Feedforward MLSE Equalization for High Speed Serial Links

Paul Kwon

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-215
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-215.html

August 11, 2023

Feedforward MLSE Equalization for High Speed Serial Links

By

Paul Kwon

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Vladimir Stojanovic, Co-chair
Professor Elad Alon, Co-chair
Professor Ali Niknejad
Professor Martin White

Summer 2023

Feedforward MLSE Equalization for High Speed Serial Links

Abstract

Feedforward MLSE Equalization for High Speed Serial Links

by

Paul Kwon

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Vladimir Stojanovic, Co-chair

Professor Elad Alon, Co-chair

An ever-increasing demand for high data rate wireline links must be met to support the continual scaling of computing and communication systems. Typical serial link architectures use feedback-based channel equalization schemes, which can be challenging and even infeasible to realize for data rates beyond 100 GBaud/s. To alleviate this feedback-induced latency bottleneck, this thesis explores feedforward equalizers inspired by the maximum likelihood sequence estimation (MLSE) algorithm. Targeting short-reach, die-to-die links, a 1-tap fully feedforward MLSE architecture is shown to achieve comparable error statistics with the conventional 1-tap decision feedback equalizer (DFE).

A 160 Gb/s NRZ receiver implementing the 1-tap MLSE equalizer is taped out in a 16 nm FinFET process to evaluate the future promise of the proposed approach. The feedforward MLSE is time-interleaved to achieve the desired throughput. Current integration techniques provide energy-efficient analog latches used in the MLSE datapath. The datapath is designed using the Berkeley Analog Generator (BAG) framework. Both the top level datapath and its subblocks are created using parameterizable circuit schematic and layout generators. Measurement and design scripts identify the impact of inter-subblock routing parasitics, facilitating agile subblock design iteration with top level floorplanning in mind.

As the channel characteristics are unknown a priori, the coefficient settings of the MLSE must be adapted in real time. An adaptation scheme for the feedforward MLSE equalizer is proposed with separate loops for each interleaved MLSE slice to account for process variation. The receiver is taped out along with a corresponding 160 Gb/s NRZ transmitter and tested over an on-package loopback channel of 8.5 mm, which incurs 3 dB of loss at the Nyquist frequency. The receiver is simulated to operate at an energy efficiency of 2.08 pJ/bit, with the datapath itself consuming 0.72 pJ/bit.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

When I started my undergraduate program at UC Berkeley, I thought I'd be out of here in 4 years. Little did I know that Berkeley would give me a decade of opportunities and growth.

I would first like to thank my advisors: Professors Elad Alon and Vladimir Stojanović. I've had the privilege to engage in many technical conversations with Professor Elad throughout my Ph.D. education. His sharp circuit intuition never ceases to amaze me. He's also motivated to not focus on just my project, but rather on the larger narrative of which my research is a small piece. Professor Vladimir has been instrumental in broadening my perspectives. His visions for the future of wireline communications and design methodologies inspire me to pursue big dreams.

I also thank Professors Ali Niknejad and Martin White for serving on my qualifing exam and dissertation committees. I'm grateful for their guidance on my work. I would also like to thank Professor Borivoje Nikolić for his technical advice during design reviews.

I appreciate the opportunity to share my research progress with Farhana Sheikh, Jahnavi Sharma, and many others from Intel. It's always great to receive feedback from industry for their unique, non-academic perspectives.

I'm thankful for the wonderful support I've received from the BWRC staff: Jeff Anderson-Lee, Brian Richards, Candy Corpus, and Mikaela Cavizo-Briggs. Special thanks to Anita Flynn for her assistance with PCB design reviews and chip testing options.

Over the years, I've been fortunate to work with and learn from many brilliant peers. As an undergraduate fledgling unsure of what to pursue, I thank Seobin Jung for the opportunity to assist in her research, as well as Jaeduk Han for providing my first exposure to wireline circuits. I would like to thank other senior students and postdocs who I've learned a lot from both in and beyond my area of expertise: Eric Chang, Minsoo Choi, Greg LaCaille, Richard Lin, Pengpeng Lu, Ali Moin, Nathan Narevsky, Emily Naviasky, Antonio Puglielli, Nick Sutardja, Konstantin Trotskovsky, Zhongkai Wang and Bonjern Yang. I'm grateful for the opportunities to closely work together with Ayan Biswas, Kunmo Kim, Yi-Hsuan Shih, Bob Zhou, Wahid Rahman, and Antroy Chowdhury for the SerDes project. I'd also like to thank Zhaokai Liu, Kwanseo Park, Zhenghan Lin, Sean Huang, Sunjin Choi, Aviral Pandey, Dan Fritchman, and Meng Wei, and Rebecca Zhao.

I would like to thank my parents and my brother for their continual support over the years. I'm thankful for the times of rest I could share with my church community. Finally, I thank my girlfriend, Lucy Choi, for her steadfast love. Surviving 10 years of Berkeley, as rewarding as it is, is no easy task. I'm glad that I've been blessed with friends and family that I can lean on during this journey and whatever comes next.

