation is very compact (see Chapter 2 for details) we can afford relatively high oversampling ratios.

To see how to discretize our analog filter, we start by looking into impulse invariant transform [33]. In this discretization method the aim is to accurately represent impulse response of the system when transitioning into digital domain. For the simplest case of one-pole system we have

$$H(s) = \frac{A}{s-p} \rightarrow H(z) = \frac{T_s A}{1 - e^{pT_s} z^{-1}} \tag{3.10}$$

where T_s is the sampling interval. A similar relation holds between transfer functions in $\mathcal{S}$- and $\mathcal{Z}$- domains if they are written in partial-fractions form [33]. Note that there is no simple mapping between zeros of analog and digital representation.

Thus, we can assume and use simple mapping

$$z_p = e^{T_s s_p} \tag{3.11}$$

for both poles and zeros of the transfer function. This discretization method is know as *pole/zero mapping* [34]. However, the mapping cannot be included in the formulation directly as signomial based solver cannot handle exponential function between variables. To overcome this one might use Taylor expansions of the exponential function or use bilinear transform.

Another popular possibility is using the bilinear (Tustin) transform given with

$$s = \frac{2}{T_s}\frac{z-1}{z+1} \tag{3.12}$$

With this in mind we can start transforming our analog model for the filter, Eq. (3.7), for digital representation

$$\frac{v_{out}}{v_{in}}(s) = \frac{g_m}{C_D}\frac{s + \frac{1}{R_S C_S}}{(s + \frac{1+g_m R_S}{R_S C_S})(s + \frac{1}{R_D C_D})} \tag{3.13}$$

and by substituting Tustin approximation, Eq. (3.12), into Eq. (3.13), after rewriting into standard form we obtain

$$\begin{aligned}
\frac{v_{out}}{v_{in}}(z) &= \frac{g_m}{C_D}\frac{T_s}{2}\frac{1 + \frac{\omega_z T_s}{2}}{(1 + \frac{\omega_{pd} T_s}{2})(1 + \frac{\omega_{ps} T_s}{2})}h(z) \\
h(z) &= (1 + z^{-1})\frac{1 - \frac{1 - \frac{\omega_z T_s}{2}}{1 + \frac{\omega_z T_s}{2}}z^{-1}}{(1 - \frac{1 - \frac{\omega_{pd} T_s}{2}}{1 + \frac{\omega_{pd} T_s}{2}}z^{-1})(1 - \frac{1 - \frac{\omega_{ps} T_s}{2}}{1 + \frac{\omega_{ps} T_s}{2}}z^{-1})} \\
\tau_S &= R_S C_S \\
\tau_D &= R_D C_D \\
\omega_z &= 1/\tau_S \\
\omega_{ps} &= (1 + g_m R_S)/\tau_S \\
\omega_{pd} &= 1/\tau_D
\end{aligned} \tag{3.14}$$

where R_S, R_D, C_S, C_D are exactly the same lumped parameters defined in Eq. (3.4), and T_s is the sampling period. This set of equations is also one of the best examples of the importance of handling signomial expressions if optimization is to be used in circuit design, and especially if the system level modeling is to be introduced at the same time. At this stage we can, finally, provide all the bindings, required by the system level formulation as noted in Table 2.2, for this block. Reading from Eq. (3.14) we obtain

$$\begin{aligned} scale &= \frac{g_m}{C_D}\frac{T_s}{2}\frac{1+\frac{\omega_z T_s}{2}}{(1+\frac{\omega_{pd}T_s}{2})(1+\frac{\omega_{ps}T_s}{2})} \\ zRX_1 &= \frac{1-\frac{\omega_z T_s}{2}}{1+\frac{\omega_z T_s}{2}} \\ dpRX_1 &= \frac{1-\frac{\omega_{pd} T_s}{2}}{1+\frac{\omega_{pd} T_s}{2}} \\ spRX_1 &= \frac{1-\frac{\omega_{ps} T_s}{2}}{1+\frac{\omega_{ps} T_s}{2}} \end{aligned} \tag{3.15}$$

This concludes the receiver equalizer description. Equations given in (3.8), (3.9) and (3.15) describe the receive equalizer reasonably well and up to the required accuracy to introduce it to the system level formulation described in Chapter 2.

One thing we need to take into account is the quality of the discretized model. It is known that the accuracy of Tustin's approximation is decreasing as the pole locations are approaching sampling frequency [33]. As we want a simple link between digital and analog domains we do not use the pre-warping [33]. Thus we have to limit, from above, the range of the frequencies our approximation is valid for. From the frequency warping we know that digital approximation acts at frequency ω the same way as the analog original acts at $(2/T_s)tan(\omega T_s/2)$ [33]. By comparing functions $tan(x)$ and x we can see that to estimate with less than 10% error we have to stay within $\omega T_s/2 \leq 0.3$. When we take into account that actual transformation between analog and digital poles is

$$\omega_D = \frac{1-\omega_A T_S/2}{1+\omega_A T_S/2} \tag{3.16}$$

we can see that our digital poles/zero will always have to be in the range from 0.53 up to 1. In our formulation we, actually, can not use the range near 1. To understand this we need to think in terms of optimization engine and its accuracy. As our problem is equivalent to finding $Y(z)$ from

$$(1 - spRX1z^{-1})(1 - dpRX1z^{-1})Y(z) = p(z) \tag{3.17}$$

as we have explained in Chapter 2. We can see, Eq. (3.17), that as the poles approach 1 we can start experiencing some scaling issues. This comes from the fact that $1-pz^{-1}$ decreases in magnitude significantly as $p \longrightarrow 1$. Under these circumstances we see great degradation in accuracy in the solution that comes from the solver as we are trying to equate two very small numbers. If either of them falls below the solver accuracy the other can be picked randomly within the accuracy bounds and solver will, formally, satisfy the constraint. To prevent this we limit ourselves to using digital poles up to 0.9 which was somewhat randomly chosen and probably could be extended. Thus, at the end, we have the range of 0.53 up to 0.9 at our disposal. Next we show that this range is sufficient and that constraining design to this range does not restrict the expected design range unreasonably.

Now the sampling frequency we will use in our formulation can be determined. The question is equivalent to determining the analog frequency range we wish to map into available digital frequency range. In digital communications the maximum frequency of signal is Nyquist's frequency, equal to half the data rate [36]. We chose to cover the frequencies up to f_{bit} as we wanted to give enough freedom to the optimization engine to push equalizer poles far enough for their influence to be small. (Of course, this would bring power number penalty, but we did not want to constrain the search space too much.) If we take this into the account we see that

$$0.5T_s(2\pi(2f_{bit})) \leq 0.3 \Rightarrow \frac{f_s}{f_{bit}} \geq 10 \tag{3.18}$$

and we pick $f_s/f_{bit} = 12$. Under these circumstances we see that the lowest frequency we can express in our top level formulation is

$$\frac{1-\omega_{low}T_S/2}{1+\omega_{low}T_S/2} = 0.9 \Rightarrow 2\pi\frac{f_{low}}{f_s} = 2\pi\frac{f_{low}}{12 f_{bit}} = 0.053 \tag{3.19}$$

which gives approximately $f_{low} = 0.1 f_{bit}$. In other words, if the circuit implementation of receiver can achieve both poles above Nyquist frequency, we would be able to see up to $20log(f_{Nyquist}/f_{low}) = 13.6dB$ of peaking in our formulation. This is much more than we see in circuits fabricated today. Thus our design space covers current real designs and leaves room for potential improvement.

The only missing part is the standard deviation of noise, requested in Table 2.2, as this part of the model is not fully developed in current version. Noise performance of the equalizer is of somewhat secondary importance due to the fact that the dominant error mechanism in HSL is intersymbol interference [43]. This point of view is fully justified if the eye aperture requested is much higher than typical $1e-15$ quantil for normal distribution, which is usually the case.

3.3 Slicer

The signal processing chain ends at the slicer input. Currently no circuit implementation is fully developed for this sub-system and only simple behavioral model is included. This simplified model has only two parameters included in Table 2.3.

Currently we assume certain equivalent noise at the input of the slicer, which is usually around $2mV$ RMS [43]. This parameter would be derived from the circuit implementation of the slicer.

Furthermore we assume certain equivalent overdrive needed for resolving the incoming symbol. This parameter combined with the typical residual offset of the latch is called *sensitivity* and defines the *level* of the signal needed for resolution.

Firstly the input amplifier stage of the slicer would introduce certain worst case offset. We have to make sure the incoming signal is unambiguous even if shifted for this amount each way. This value is usually less than $10mV$ for offset compensated latches and up to $50mV$ and more for those that are not. Secondly, the input signal (even in an ideal case without any input offset voltage) needs to provide enough driving for the positive feedback in the back end latch to completely resolve within one clock cycle. Due to the positive feedback in the slicer, the sensitivity can be found from the voltage necessary at the latch input, after the slicer. If that voltage is V_{out} then we must have $V_{out} = V_{sens}e^{T_{bit}/\tau}$ for certain time constant depending on the slicer circuit. Thus, $V_{sens} = V_{out}e^{-1/(\tau f_{bit})}$. This value is usually $20mV$, which is representative of the sub-100nm 10Gb/s samplers [43]. In most of our simulations we use $100mV$ for eye aperture, for illustration purposes, which justifies our noise model for receiver equalizer and leaves ample noise margin for other non-modeled interference (crosstalk) and noise sources (supply noise).

3.4 Summary

In this chapter we explained the circuit level models for the most important sub-systems in the signal chain. We, also, show that in the coding model we accepted each circuit is represented through internal constraints and interface variables.

We use internal constraints at the circuit level model for each block to ensure proper biasing, mode of operation, and introduce specific implementation constraints. We also introduce internal constraints to link interface variables in proper relation to circuit parameters. These constraints are never exposed to higher hierarchy levels. This is an attempt to decouple the implementation issues and abstract performance models as much as possible.

Interface variables are set at the system level. Ideally, the system level can be formulated independently of the circuit implementation. The purpose of this hierarchical code organization is to achieve high degree of modularity and enable different implementations to be quickly explored in our framework.

Each circuit sub-block must only export its interface variables for the system level model in order to be linked into the flow. This approach very naturally extends the traditional circuit-to-system designer interaction and defines very clear domain boundaries for easier code maintenance.

Chapter 4

Connecting the Circuit and System levels

In this chapter, we put together the whole problem formulation, linking the system and circuit level descriptions from Chap. 2 and Chap. 3. We show the structure of the overall formulation and the roles of various interface variables. Using the overall formulation we can generate a set of different designs for different sets of system and circuit specifications (for example by changing the signaling rate and assumed sampling position).

To the best of our knowledge, this is the first report where interaction between system and circuit level design decisions and specifications is explored simultaneously within an efficient and unified design flow.

4.1 Structure of the optimization code

In this section we will describe the optimization formulation developed as an example to demonstrate the joint system-circuit optimization approach. The objective is to provide an optimization formulation that jointly and interdependently searches for system and circuit parameters in order to achieve given specifications. In our high-speed link signal processing chain example, the optimization objective is to minimize power for a given channel and error rate.

In Fig. 4-1 we present the structure of the optimization formulation.

The formulation is highly modular. The actual division of code is made according to a possible division of the design within a group of designers. Furthermore, with this approach, each hierarchy level remains relatively simple, thus easy to maintain and modify.

In Fig. 4-1 each sub-formulation presents the internal variables and constraints (blue code) and interface variables (red code). We have described most of the internal functionality of every module in previous Chapters 2 and 3. The purpose of the *linker level* in Fig. 4-1 is to formally instantiate all the necessary sub-formulations and provide necessary bindings and interdependency constraints.

We note that transmit (TXeq) and receive (RXeq) equalizers are just the wrapper layers in which we express the system level parameters of the respective circuit models. The naming conventions are left identical as in the example code to provide easy match between our explanations here and the optimization formulation.

While purely technical aspect of design, we have to note that following good hierarchical design practice is not only possible but very natural in an equation-based optimization-driven design flow.

Finally, we should note the existence of two different formulations with *level* suffix, *PRElevel and TOPlevel.* The names are, once again, legacy of our code development phase. We now explain the purpose of these two modules.

The *TOPlevel* module is exactly the system level formulation we described in the Chapter 2. It uses the system level parameters of the design (input swing into the channel, channel impulse response, filtering coefficients at the transmitter, receiver transfer function, ...) to express the eye aperture at a certain sampling phase. This information we combine with BER specification and slicer parameters, Table 2.3, to form a constraint that would ensure sufficiently large sample to achieve given performance. All the details of this formulation we discuss, as we mentioned, in Chapter 2.

The *PRElevel* module is very similar to the *TOPlevel.* To see the purpose of it, we refer to Fig. 4-1. In order to ensure proper biasing of the receive peaking amplifier

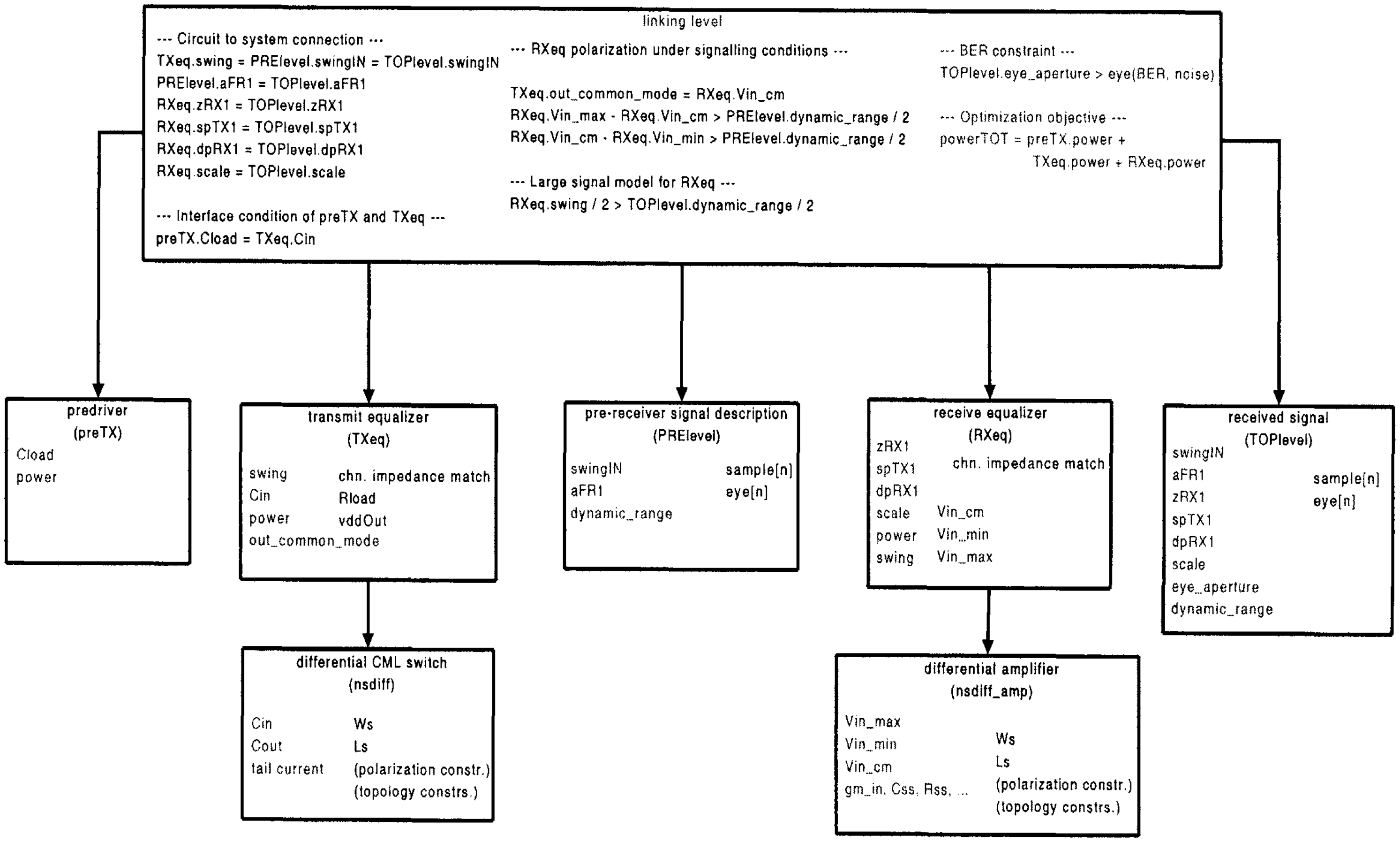

linking level
--- Circuit to system connection ---
TXeq.swing = PRElevel.swingIN = TOPlevel.swingIN
PRElevel.aFR1 = TOPlevel.aFR1
RXeq.zRX1 = TOPlevel.zRX1
RXeq.spTX1 = TOPlevel.spTX1
RXeq.dpRX1 = TOPlevel.dpRX1
RXeq.scale = TOPlevel.scale
--- Interface condition of preTX and TXeq ---
preTX.Cload = TXeq.Cin
--- RXeq polarization under signalling conditions ---
TXeq.out_common_mode = RXeq.Vin_cm
RXeq.Vin_max - RXeq.Vin_cm > PRElevel.dynamic_range / 2
RXeq.Vin_cm - RXeq.Vin_min > PRElevel.dynamic_range / 2
--- Large signal model for RXeq ---
RXeq.swing / 2 > TOPlevel.dynamic_range / 2
--- BER constraint ---
TOPlevel.eye_aperture > eye(BER, noise)
--- Optimization objective ---
powerTOT = preTX.power + TXeq.power + RXeq.power
predriver (preTX)
Cload
power
transmit equalizer (TXeq)
swing
Cin
power
out_common_mode
chn. impedance match
Rload
vddOut
pre-receiver signal description (PRElevel)
swingIN
aFR1
dynamic_range
sample[n]
eye[n]
receive equalizer (RXeq)
zRX1
spTX1
dpRX1
scale
power
swing
chn. impedance match
Vin_cm
Vin_min
Vin_max
received signal (TOPlevel)
swingIN
aFR1
zRX1
spTX1
dpRX1
scale
eye_aperture
dynamic_range
sample[n]
eye[n]
differential CML switch (nsdiff)
Cin
Cout
tail current
Ws
Ls
(polarization constr.)
(topology constrs.)
differential amplifier (nsdiff_amp)
Vin_max
Vin_min
Vin_cm
gm_in, Css, Rss, ...
Ws
Ls
(polarization constr.)
(topology constrs.)

Figure 4-1: Code hierarchy and linking: transmit pre-driver, transmit equalizer and receiver equalizer circuit formulations and signal description formulations before and after the receive equalizer can be seen. Red colored variables are exported to appropriate higher hierarchy level, while blue ones are selected internal variables. Linking level connects the appropriate interface variables.

we need to know the operating conditions and the properties of the input signal after the channel at the equalizer input. For example, we need to include the effect of transmit-side filtering on the dynamic range of the received signal, to take maximum advantage of the receiver design. To do so, we generate a 'top' level description that takes into account only the filtering before the receiver (transmit equalization and channel), thus obtaining the description of the eye diagram just in front of the receiver equalizer. We use the information contained in the unwrapped eye at this point to properly bias the receiver amplifier, as we explained in Chapter 2.

Both modules are easy to generate using the scripts that we developed. They are explained in more detail in Appendix A. Now we explain the process.