Chapter 1

Introduction

1.1 Background

Ultra-high throughput, energy-efficient wireline links are key to enabling the next generation of computing and communication systems. Serializer-Deserializer (SerDes) transceivers form the backbone of many applications, ranging from high performance computing to artificial intelligence and Internet of Things. These systems scale over time, increasing the demand for higher rate serial links. In fact, the per-lane data rates for common I/O standards have doubled every 3-4 years (Figure 1.1).

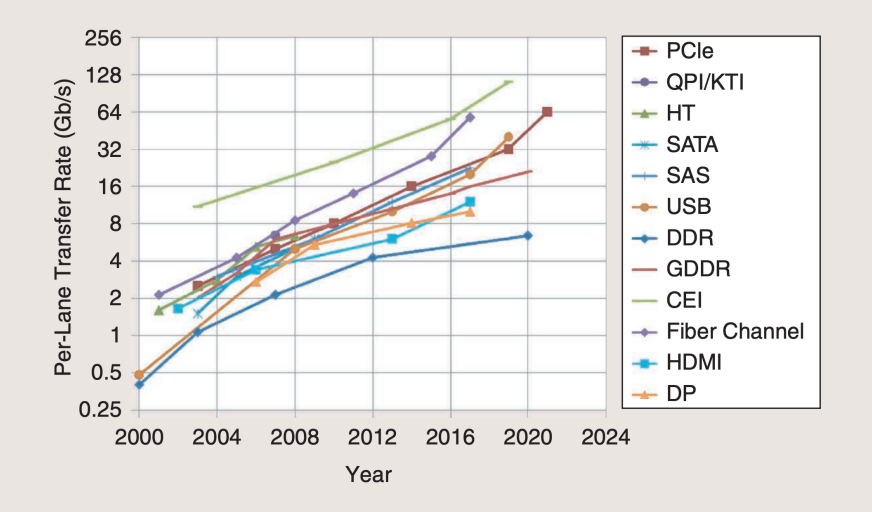


Figure 1.1: Link data rate trends [1]

To support the wide variety of applications, serial interfaces are often classified to different reaches. For example, networking systems typically require data to travel over a backplane with connectors. Wireline transceivers for such board-to-board communication must be

able to operate with long reach (LR) channels, which can introduce significant loss at high frequencies. On the other end of the spectrum, chip-to-chip interfaces require low-latency SerDes to transmit data over short, multi-chip module (MCM) links, with minimal channel loss.

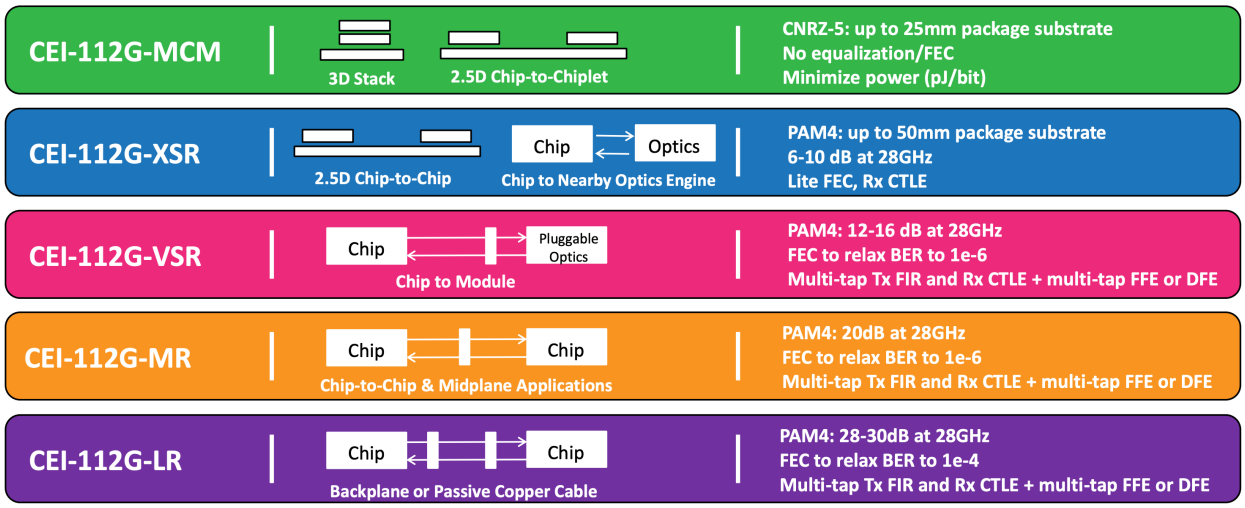


Figure 1.2: Optical Internetworking Forum (OIF) Common Electrical I/O (CEI) standards for different reach applications [2]

In recent years, a new application for wireline links has emerged. Whereas the traditional system-on-chip (SoC) approach places all integrated circuits on a monolithic die, growing transistor and design costs are pushing for these systems to be parititioned into smaller dies, or chiplets. The multi-chiplet, heterogeneous integration approach allows for higher yield due to smaller die sizes, and chiplets can be optimally designed in different technology nodes. This is best summed up by Gordon Moore's prediction in 1965: "It may prove to be more economical to build large systems out of smaller functions, which are separately packaged and interconnected." [3] To take advantage of multi-chiplet systems, die-to-die links must provide excellent bandwidth and energy efficiency.

1.2 Channel Equalization

Designing robust links requires equalization to counteract the loss of the wireline channel. Electrical channels often have low-pass filtering behaviors due to the skin effect and dielectric loss of the material. In addition, any impedance mismatches along the channel (e.g., between the connector and backplane) causes reflections, which further worsens the signal integrity.

The loss characteristics of the channel can be visualized in both frequency and time domain (Figure 1.4). Suppose the channel is modeled as a finite impulse response (FIR) filter, whose inputs are the symbols transmitted at each unit interval (UI). If modeled as an

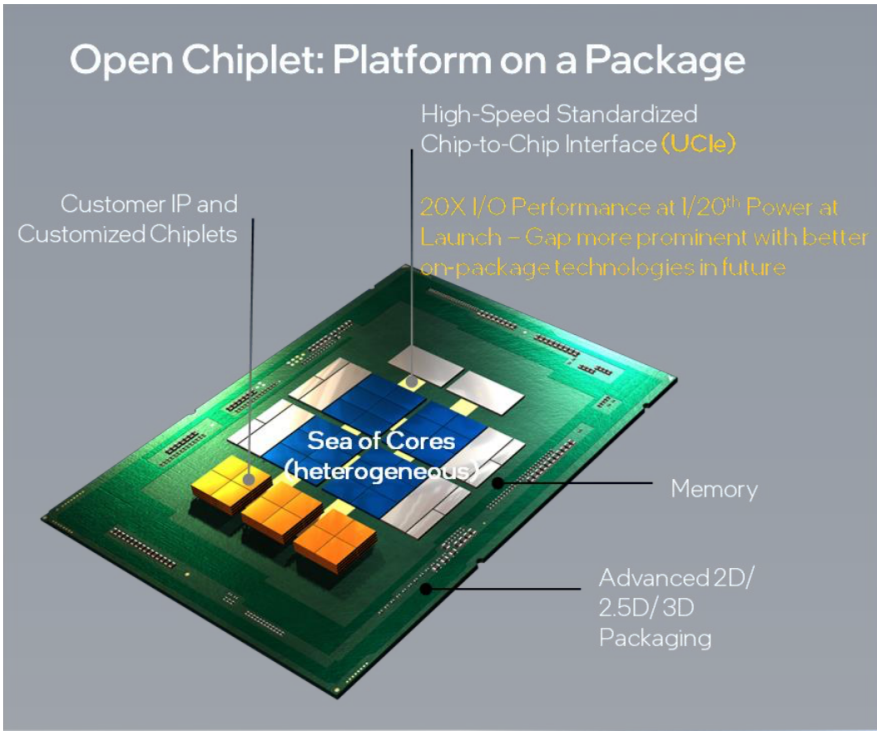


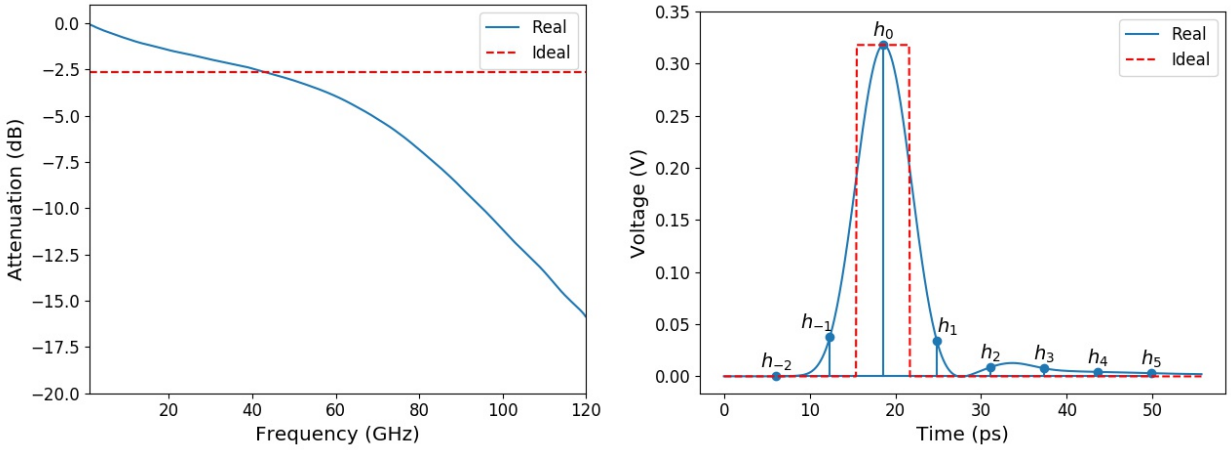
Figure 1.3: Chiplet interconnects [4]