While Fig. 4-1 represents one possible organization any reasonable hierarchy structure should work as long as it satisfies the needs of the designers in expressing the necessary interactions. For example, in our code structure in Fig. 4-1 the pre-driver (preTX) and transmit equalizer (TXeq) could be instantiated into transmitter abstraction that would be introduced on the top level, thus decoupling the actual structure of the transmitter from its representation at the system level.

4.2 Optimization results

In this section we illustrate the potential of the developed design-space exploration flow. Using the techniques described, we analyze designs obtained for different data rates in terms of system and circuit performance and resource allocation. Our main goal is minimization of the total system power for a given performance requirement (eg. eye aperture). As we have already noted, we analyze only the signal processing chain model with appropriate circuit implementation models.

For these desing-space explorations to be conducted we use the channel presented in Chapter 2, Fig. 2-1(b). The data rates we consider are 4Gb/s, 6.25Gb/s, 8Gb/s and 10Gb/s, with 10dB, 20dB, 29dB and 33dB of attenuation at the appropriate Nyquist frequency, respectively.

For each signaling frequency, we will minimize the system power, under certain eye aperture constraint at different sampling phases relative to the main tap of the unequalized pulse response.

In all the optimization runs we use Predictive 90nm Technology node transistor models [45].

4.2.1 4Gbps without DFE

To obtain the results we are about to present here, a 4Gbps binary signaling is assumed in the formulation. The channel attenuation at the Nyquist frequency is, as we mentioned before, around 10dB. Pulse response of the channel at 4Gb/s, oversampled 12 times above the symbol rate is given in Fig. 4-2. We use sample numbers of the unequalized pulse response to indicate the sampling phase.

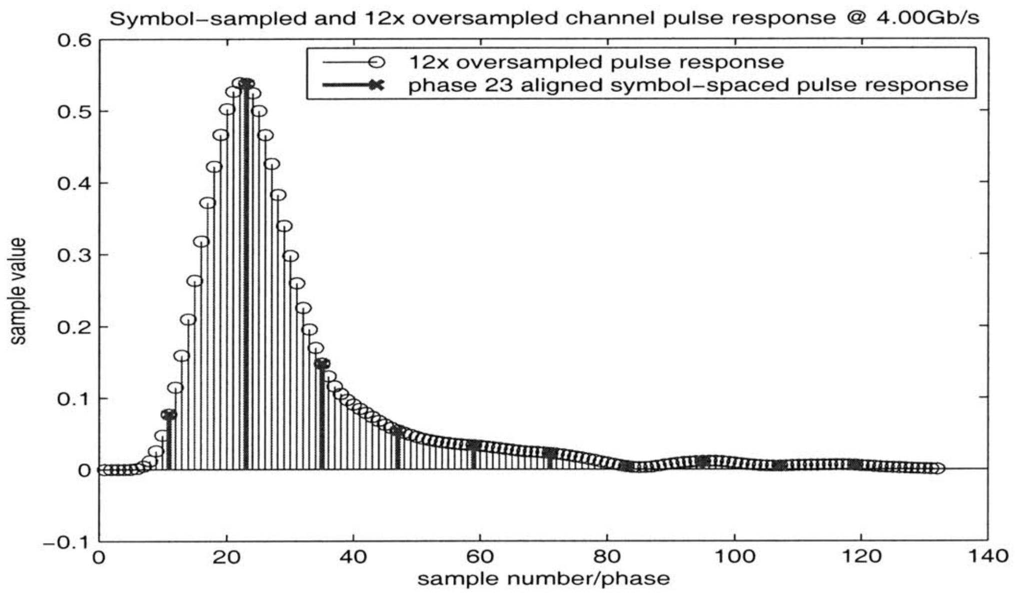


Figure 4-2: Pulse response @ 4Gbps (i.e. the channel response to a 250ps-wide unit pulse)

The full system-circuit formulation, explained in the beginning of this chapter, was run with different sampling phase assumptions. The objective is power minimization, under constraint that eye aperture has to be at least 100mV, at the chosen sampling phase.

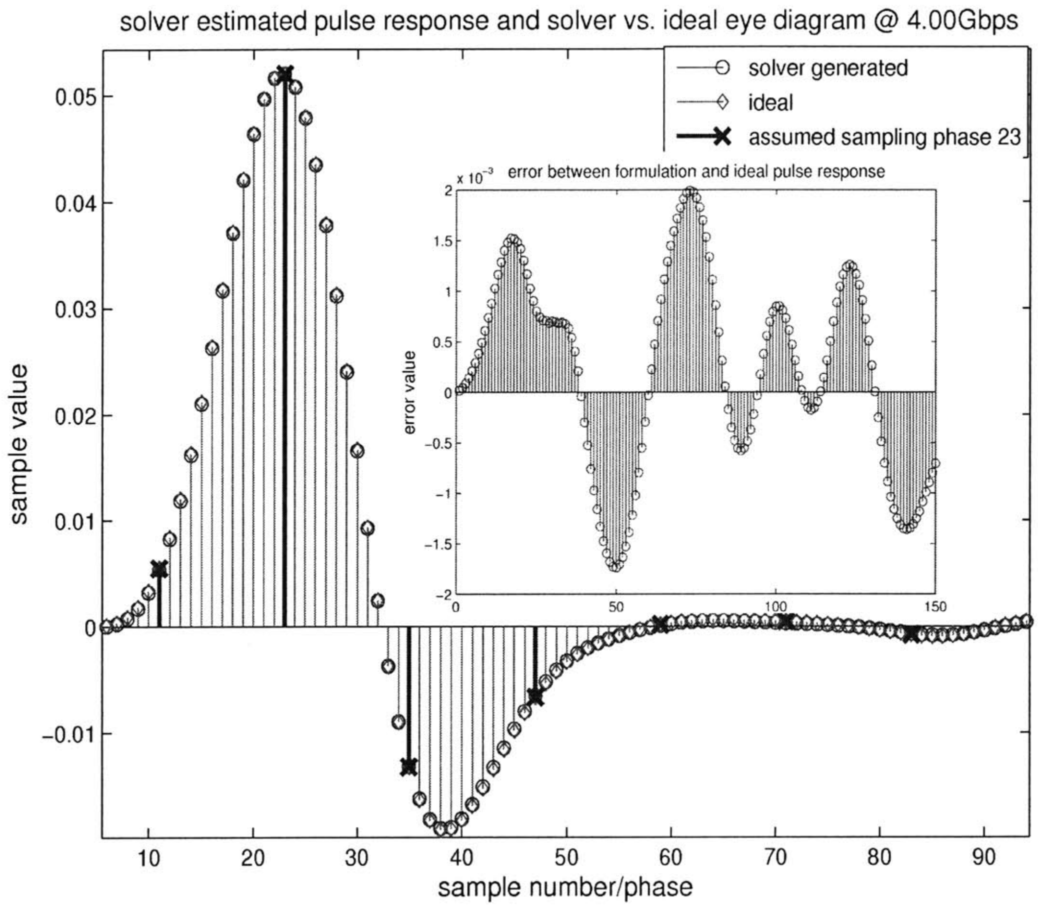


Figure 4-3: Comparison of the solver generated and ideal equalized pulse response for a given system level parameters

Finally, we can confirm the quality of the system level formulation we described in Chapter 2 by comparing the ideal, Matlab generated, equalized response for the given set of system level parameters (*aFR, zRX, spRX, dpRX, swingIN, ...*) with the solver provided values of the equalized impulse response under same set of parameters. This comparison is presented in Fig. 4-3, for sampling phase 23 and appropriate system level parameters given in Table 4.1. We can observe excellent matching between two waveforms.

It is interesting to note, Table 4.1, how equalization in the receiver changes with respect to the choice of main tap. We note that attempt to push the required eye opening value before the main tap phase of the unequalized response, Fig. 4-2, results in large peaking in receiver, and could increase the overall system power unreasonably. The obtained trade-off between sampling phase and signal processing chain power is given in Fig. 4-4. More detailed view is given in Table 4.1. This trade-off is obtained by iterating the optimization runs for different target sampling phase. This is intuitively clear as moving the optimal sampling position to the left compared to the unequalized main tap position indicates strong differential action of equalizers and receiver equalizer is well suited for this. On the other hand, we see that under signaling conditions we imposed, and for the given channel, much better result can be obtained by using the receiver equalizer only as a wideband amplifier without any frequency selectivity/peaking (if a proper target timing is chosen).

To get more insight into transmit and receive equalizer interaction we should pay attention to the last row of the Table 4.1. In this run we used the same, optimal,

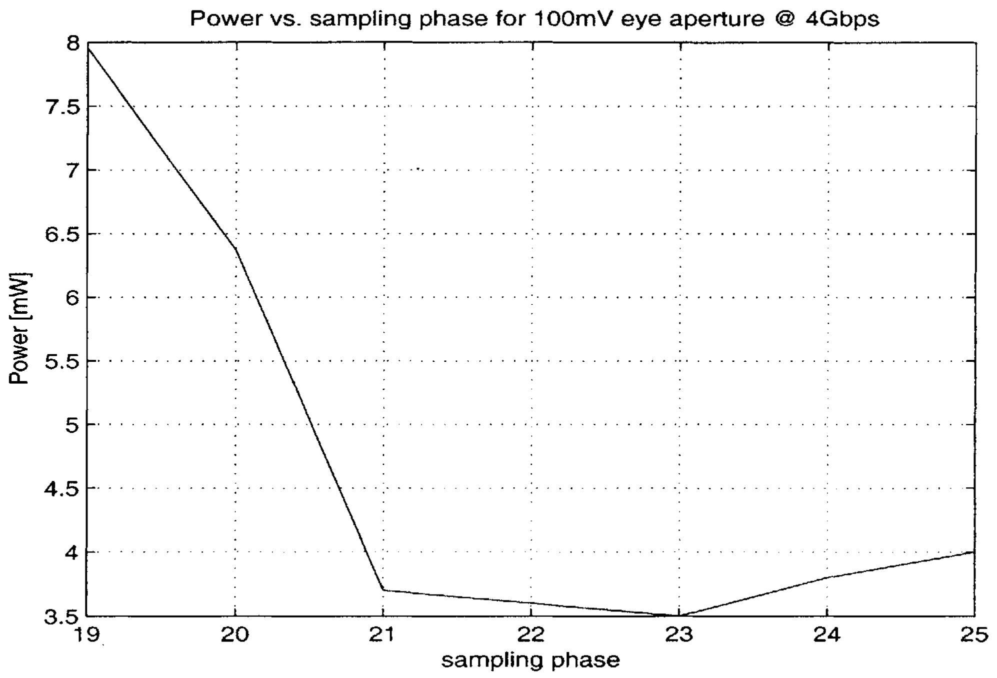


Figure 4-4: Model power change as a function of sampling phase

settings as for the phase 23, but we forced transmit equalizer to act only as a buffer by putting $aFR1 = 0$. We can observe that resulting power is significantly higher than the power needed when transmit equalizer is being optimized. The reason for inefficiency of receive equalizer is that in capacitively degenerated differential amplifier peaking is achieved by decreasing the DC gain. We can can see that as the receiver equalizes the channel more, the DC gain of the receiver drops from 3.5 to 1.1. This, on the other hand impacts the transmit equalizer as the necessary swing of the signal transmitted into channel has to be increased to compensate for this loss of DC gain in the receiver, resulting in higher transmit equalizer current and ultimately in significant increase in power.

While circuit level models we use are relatively simple, and could be improved, this analysis proposes wideband receiver amplifier as an interesting alternative to solutions with peaking receiver that were employed so far. To the best of our knowledge, this work is the first one to analyze such system level interaction and its implications to the circuit level implementation in an unified and efficient manner.

Table 4.1: Design parameters for different sampling phases at 4Gbps

phase	*power* *[mW]*	*powerPRE* *[mW]*	*powerTX* *[mW]*	*swingTX* *[mV]*	*vddTX* *[mV]*	*aFR*	*powerRX* *[mW]*	*DCgainRX*	*zeroRX* *[Hz]*	*SpoleRX* *[Hz]*	*DpoleRX* *[Hz]*
19	*7.96*	*0.64*	*6.35*	*255*	*600*	*[1 -0.40]*	*0.97*	*1.2*	*790M*	*3.5G*	*4G*
20	*6.37*	*0.75*	*4.9*	*210*	*570*	*[1 -0.33]*	*0.72*	*1.3*	*840M*	*3.3G*	*4G*
21	*3.7*	*0.70*	*2.8*	*100*	*650*	*[1 -0.45]*	*0.25*	*3.5*	*2.5G*	*2.5G*	*3.7G*
22	*3.6*	*0.56*	*2.7*	*100*	*640*	*[1 -0.42]*	*0.25*	*3.0*	*2.2G*	*2.7G*	*3.7G*
23	*3.48*	*0.55*	*2.7*	*100*	*630*	*[1 -0.41]*	*0.25*	*3.5*	*2.3G*	*2.44G*	*3.4G*
24	*3.8*	*0.77*	*2.7*	*101*	*620*	*[1 -0.33]*	*0.23*	*3.8*	*2.2G*	*2.2G*	*3.4G*
25	*4.0*	*0.69*	*2.7*	*100*	*600*	*[1 -0.30]*	*0.48*	*4.9*	*2.4G*	*2.4G*	*3.8G*
23	*5.56*	*0.8*	*4.66*	*200*	*580*	*none*	*1.04*	*1.1*	*800M*	*2.6G*	*2.1G*

4.2.2 6.25Gbps without DFE

The channel attenuation at the Nyquist frequency is, as we mentioned before, around 20dB. Pulse response of the channel at 6.25Gb/s, oversampled 12 times above the symbol rate is given in Fig. 4-5. We use sample numbers of the unequalized impulse response to indicate sampling phase.

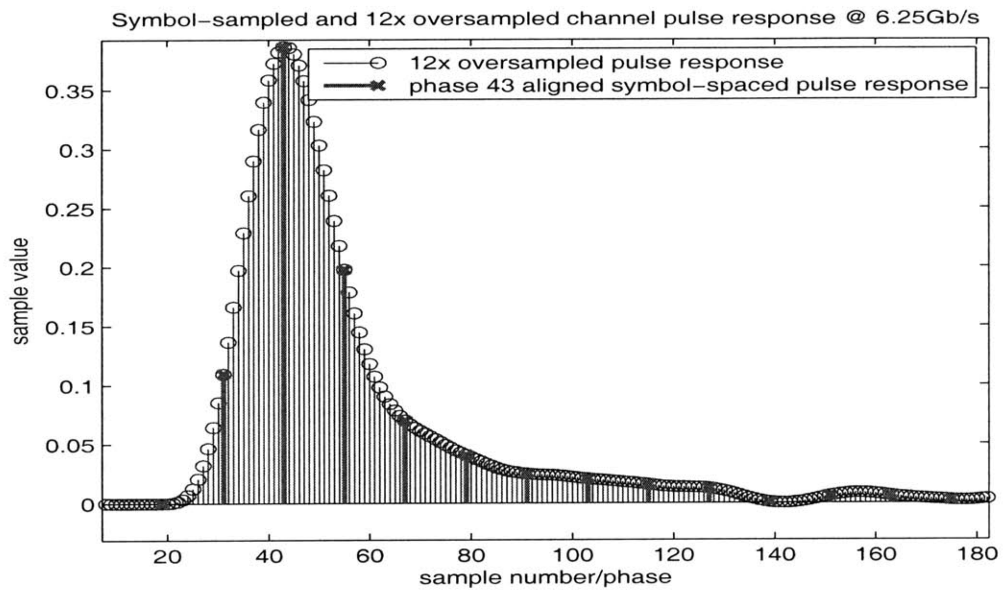


Figure 4-5: Pulse response @ 6.25Gbps

The full formulation, explained in the beginning of this chapter, was run with different sampling phase assumptions. The objective is power minimization, under constraint that eye aperture has to be at least 100mV, at the chosen sampling phase.

We can confirm the quality of the system level formulation we described in Chapter 2 by comparing the ideal, Matlab generated, equalized response for the given set of system level parameters (*aFR, zRX, spRX, dpRX, swingIN, ...*) with the solver provided values of the equalized impulse response under the same set of parameters. This comparison is presented in Fig. 4-6, for the set of parameters given in Table 4.2 under phase 43.

The obtained trade-off between sampling phase and signal processing chain power is given in Fig. 4-7. More detailed report is given in Table 4.2.

It is, also interesting to note, Table 4.2, how equalization in the receiver changes with respect to the choice of main tap phase. We note that attempt to push the required eye opening value before the main tap of the unequalized response, Fig. 4-2, results in large peaking in receiver. While total system power, at phase 38, is still dominated by the transmitter power, this configuration seems to be suboptimal. This is intuitively clear as moving the optimal sampling position to the left compared to the unequalized main tap position indicates strong differential action of equalizers and receiver equalizer is well suited for this. On the other hand, we see that under signaling conditions we imposed, and for the given channel, much better result can be obtained by using the receiver equalizer only as a wideband amplifier without any

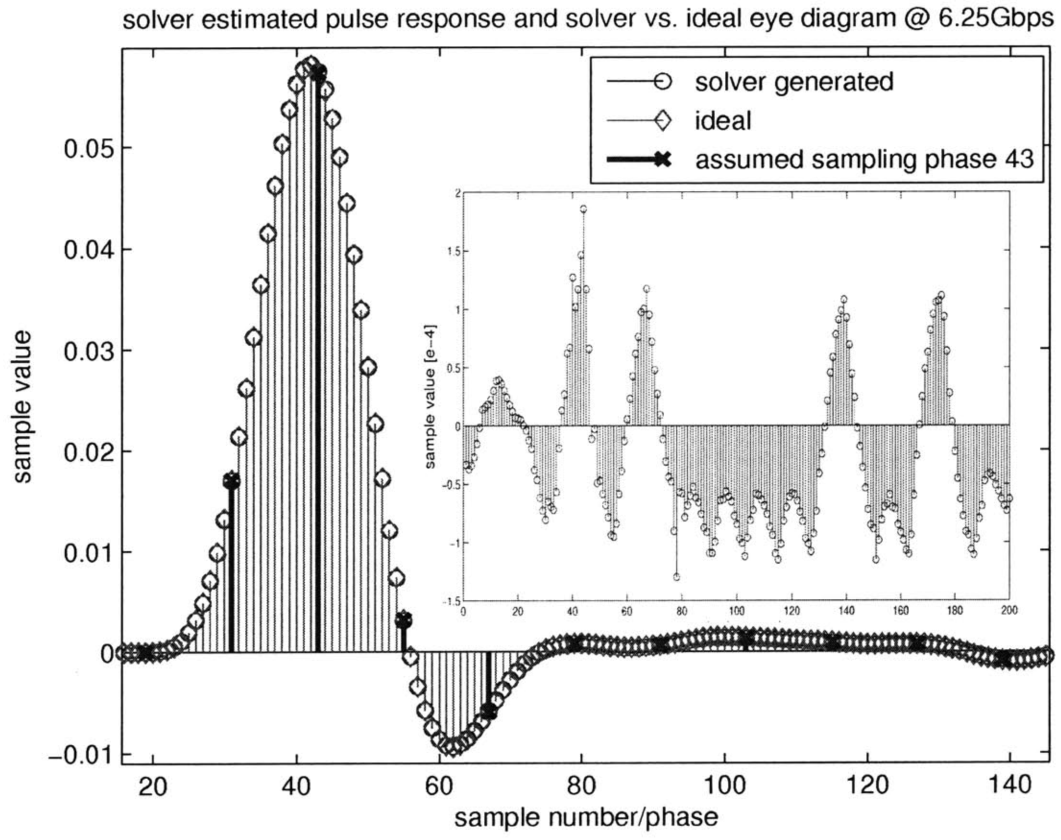


Figure 4-6: Comparison of the solver generated and ideal equalized impulse response for a given system level parameters @ 6.25Gbps