FIR with M precursor taps, N postcursor taps, and 1 main tap, the channel will output y as a convolution of the FIR taps h and the input signal x:

$$y[n] = (h * x)[n] = \sum_{k=-\infty}^{\infty} h_k x[n-k] = \sum_{k=-M}^{N} h_k x[n-k] \tag{1.1}$$

The channel coefficients can be measured using a pulse response, where the output waveform is observed for a 1 UI-wide input pulse. The resulting time-domain response reveals the channel FIR taps spaced 1 UI apart (Figure 1.4b). Note that h_0 represents the main channel tap (the desired pulse). All other taps degrade signal integrity through intersymbol interference (ISI), where neighboring symbols are superpositioned onto the current receiver sample to varying degrees. ISI introduces data-dependent error terms which increase the link's bit error rate (BER). The goal of equalization thus is to flatten out the frequency response of the channel such that the time-domain pulse response shows little to no residual ISI.

With increasing data rates, more powerful equalization is required to maintain signal integrity. To the first order, the channel loss tends to increase linearly in dB with respect to frequency. In addition, the effective channel seen by the transceiver must include not only the physical routing channel, but also the transceiver circuits themselves. This becomes



(a) (b)

Figure 1.4: Low-loss channel frequency response (a) and pulse response (b)

especially important as data rates are pushed to the limits of the technology, since frontend circuits will face bandwidth limitations and contribute to more loss in the signal chain. The following sections will introduce conventional wireline receiver equalization techniques which may have disadvantages at high data rates.

1.2.1 Continuous Time Linear Equalizer (CTLE)

To counteract the channel's low-pass behavior, continuous time linear equalizers (CTLE) provide peaking gain. Typical implementations have a zero-pole pair, and the peaking gain is set by the ratio of the pole to zero frequencies. Active, transistor-based topologies attenuate low frequency signals through the resistive source degeneration. As frequencies increase, the source capacitance C_s begins to dominate, creating a low source impedance and reducing the effect of source degeneration. The peaking gain is set by the source degeneration effect $(1 + g_m R_s/2)$, and the zero and pole locations are computed below:

$$\omega_z = \frac{1}{R_s C_s} \tag{1.2}$$

$$\omega_p = \omega_z \cdot (1 + g_m R_s/2) \tag{1.3}$$

Active CTLE designs also suffer from an additional pole at the output, which causes a 20 dB per decade roll-off in gain for frequencies above $\frac{1}{R_D C_L}$. Passive CTLE topologies are designed with parallel RC structures, where the resistive divider determines low-frequency gain and the capacitive divider determines high-frequency gain. The zero and pole are

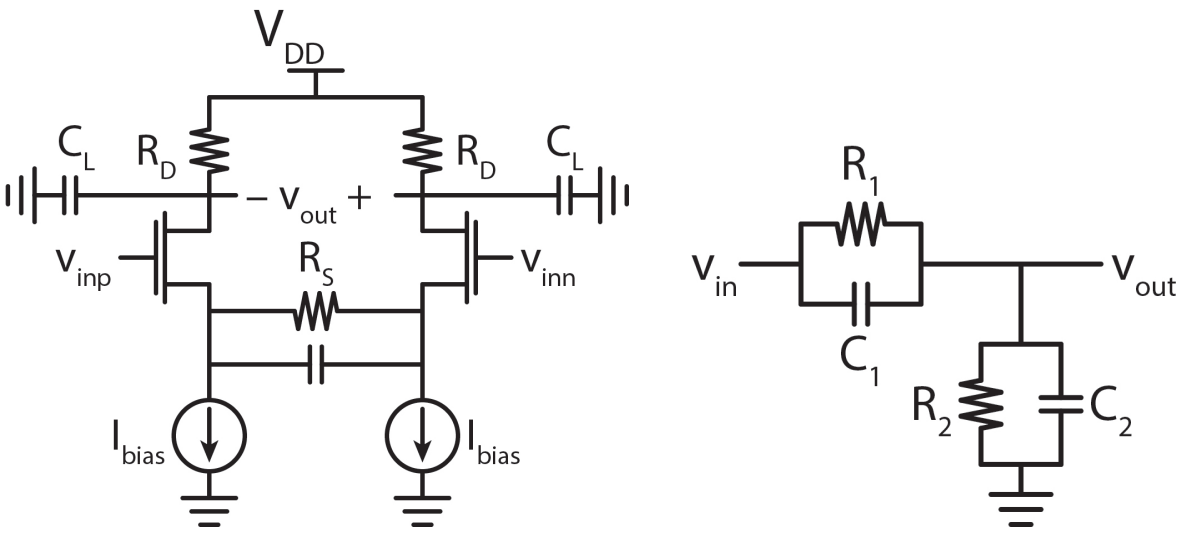


Figure 1.5: CTLE: active (a) and passive (b) topologies

computed below:

$$\omega_z = \frac{1}{R_1 C_1} \tag{1.4}$$

$$\omega_p = \frac{1}{(R_1 \parallel R_2)(C_1 + C_2)} \tag{1.5}$$

Being a frequency-domain equalizer, the CTLE is often used to cancel long-tail postcursor ISI that would otherwise be challenging for time-domain equalizers (whose complexity scales with number of taps). Equalizing near-tap ISI with the CTLE poses circuit design challenges due to how high the poles and zeros would need to be placed. Because the CTLE is a frequency-domain equalizer, it must be placed at the datapath frontend and operate at full bit rate. With increasing data rates, parasitics are likely to limit the bandwidth and equalizing capabilities of the CTLE. Furthermore, the CTLE suffers from noise enhancement, as both the noise and signal are equally boosted at high frequencies.

1.2.2 Feedforward Equalizer (FFE)

The feedforward equalizer (FFE) is an FIR whose tap coefficients are set to cancel pre- or postcursor ISI taps from the channel. The FFE can be implemented at either the transmitter or the receiver. Like the CTLE, the receiver FFE also amplifies noise because the inputs to the FIR are analog samples. At the transmitter, the FFE is robust to noise as its inputs are digital symbols. However, headroom constraints on the transmitter output swing require a normalization of the FFE tap coefficients, reducing the effective signal amplitude. Despite

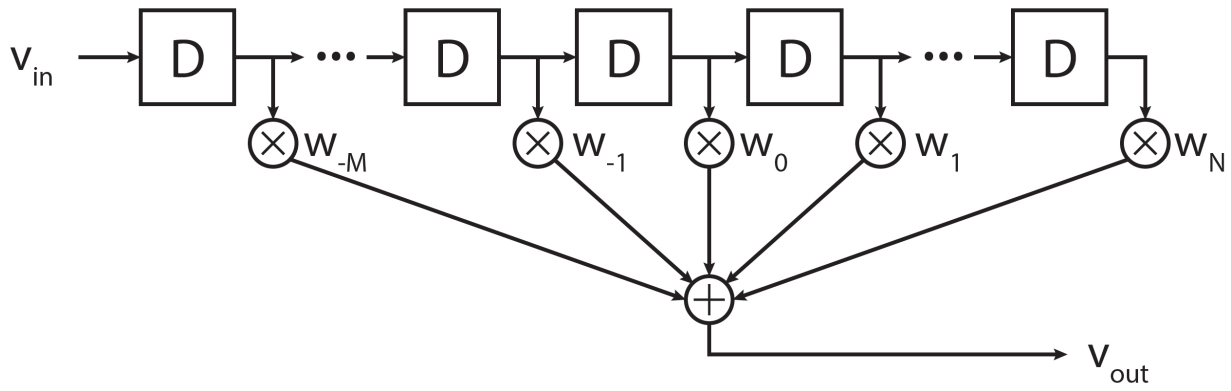


Figure 1.6: FFE with M pre-cursor and N post-cursor taps

its data rate scalability as a parallelizable equalizer scheme, the FFE is usually combined with other equalizers for low BER applications due to noise enhancement and headroom issues.

1.2.3 Decision Feedback Equalizer (DFE)

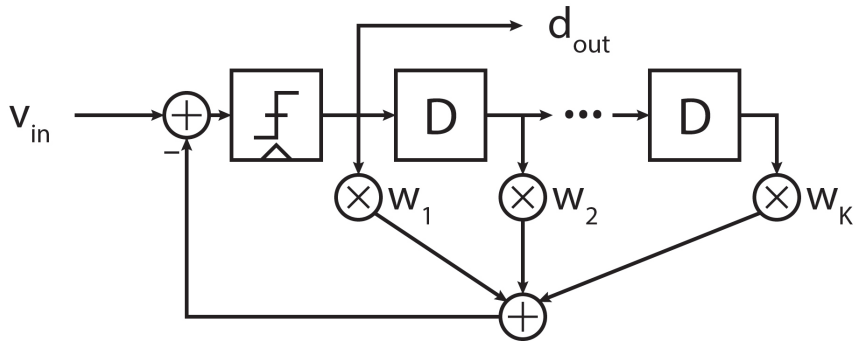


Figure 1.7: DFE

The decision feedback equalizer (DFE) is another FIR-based equalizer which cancels ISI based on previous decisions. The main advantage of the DFE is its robustness to noise because the FIR takes digital symbols as inputs. The DFE is limited to equalizing postcursor taps as it operates on previously decoded bits. Any past incorrect decisions can also propagate through the FIR, potentially generating additional errors. From a frequency scalability perspective, the DFE is bottlenecked by its feedback loop, where the tap coefficient weighting, analog summation, and symbol resolution all must be completed in 1 UI.