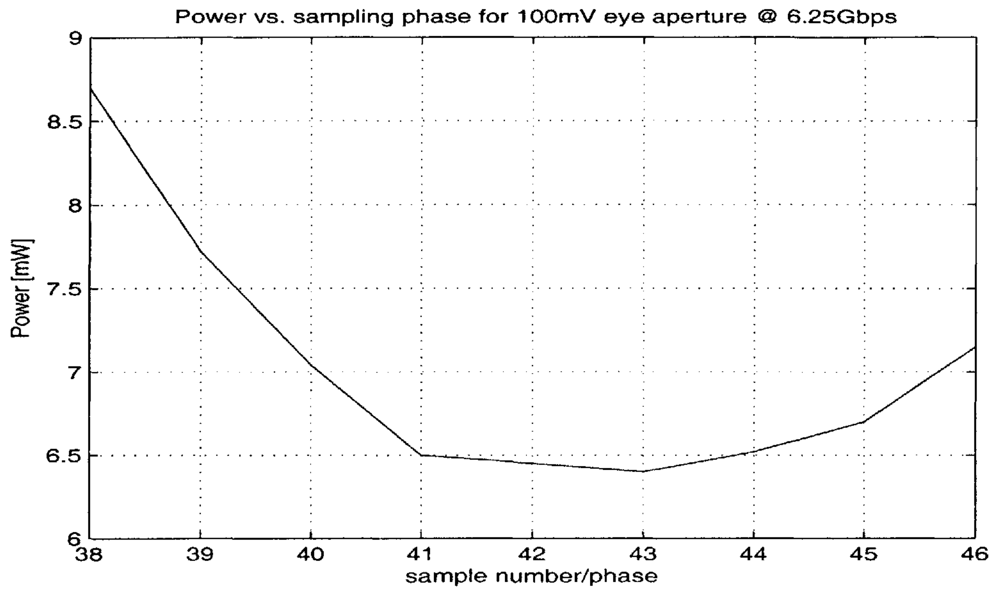


Figure 4-7: Model power change as a function of sampling phase

frequency selectivity/peaking.

Furthermore, at certain phases (after phase 41) the peaking in the receiver is observed past the Nyquist frequency of signaling. This is not very intuitive, and might be a consequence of numerical accuracy of the solver. Namely, the gradient of the objective (power) might not be very large in the neighborhood of a solution and solver decides that with certain accuracy of the gradient, it is near the real solution. We can note that the receiver equalizer power is also masked with the transmitter power which is an order of magnitude larger, for our model.

Finally, we should note that the same experiment, as in previous case, without transmit side equalization is infeasible for this signaling speed.

In our circuit model we include certain fixed loading of the receiver equalizer, and increase the capacitive loading depending on the output capacitance of the differential pair in the receiver equalizer. For results presented in this subsection total capacitive load at the output of the receive equalizer is approximately $22fF$.

Table 4.2: Design parameters for different sampling phases at 6.25Gbps

phase	*power* [mW]	*powerPRE* [mW]	*powerTX* [mW]	*swingTX* [mV]	*vddTX* [mV]	*aFR*	*powerRX* [mW]	*DCgainRX*	*zeroRX* [Hz]	*SpoleRX* [Hz]	*DpoleRX* [Hz]
38	8.7	0.98	6.86	262	630	[1 -0.50]	0.92	1.1	1.3G	5.1G	6G
39	7.72	1.1	5.53	222	590	[1 -0.63]	1.1	1.8	2.5G	6.1G	6.2G
40	7.00	1.2	5.01	200	570	[1 -0.62]	0.74	2.1	3.2G	6.1G	6.1G
41	6.5	1.25	4.85	200	565	[1 -0.61]	0.43	2.3	2.9G	4.3G	5.6G
42	6.45	1.25	4.85	200	560	[1 -0.58]	0.35	2.6	5.2G	5.9G	5.9G
43	6.4	1.25	4.8	201	555	[1 -0.52]	0.35	2.7	4.7G	5.7G	5.8G
44	6.52	1.22	5.1	200	572	[1 -0.52]	0.35	2.8	5.7G	5.9G	5.8G
45	6.7	1.25	4.9	200	560	[1 -0.44]	0.6	3.6	5.6G	5.9G	5.9G
46	7.15	1.16	5.12	200	575	[1 -0.43]	0.88	4.5	5.6G	5.6G	6.0G

4.2.3 8Gbps with system-level 1-tap DFE correction

At 8Gb/s signaling, channel attenuation at Nyquist frequency is around 29dB. The unequalized pulse response of the channel is shown in Fig. 4-8.

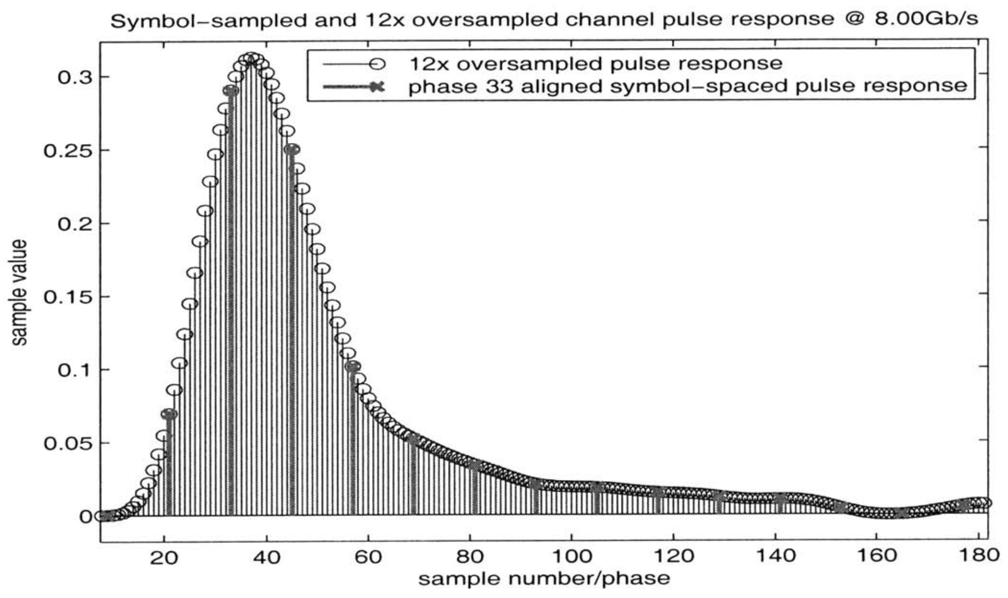


Figure 4-8: Pulse response @ 8Gbps

At this speed we could not obtain a feasible design from our initial formulation with only transmit and receive equalizers. Optimization runs at all the sampling phases returned eye opening less then the one required, thus indicating that such performance is not achievable under this signaling speed.

To overcome this problem, designers use DFE structure in the signal chain. We do not have a circuit model for slicer/latch and feedback implementation yet. However, we can test our flow by assuming, at the system level, that DFE is present. This requires only simple correction of the optimization formulation and is easily done. Namely, DFE cancels the influence of certain number of postcursors and this is easily accomplished at the system level by removing certain number of samples after the main tap in Eq. (2.10), thus neglecting their influence on the eye. In this case we assumed a 1-tap DFE, canceling only one postcursor.

Of course, we should remember that power number will be too optimistic as we do not have appropriate circuit model for a DFE implementation.

The results of the optimal run, at the phase 33, for this signaling speed (under assumptions above) are

- Total system power: power = 10.2 mW
- Predriver: powerPRE = 1.2 mW
- Transmit equalizer: (powerTX, swingTX, vddTX) = (7.54 mW, 284 mV, 633 mV)
- Transmit equalizer taps: aFR = [1 -0.43 -0.05]
- Receiver equalizer: (powerRX, DCgainRX, zeroRX, spoleRX, dpoleRX) = (2.15 mW, 2.1, 2.2 GHz, 3.6 GHz, 10 GHz)

It is very interesting to compare these results to the results in Tables 4.1 and 4.2. We should keep in mind that this power number does not account for DFE circuit implementation. As we can see the transmitter power is still dominant, but this time not by an order of magnitude, and is around 3.5 times greater than the receiver amplifier power. We can see that the solver attempted to increase the gain by increasing current, in order to preserve the bandwidth for such high speed operation. In previous examples (4Gb/s and 6.25Gb/s) the gain was mostly achieved through increased load resistance in the receiver.

To gain more insight in functionality of our formulation and confidence in the solver technology we show equalized pulse response and eye diagram estimates in Fig. 4-9.

As we can see the eye, after receiver equalizer is barely open. In this example we run formulation at the phase 33, where both ideal and solver estimated eye diagrams peak, Fig. 4-9. Due to $12x$ oversampling, the first postcursor in this picture is at the phase 45 and is around $25mV$. Thus, differential eye sample at this sampling phase, after DFE canceling the first postcursor, is $2 \times 25mV = 50mV$ which means that the aperture is $2 \times 50mV = 100mV$, as requested.