To reduce the critical path, loop-unrolling techniques have been applied to the DFE. A 1-tap loop-unrolled DFE structure is shown in Figure 1.8. In practice, loop-unrolling is

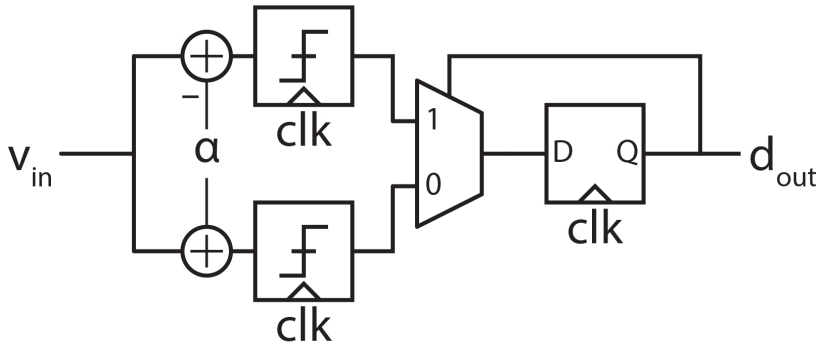


Figure 1.8: 1-tap loop-unrolled DFE

limited to 1-tap because this technique causes the complexity to scale exponentially with the number of loop-unrolled taps, and loop-unrolling increases the delay associated with later non-unrolled taps. Nevertheless, a feedback loop still exists and limits the speed of this equalizer.

1.3 Thesis Organization

Given the limitations of traditional SerDes equalization algorithms, a noise-robust, feedforward equalizer architecture is critical to building power-efficient links at 100+ GBaud/s rates. Short die-to-die links do not require heavy equalization, but must deal with noise-limited channels where noise amplification could limit the achievable BER of the transceiver. In such applications, the DFE is particularly attractive as its robustness to noise, but its feedback-induced timing constraint becomes challenging, if not feasible, to meet for such high data rates. Consequently, Chapter 2 explores new feedforward architectures inspired by the classical Maximum Likelihood Sequence Estimation (MLSE) algorithm as alternatives to the DFE. Tradeoffs between computational complexity and error statistics of proposed algorithms are discussed, along with statistical analysis strategies for sequence-detection equalizers. Targeting a data rate of 160 Gb/s NRZ, Chapter 3 presents a receiver datapath utilizing the feedforward MLSE equalizer. Circuit design techniques and methodologies are introduced to achieve energy efficient links as communication speeds approach limits of the technology. Next, Chapter 4 describes the remaining receiver features needed to verify the datapath, including the clocking path and on- and off-chip calibration loops. Simulation results show that the receiver can achieve 160 Gb/s at 2.08 pJ per bit under a 3 dB loss on-package channel. Chapter 5 concludes the thesis with a summary of the proposed work and potential directions for future research.

Chapter 2

Statistical Analysis and Architecture Exploration

2.1 Maximum Likelihood Sequence Estimation

2.1.1 Overview

As its name suggests, the MLSE algorithm chooses the most likely sequence of data symbols by comparing all possible sequences within a defined window. When dealing with additive white Gaussian noise (AWGN) channels, the most probable sequence is that which has the smallest Euclidean, or L2, norm [5].

The MLSE has a few key differences when compared to typically used equalizers like the FFE, DFE, and CTLE. First, such conventional equalizers often aim to reduce or cancel ISI energy, whereas MLSE retains and uses that ISI energy to help decode the transmitted sequence. Second, while typical equalizers often perform symbol-by-symbol detection, the MLSE inherently recovers a sequence of symbols by applying the redundant information embedded in the ISI.

An example is shown in Figure 2.1. Suppose that symbols are transmitted through NRZ signaling with voltages of ± 1, and that the channel of interest is a 1-tap postcursor channel, with main cursor $h_0 = 1$ and first postcursor $h_1 = \alpha$. The channel is effectively a 2-tap finite impulse response (FIR) filter, whose noiseless outputs are $\pm 1 \pm \alpha$ depending on the data pattern. At unit interval (UI) n, the received sample $v_r[n]$ deviates from the 4 noiseless expected states with the following error magnitudes:

$$\begin{aligned} e_{11}[n]^2 &= [v_r[n] - (1+\alpha)]^2 \\ e_{01}[n]^2 &= [v_r[n] - (1-\alpha)]^2 \\ e_{10}[n]^2 &= [v_r[n] - (-1+\alpha)]^2 \\ e_{00}[n]^2 &= [v_r[n] - (-1-\alpha)]^2 \end{aligned} \tag{2.1}$$

Note that each received sample provides information about 2 consecutive symbols because

the channel memory spans 2 taps.

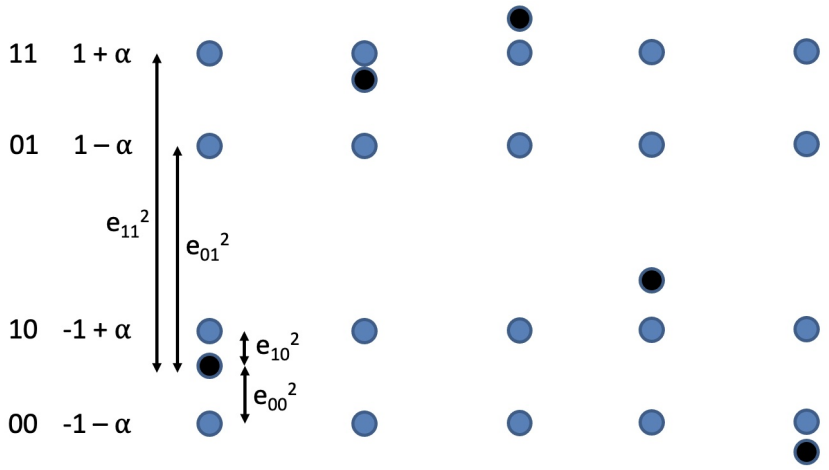


Figure 2.1: Ideal voltage states (blue) and received voltage samples (black) for a channel with coefficients $[h_0, h_1] = [1, \alpha]$ and NRZ signaling with data levels of ± 1

Let the window length L be the number of consecutive UI over which we observe to compute the most likely sequence. In other words, L becomes the number of UI over which errors are accumulated in an L2-norm fashion.

Considering the same 1-tap post-cursor channel, the most likely sequence for a window length L contains $L + 1$ symbols and can be computed as

$$W(n) = \begin{bmatrix} d_r[n-L] & \cdots & d_r[n-1] & d_r[n] \end{bmatrix} = \underset{x \in \{0,1\}^{L+1}}{\arg\min} \left\{ \sum_{i=1}^{L} e_{x_i x_{i+1}}[n+i-L]^2 \right. \tag{2.2}$$

Most classic implementations of the MLSE use the Viterbi algorithm, which recursively decodes the most likely window. Suppose the path metric $P_b[n]$ is defined as the minimum L2 norm across all sequences whose current bit is $b \in \{0, 1\}$:

$$P_b[n] \equiv \min_{x \in \{0,1\}^L} \left\{ \left(\sum_{i=1}^{L-1} e_{x_i x_{i+1}}[n+i-L]^2 \right) + e_{x_L b}[n]^2 \right. \tag{2.3}$$

The path metric equation can be re-defined using its previous value:

$$
\begin{aligned}
P_b[n] &= \min_{x\in\{0,1\}^{L-1}} \begin{cases} \left(\sum_{i=1}^{L-2} e_{x_i x_{i+1}}[n+i-L]^2\right) + e_{x_{L-1}0}[n-1]^2 + e_{0b}[n]^2 \\ \left(\sum_{i=1}^{L-2} e_{x_i x_{i+1}}[n+i-L]^2\right) + e_{x_{L-1}1}[n-1]^2 + e_{1b}[n]^2 \end{cases} \\
&= \min \begin{cases} \left[\min_{x\in\{0,1\}^{L-1}} \left\{\left(\sum_{i=1}^{L-2} e_{x_i x_{i+1}}[n+i-L]^2\right) + e_{x_{L-1}0}[n-1]^2\right\}\right] + e_{0b}[n]^2 \\ \left[\min_{x\in\{0,1\}^{L-1}} \left\{\left(\sum_{i=1}^{L-2} e_{x_i x_{i+1}}[n+i-L]^2\right) + e_{x_{L-1}1}[n-1]^2\right\}\right] + e_{1b}[n]^2 \end{cases} \\
&= \min \begin{cases} P_0[n-1] + e_{0b}[n]^2 \\ P_1[n-1] + e_{1b}[n]^2 \end{cases}
\end{aligned} \tag{2.4}
$$

The Viterbi algorithm applies this principle to simplify the most likely sequence estimation (Figure 2.2). For each cycle, the branch metrics are computed, which represent the squared error terms $e_{11}[n]^2, e_{01}[n]^2, e_{10}[n]^2, e_{00}[n]^2$ for the current sample n. Then, the path metrics are updated using an add-compare-select (ACS) unit with the branch metric as new single-UI error terms. By using feedback, the number of comparisons needed to decode the sequence increases linearly with window length. This advantage is unfortunately incompatible with feedforward designs that are scalable with data rate.

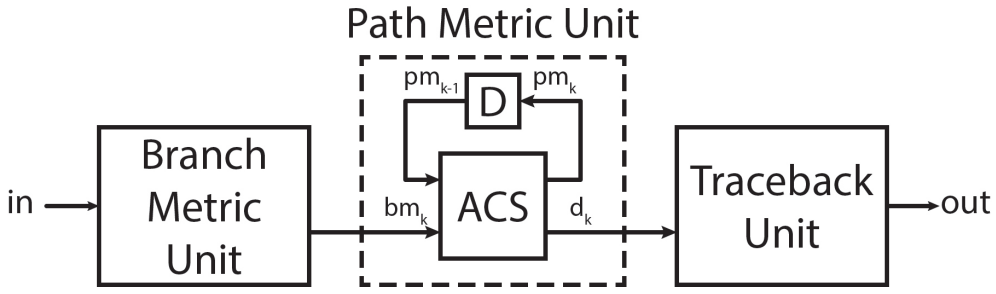


Figure 2.2: Viterbi equalizer

To architect an energy-efficient MLSE-based equalization algorithm at ultra-high data rates, one must address a few key challenges. First, the complexity of the MLSE algorithm scales exponentially with the number of taps and window length. For an N-tap feedforward MLSE (i.e., one that will equalize a total of N precursors and postcursors) with a window length of L and a PAM-M modulation scheme, the number of states for a given sample is M^N, and the number of possible sequences in the window is M^{N+L}. In contrast, the FFE and DFE increase linearly in complexity with respect to the number of taps. For