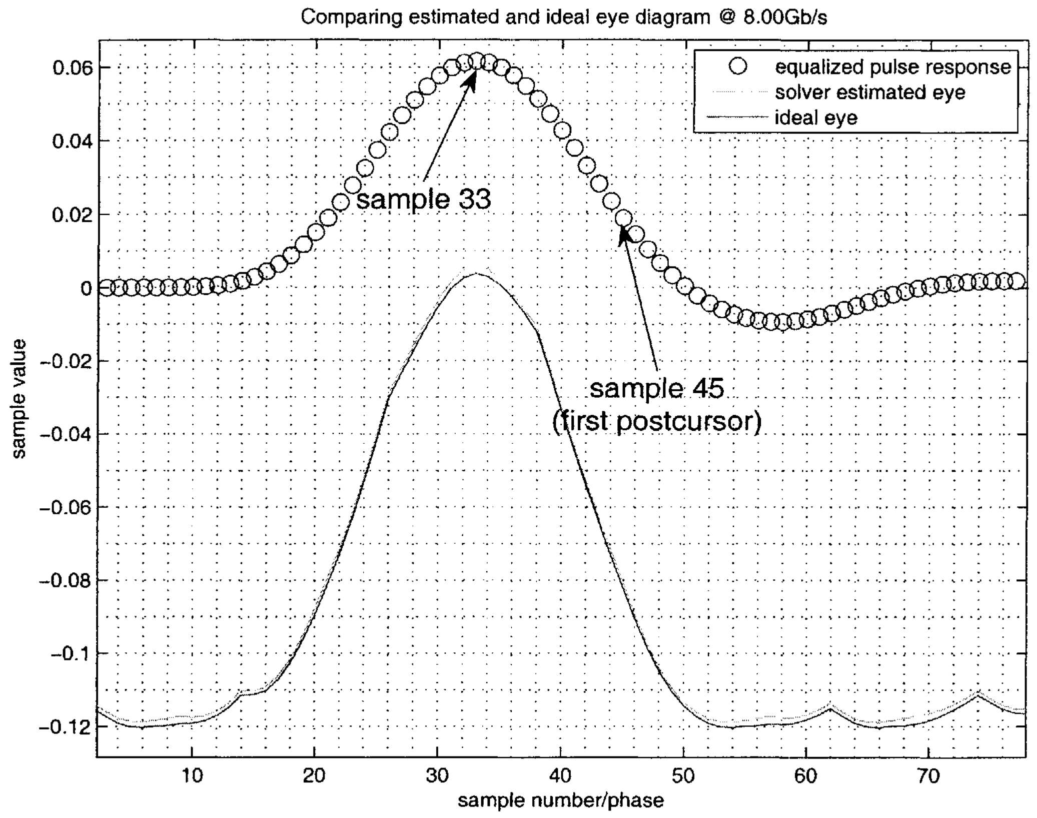


Figure 4-9: Comparison of the solver generated and ideal equalized impulse response and eye diagrams for a given system level parameters @ 8Gbps

It is interesting to note that optimal sampling phase was very close to the maximum sample in the impulse response for 4.00Gb/s and 6.25Gb/s. On the other hand, the optimal sampling phase moved to the left (of the maximum pulse response sample) in our tests at 8.00Gb/s.

This can be explained as an impact of the DFE at the system level. Namely, without the DFE transmit and receiver equalizer must take into account all the samples of the pulse response, and perform equalization in the best possible way. With DFE assumption, we take out the most significant interference sources for the given channel, several most significant postcursors, and the problem of equalization becomes more flexible because of the slack obtained. More accurately, in DFE case the first uncanceled postcursor is significantly smaller than the main tap, and more importantly it is on the tail of the impulse response. This means that moving the 'main' tap to the left increases the first uncanceled postcursor only slightly. We can also note that the main tap is decreasing as well. However, in our situation, the precursors are decreasing faster than the main tap and this can be exploited by moving the sampling phase to the left to obtain better equalization and eye opening.

4.2.4 10Gbps with system-level 4-tap DFE correction

In our final test we run the formulation for 10Gb/s signaling. In this case we work with 33dB of attenuation at the Nyquist frequency of signaling, as we can see in Fig. 2-1(b). The pulse response for this case is given in the Fig. 4-10.

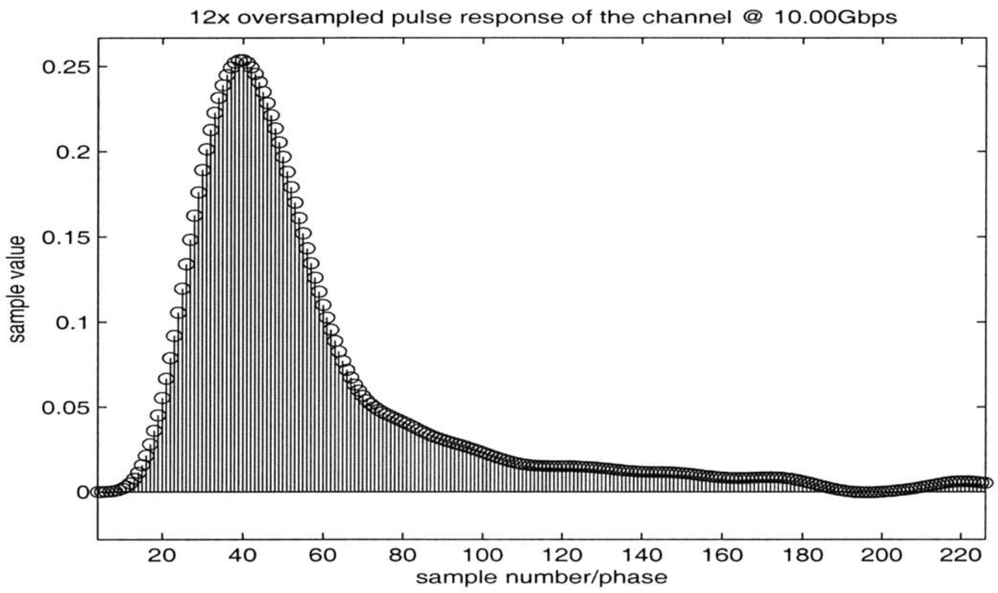


Figure 4-10: Pulse response @ 10Gbps

As we could have expected from the previous example, the formulation used for 4Gb/s and 6.25Gb/s would not find a feasible point. Thus, we decided to include DFE assumption, as for the 8Gb/s case, in this experiment as well. For this example, a 4-tap DFE, canceling first four postcursors, is eventually used.

The results of the optimal run, at the phase 35, for this signaling speed (under assumptions above) are

- Total system power: power = 19.4 mW
- Predriver: powerPRE = 1.16 mW
- Transmit equalizer: (powerTX, swingTX, vddTX) = (15.1 mW, 330 mV, 1.1 V)
- Transmit equalizer taps: aFR = [1 -0.40 -0.15]
- Receiver equalizer: (powerRX, DCgainRX, zeroRX, spoleRX, dpoleRX) = (3 mW, 3.3, 3.4 GHz, 3.6 GHz, 3.3 GHz)

in which case we could not achieve more than $70mV$ of eye aperture. Sensitivity analysis of the solution and additional optimization runs can be used to pinpoint the problem. In this case, it is our constraint on the pole created at the transmit equalizer output. In order to neglect the impact of this pole to the system, and thus simplify the formulation, we have included a constraint that requires this pole to be placed at a frequency, at least, four times greater than the Nyquist frequency. This is, in a way, along the same lines as the impedance matching. As the output resistance is fixed, this constraint actually limits the maximum output capacitance of the transmit equalizer, or equivalently the transistor size of the switches. As the predriver has fixed power supply, we can conclude that this constraint ultimatelly limits the maximum current through the switch (as we have bounded gate-source voltage and limited width of transistor), which is directly related to the achieved swing at the input to the channel.

4.2.5 Efficiency analysis

In previous, result, subsections we have demonstrated some of the potential of our design flow. In this process we have, also, presented evidence of very high accuracy of our system level formulation, through exceptional matching of ideal, Matlab generated, and solver determined equalized pulse response. Finally, in order to give a complete performance report of the developed flow we will include some comments on duration of each optimization run.

The size of the whole formulation is dominated, in terms of number of different optimization variables and optimization constraints, by the system formulation due to the fact that every sample of the equalized pulse response requires multiple optimization variables to be fully described and large number of samples we have to keep track of. Thus the parsing and solver time are mostly influenced by the size of the system-level formulation. Also, the circuit formulations we use in this example are relatively simple and information presented here might slightly change when more accurate and sophisticated circuit models are considered.

In general, as the data rate increases we have to keep track of higher number of pulse response samples as more of them become significant sources of ISI. For results presented in previous sections full formulations (including both circuit and system as described in Fig. 4-1) had between 1500 and 1700 variables and between 4500 and 5500 constraints. Most of these, as we explained, are introduced by the top level formulation.

In terms of run time, developed flow is very efficient. Our optimization runs were executed on a system with 1.60GHz Pentium M processor and 2GB of memory, running Debian Linux. Parsing time, as a function of the optimization problem size, varies between 15 minutes and 1 hour, when all unnecessary equations are removed from the formulation as we explain in Appendix A. The solver time is between 1 and 3 hours. Thus, the overall solution time is less than 4 hours for any configuration. While this is, in great part, consequence of the reduced size model obtained from the circuit and system designer, it can without doubt be considered a very efficient tool for design space exploration and porting/re-design.

4.3 Summary

In this Chapter we described how we can link the system model, described in Chapter 2 and circuit models from Chapter 3, thus obtaining an unified optimization model for circuit and system design. We have seen how operating conditions of circuits in the signal processing chain can be estimated from the properties of the signal before and after the circuit block.

Using the obtained formulation we show how different assumptions in the system and circuit models influence the resulting desing. We primarily focused on the system power as a function of sampling position. By exploring this dependency we found alternative transmit/receive equalizers settings rarely reported before. In this configuration power is reduced by using the receive equalizer as a wideband amplifier without any frequency selectivity in the Nyquist range.

We also demonstrated system level model accuracy, effectively checking our assumptions and solver performance for the problem described in Chapter 2.

Finally, we discussed efficiency of the proposed design flow.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

Integrated circuit design is becoming increasingly difficult with each consecutive technology node.

On one hand, physical dimensions of MOS transistor are harder to control accurately resulting in highly variable electrical parameters of transistor in deep submicron technologies. This is affecting circuit design greatly as operating conditions of circuits are harder to predict and design robustly. To compensate for this designers are usually forced to design more conservatively, usually sacrificing power and area. As digital circuits are less sensitive to these problems, analog frontends are becoming increasingly expensive in terms of system resources allocated to them.

On the other hand, systems are becoming increasingly complex in attempt to utilize advantage in growing number of active elements available per unit area. More advanced system level concepts are utilized in attempt to compensate for circuit performance and to provide steady increase in performance. Thus, every new generation of chips is bringing increased system complexity and performance while working with less predictable and less robust transistor technology.

This increased difficulty in design of mixed IC is observed both in time needed to design each consecutive generation of chips and its price. It takes significantly more attempts to match up the expectations of system designers and the performance that circuit designers manage to achieve.

In order to ameliorate the problem in circuit design phase, many different techniques for circuit design automation are proposed. In general, equation-based approaches are considered to be fast, but significant amount of work required on the part of a designer is considered to be a problem. At the same time, the simulation-based methods are easy to set up and are considered accurate, but exhibit very poor scaling with number of transistors due to overwhelming amount of data calculated in each iteration of an optimization algorithm, and can be very time consuming.

We decide to build on top of signomial optimization-driven circuit design work reported earlier [9, 23, 22, 12, 14, 13, 11], due to the modeling advantage this approach offers, necessary in circuit and system design. Furthermore, we see the involvement of circuit and system designers as an advantage of this approach; they provide effective model size reduction by approximations developed through experience and intuition.

In order to do so, we developed a robust and accurate linear system level model that fits the same signomial template used in circuit optimization. Through series of transformations we explored possibilities of a signomial solver and performance with different types of system level formulation. Through this process we arrived to a simple and easy to generate system formulation that does not suffer from numerical accuracy problems and excessive filesize. Due to simplified structure we managed to obtain through transformations, the formulation can be parsed and solved efficiently and reliably.

As system level formulation is convoluted due to many samples of pulse response we have to keep track of, each requiring multiple optimization variables to be properly described we also designed a script that is easy to configure and outputs the symbolic system level model we can link with circuit models to complete the formulation.

Finally, we put our system-and-circuit optimization compliant model of an HSL signal processing chain to a test. We provide, to the best of our knowledge, first report about interaction between transmit and receiver equalizers that takes into account both system and circuit levels of design. Furthermore, we have shown the efficiency of our proposed design flow by noting the run time of each optimization is on the order of hours.

Our proposed flow features simple and intuitive hierarchical and highly modular design/code structure. It is easy and efficient to use, opening possibilities for fast design space scouting in porting or redesign. Accuracy of the system description has been demonstrated, to complete previous reports about good accuracy of optimization-based circuit design. All this makes our optimization-driven system-to-circuit design flow a promising alternative to the traditional iterations between circuit and system engineers, in early stages of design.

5.2 Future work

This work demonstrates that efficient optimization-driven design flow can be extended to include both circuit models and system level model, and provide solution to this joint problem. It deals mostly with design procedure using a particular example of a HSL, and as such can be extended in many different directions.

Most generally, after successful demonstration on such system as a HSL with strong interaction between sub-systems and large number of parameters to keep track of, it is reasonable to attempt using this approach for other types of systems such as, for example, wireless links.

More particularly, our HSL example can be extended in many ways.

Firstly, signal processing chain could be improved by introducing proper circuit models of transmitter pre-driver, slicer/latch and DFE. The model would also benefit from better noise models.

Secondly, the clock and data recovery sub-system in the receiver and a PLL in transmitter could be modeled. This path would subsequently enable modeling of the timing noise effects and performing more sophisticated analysis of the whole link and its sub-system interactions.

Finally, once reasonably accurate system description is obtained, multiple corner/scenario optimization can be used to achieve robust designs that would be tolerant to system and circuit uncertainties which are one of the biggest difficulties in achieving high yield robust designs. In connection with this, by generating multiple signal descriptions for example for different communication channels we can design for different types of communication environments as well.

Appendix A

Generating the system level module

We presented the system level formulation approach in Chapter 2, Fig. 2-8.

Now we will describe how the system level formulation generator works. We describe the basic functionality of system description generator. We, also, explain how a signal processing chain model can be used to write configuration files for the generator script.

To generate the system level description we use linear system analysis of the signal processing chain, thus every filter can be represented with a rational transfer function [36, 33]. Moreover, every rational transfer function can be decomposed into serial connection of FIR and IIR filters. Such from is, obviously, not unique but certain solutions can be easily found: taking the numerator (FIR) and the denominator (IIR) part of an expression. Thus, the system level formulation generating scripts recognize two basic building blocks: an IIR filter and an FIR filter. Both filter types are derived form a filter class which is an object with two main members: nominator and denominator. In the FIR class we define only the nominator, setting the denominator to 1. Opposite is done for the IIR class.

We can name every filter, and request certain oversampling of the filter transfer function. This (oversampling) feature is (mainly) for convenience when defining transmit equalization that is symbol-spaced, but could be used in different settings, if necessary. Thus, an FIR filter can be introduced into a model with

filter: FIR

name: aFR

order: 1

signs: +-

oversampling rate: 12

Such entry the generator script will interpret in the following way:

1. The filter coefficients are defined (for example, for the first order filter the coefficients will be $1, aFR_1$).

2. A function is defined by using the coefficients and appropriate signs. It is good to simplify the formulation by providing the signs if known. However, we can define an unknown sign by putting '*' instead of $\pm$. In this case two auxiliary variables ('positive' and 'negative') will be defined so that $aFR_n = aFR_n^+ - aFR_n^-$.

3. The function is oversampled by repeating the appropriate sample required number of times. This is known as zero-order hold [33].

4. Finally a *filter* object of the *FIR* type is created, and the numerator is assigned the value of the function we previously constructed. Denominator function is defined as 1.

It should be straightforward to see how an IIR filter is handled.

To generate the system level description, as discussed in Chapter 2, we need to define number of samples we want to consider, and to define the channel response. Also, the oversampling of the channel impulse response relative to the symbol-rate is necessary to perform the eye calculations. This information is introduced at the beginning of the configuration file

(sampled) impulse response vector: imp12x_800gbps.txt

response length: 20

symbol sampling rate: 12

where *imp12x_800gbps.txt* is a filename of an ASCII file with impulse response samples. The *response length* is number of significant symbol-time intervals we expect to have after the equalization. It is can be estimated from the unequalized impulse response. We should, always, be conservative when defining this value to prevent aliasing. However, requesting too many intervals will produce an ill-conditioned optimization problem due to the fact that number of points for IFT will be large and IFT matrix will be ill-conditioned. We already discussed this problem in Chapter 2.

Finally, we should give a few comments about the current code version (v1.0).

- Current configuration file parser is extremely simple. Thus, even the ordering of commands within each configuration block is important as well as spacing in each command line. For example, having multiple spacing after each colon would, probably, confuse the parser.

- Similarly, the file containing the impulse response should have only one line of text. All entries are separated with exactly one space, and there should not be any leading or trailing spaces.

- Currently, the script is not capable of handling the constant non-symbolic filters. Thus, if we refer back to Chapter 3 and Eq. (3.14) we can observe that the transfer function of receive equalizer in $\mathcal{Z}$-domain contains a $1+z^{-1}$ term. This additional filtering cannot be performed in the generator script currently, and should be included in the impulse response vector when we generate system level formulation that includes the receiver.

- Also, some variables, like *swingRED* and *scale*, as defined in Chapters 2 and 3, need to be defined by hand every time. This can be automated. However, it was outside of the scope of our work and is more in the domain of user-interface for our helper scripts.

- Finally, the script will generate all the necessary expressions to define Q_j and Y_i as well as all the possible samples of the *unwrapped eye*. However, in each particular run, we use only one particular eye sample value and to speed up parsing and solving time all the others should be commented out. Similarly, the dynamic range of the signal can be estimated from only the 'tail' of the unwrapped eye Fig. 2-7, and most of other samples are not needed as well. Furthermore, only the dynamic range is of importance at the input of the receiver equalizer so all the samples of the unwrapped eye at that point (PRElevel formulation in Fig. 4-1) can be removed from the formulation, thus saving in parsing and solving time.

We use an example to clarify these observations.

Example. According to our discussion in Chapter 3, and referring to Fig. 4-1, we generated the pre-receiver signal description *PRElevel* and the system level description *TOPlevel* using the following configuration files

```
(sampled) impulse response vector: imp12x_625gbps.txt
response length: 18
downsample times: 1
calculate points: 500
symbol sampling rate: 12
phases to sweep: 1

# the digital TX equalization
filter: FIR
name: aFR
order: 1
signs: +-
oversampling rate: 12
maxout: no
```

for the *PRElevel* module and

```
(sampled) impulse response vector: imp12x_625gbps_integrated.txt
response length: 18
```

```
downsample times: 1
calculate points: 500
symbol sampling rate: 12
phases to sweep: 1

# the drain pole of the RX equalizer
filter: IIR
name: dpRX
order: 1
signs: +-
oversampling rate: 1
bit-intervals to settle: 1

# the source pole of the RX equalizer
filter: IIR
name: spRX
order: 1
signs: +-
oversampling rate: 1
bit-intervals to settle: 1

# the digital TX equalization
# we're trying out the new mechanism
filter: FIR
name: aFR
order: 1
signs: +-
oversampling rate: 12
maxout: no
```

```
# the analog RX equalization zero influence
filter: FIR
name: zRX
order: 1
signs: +-
oversampling rate: 1
maxout: no
```

for the *TOPlevel* module.

As we have discussed in this section, we use different impulse response files to account for additional integration in the discretized receiver model. As we mentioned before, information about eye can be removed from the *PRElevel* module, and only the dynamic range information is of interest. Finally, in the *TOPlevel* module the header should be manually finished to look like

```
CKT_define topPOST()
CKT_var swingIN;
CKT_bounds swingIN 1e-1 1;

CKT_var dpRX1;
CKT_bounds dpRX1 0.58 0.88;

CKT_var spRX1;
CKT_bounds spRX1 0.58 0.88;

CKT_var aFR1;
CKT_bounds aFR1 0.05 0.99;

CKT_var zRX1;
CKT_bounds zRX1 0.58 0.92;
```

```
CKT_var scale;
CKT_bounds scale 1e-4 20;

CKT_var swingRED;
CKT_constr swr : swingRED*(1+aFR1) == swingIN;
```

thus providing the necessary variables, as well as taking into account the peak power (swing) constraint that we described in Chapter 2 and 3. Also, an eye sample should be exported which can be done (in agreement with the linker level code) simply by adding

```
objectv = scale*eye46;
CKT_save objectv;
```

where we include the scaling factor from the receiver equalizer model, Eq. (3.14, and export the phase 46 in our formulation, in this particular case. Usually, finding the right phase will require some experimentation and is best done by inspection of the unequalized impulse response.

Bibliography

[1] International technology roadmap for semiconductors, 2005.

[2] Dimitri Bertsimas and John N. Tsitsiklis. *Introduction to Linear Programming.* Athena Scientific, 1997.

[3] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, Cambridge, UK, 2004.

[4] J. R. Phillips C. P. Coelho and L. M. Silvera. Robust rational function approximation algorithm for model generation. 1999.

[5] B.K. Casper, M. Haycock, and R. Mooney. An accurate and efficient analysis method for multi-gb/s chip-to-chip signaling schemes. *Symposium on VLSI Circuits Digest of Technical Papers, 2002.*, (SN -):54–57, 2002.

[6] J.R. Chen-Chu Yeh; Barry. Adaptive minimum bit-error rate equalization for binary signaling ,. *Communications, IEEE Transactions on , vol.48, no.7, pp.1226-1235, Jul 2000*, 2000.

[7] Patrick Chiang, William J. Dally, Ming-Ju Edward Lee, Ramesh Senthinathan, Jangjin Oh, and Mark Horowitz. A 20gb/s 0.13um cmos serial link transmitter using an lc-pll to directly drive the output multiplexer. In *Symposium On VLSI Circuits*, 2001.

[8] Seong Rag Kim; Young Gyun Jeong; In-Kyeong Choi. A constrained mmse receiver for ds/cdma systems in fading channels. *Communications, IEEE Transactions on*, 48(11):1793–1796, Nov 2000.

[9] D.M. Colleran, C. Portmann, A. Hassibi, C. Crusius, S.S. Mohan, S. Boyd, T.H. Lee, and M. del Mar Hershenson. Optimization of phase-locked loop circuits via geometric programming. *Proceedings of the IEEE Custom Integrated Circuits Conference, 2003.*, (SN -):377–380, 2003.

[10] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. of Control, Signals and Systems, 2, pp. 303 - 314*, 1989.

[11] Maria del Mar Hershenson. Design of pipeline analog-to-digital converters via geometric programming. In *ICCAD '02: Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pages 317–324, 2002.

[12] Maria del Mar Hershenson, Stephen P. Boyd, and Thomas H. Lee. Gpcad: a tool for cmos op-amp synthesis. In *ICCAD '98: Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, pages 296–303, 1998.

[13] Maria del Mar Hershenson, Ali Hajimiri, Sunderarajan S. Mohan, Stephen P. Boyd, and Thomas H. Lee. Design and optimization of lc oscillators. In *ICCAD '99: Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design*, pages 65–69. IEEE Press, 1999.

[14] Maria del Mar Hershenson, Sunderarajan S. Mohan, Stephen P. Boyd, and Thomas H. Lee. Optimization of inductor circuits via geometric programming. In *DAC '99: Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 994–998, 1999.

[15] R. Farjad-Rad, N.g. Hiok-Taiq, M.J. Edward Lee, R. Senthinathan, W.J. Dally, A. Nguyen, R. Rathi, J. Poulton, J. Edmondson, J. Tran, and H. Yazdanmehr. 0.622-8.0 gbps 150 mw serial io macrocell with fully flexible preemphasis and equalization. *Symposium on VLSI Circuits, 2003. Digest of Technical Papers. 2003*, (SN -):63–66, 2003.

[16] Gibson G.J., Siu S., and Cowan C.F.N. The application of nonlinear structures to the reconstruction of binary signals,. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] , vol.39, no.8, pp.1877-1884, Aug 1991*, 1991.

[17] Paul R. Gray, Paul J. Hurst, Stephen H. Lewis, and Robert G. Meyer. *Analysis and Design of Analog Integrated Circuits, 4th Edition.* Wiley, 2001.

[18] M.M. Griffin, J. Zerbe, G. Tsang, M. Ching, and C.L. Portmann. A process-independent, 800-mb/s, dram byte-wide interface featuring command interleaving and concurrent memory operation. *Solid-State Circuits, IEEE Journal of, Vol.33, Iss.11, Nov 1998*, pages 1741–1751.

[19] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Transactions on Power Deliery*, 14(3), 1999.

[20] A. Hajimiri and T. Lee. A general theory of phase noise in electrical oscillators. 1998.

[21] H. Hatamkhani and C.K. Ken Yang. A study of the optimal data rate for minimum power of i/os. *IEEE Transactions on Circuits and Systems II*, 53(11 SN - 1057-7130):1230–1234, 2006.

[22] M. Hershenson. Efficient description of the design space of analog circuits. *In Proceedings Design Automation Conference, 2003.*, (SN -):970–973, 2003.

[23] M. Hershenson, S.P. Boyd, and T.H. Lee. Optimal design of a cmos op-amp via geometric programming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(1 SN - 0278-0070):1–21, 2001.

[24] Yiteng Arden Huang and Jacob Benesty. Adaptive multi-channel least mean square and newton algorithms for blind channel identification. *Signal Process.*, 82:1127–1138, 2002.

[25] S. Joshi and S. Boyd. An efficient method for large-scale gate sizing. *IEEE Transactions on Circuits and Systems*, 2006.

[26] M. Kun-Yung Ken Chang; Shang-Tse Chuang; McKeown, N.; Horowitz. A 50 gb/s 3232 cmos crossbar chip using asymmetric serial links. *VLSI Circuits, 1999. Digest of Technical Papers. 1999 Symposium on, Vol., Iss., 1999*, pages 19–22, 1999.

[27] Nir Magen, Avinoam Kolodny, Uri Weiser, and Nachum Shamir. Interconnect-power dissipation in a microprocessor. In *SLIP '04: Proceedings of the 2004 international workshop on System level interconnect prediction*, pages 7–13, New York, NY, USA, 2004. ACM Press.

[28] Pradip Mandal and V. Visvanathan. Cmos op-amp sizing using a geometric programming formulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(1):22–38, 2001.

[29] O. L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research, Vol. 13, No. 3, pp. 444-452*, 1965.

[30] C. Maranas and C. Floudas. Global optimization in generalized geometric programming, 1997.

[31] Piotr Mitros. A framework for analog circuit optimization. Master's thesis, MIT, 2004.

[32] Dimitri P. Bertsekas; Angelina Nedic and Asuman E. Ozdalgar. *Convex Analysis and Optimization.* Athena Scientific, 2003.

[33] Alan V. Oppenheim and Ronald W. Schafer. *Digital Signal Processing.* Prentice-Hall, 1975.

[34] George A. Perdikaris. *Computer controlled systems : theory and applications.* Kluwer Academic Publishers, 1991.

[35] R. Phelps, M.J. Krasnicki, R.A. Rutenbar, L.R. Carley, and J.R. Hellums. A case study of synthesis for industrial-scale analog ip: redesign of the equalizer/filter frontend for an adsl codec. *In Proceedings 37th Design Automation Conference*, (SN -):1–6, 2000.

[36] John Proakis. *Digital Communications.* McGraw-Hill, 2000.

[37] Jihong Ren and Mark Greenstreet. A unified optimization framework for equalization filter synthesis. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 638–643, 2005.

[38] Jihong Ren, Haechang Lee, Qi Lin, Brian Leibowitz, E hung Chen, Dan Oh, Frank Lambrecht, Vladimir Stojanovic, Chih-Kong Ken Yang, and Jared Zerbe. Precursor isi reduction for high-speed i/o. In *Symposia on VLSI Technology and Circuits*, 2007.

[39] J. J. E. Slotine and Weiping Li. *Applied Nonlinear Control.* Prentice Hall, 1991.

[40] Kin Chong Sou, Alexandre Megretski, and Luca Daniel. A quasi-convex optimization approach to parametrized model order reduction. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 933–938, 2005.

[41] V. Stojanovic, A. Ho, B. Garlepp, F. Chen, J. Wei, E. Alon, C. Werner, J. Zerbe, and M.A. Horowitz. Adaptive equalization and data recovery in a dual-mode (pam2/4) serial link transceiver. *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on , vol., no., pp. 348-351, 17-19 June 2004.*

[42] V. Stojanovic and M. Horowitz. Modeling and analysis of high-speed links. In *IEEE Custom Integrated Circuits Conference*, 2003.

[43] Vladimir Stojanovic. *Channel-Limited High-Speed Links: Modeling, Analysis And Design.* PhD thesis, Stanford, 2005.

[44] Yannis Tsividis. *Operation and Modeling of the MOS Transistor.* Oxford Uniersity Press, 2003.

[45] www.eas.asu.edu/ ptm/.

[46] J.L. Zerbe, C.W. Werner, V. Stojanovic, F. Chen, J. Wei, G. Tsang, D. Kim, W.F. Stonecypher, A. Ho, T.P. Thrush, R.T. Kollipara, M.A. Horowitz, and K.S. Donnelly. Equalization and clock recovery for a 2.5-10-gb/s 2-pam/4-pam backplane transceiver cell. *IEEE Journal of Solid-State Circuits*, 38(12 SN - 0018-9200):2121–2130, 2003.

5938 · 69