low-power applications, the receiver can employ an MLSE with a low (1-2) number of taps, with additional equalization like FFE or DFE as necessary to cover the span of the channel. Decreasing the window length reduces the complexity as well, at the cost of higher error rates, as will be discussed in later sections.

Second, one must consider how to resolve conflicting interactions between overlapping windows. For example, let's take a 1-tap postcursor MLSE with window length = 1. (2.2) simplifies to

$$W(n) = \begin{bmatrix} d_r[n-1] & d_r[n] \end{bmatrix} = \underset{x \in \{0,1\}^2}{\arg\min} \begin{cases} e_{11}[n]^2 & \text{if } x = [1,1] \\ e_{01}[n]^2 & \text{if } x = [0,1] \\ e_{10}[n]^2 & \text{if } x = [1,0] \\ e_{00}[n]^2 & \text{if } x = [0,0] \end{cases} \tag{2.5}$$

Thus, for the received samples shown in Figure 2.3, we would get the following decoded sequences:

$$W(1) = [0,0] \Rightarrow d_r[0] = 0, d_r[1] = 0 \tag{2.6}$$
$$W(2) = [0,1] \Rightarrow d_r[1] = 0, d_r[2] = 1 \tag{2.7}$$
$$W(3) = [0,1] \Rightarrow d_r[2] = 0, d_r[3] = 1 \tag{2.8}$$

The decoded sequences from UIs 1 and 2 yield a consistent bit for $d_r[1]$. However, the sequences from UIs 2 and 3 yield different bits for $d_r[2]$. This conflict results from these windows independently decoding bits. This problem could be avoided if the previous window's decision could be used to decode the current window. In the above example, if $d_r[2] = 1$ was known from decoding of window 2, the possible states in the decoding of window 3 would be limited to $[1,0]$ and $[1,1]$, corresponding to voltages $-1+\alpha$ and $1+\alpha$, and the conflicting scenario $[0,1]$ would never be considered. However, this would introduce feedback to the algorithm. A feedforward MLSE equalizer must thus decode each window independently while resolving any conflicts.

2.2 Statistical Analysis Framework

Error statistics are key to evaluating the effectiveness of a MLSE equalizer. For conventional symbol-by-symbol equalizers in NRZ signaling, a slicer compares the input differential signal to differential zero to determine the decoded bit. Ideally, this input signal is just $\pm h_0$, depending on whether the transmitted bit was a 1 or 0. In reality, the input contains error terms as well, such as any residual ISI post-equalization, and voltage noise. If the absolute sum of these error terms is enough to swap the polarity of the differential input, an error would occur. The BER can thus be written as

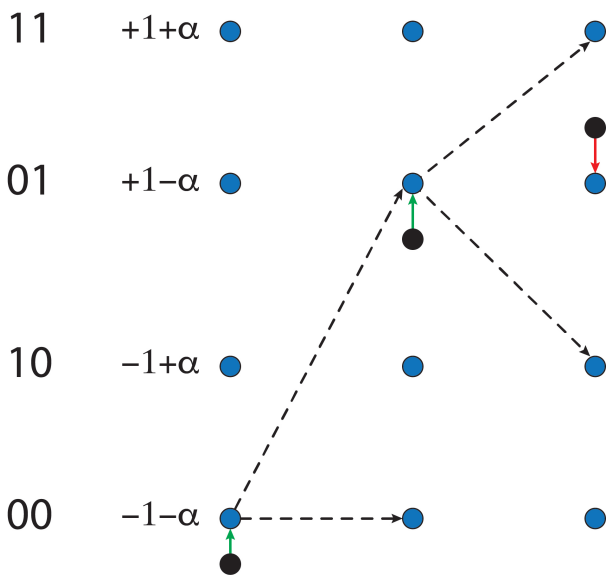


Figure 2.3: Window conflict example. Received voltage samples (black), with expected state transitions (dashed lines)

$$\begin{aligned} BER &= P(d_t = 1 \cap h_0 + v_n + v_{ISI} < 0) + P(d_t = 0 \cap -h_0 + v_n + v_{ISI} > 0) \\ &= P(d_t = 1)P(h_0 + v_n + v_{ISI} < 0) + P(d_t = 0)P(-h_0 + v_n + v_{ISI} > 0) \\ &= \frac{1}{2}[P(v_n + v_{ISI} < -h_0) + P(v_n + v_{ISI} > h_0)] \\ &= \frac{1}{2}[2P(v_n + v_{ISI} < -h_0)] = P(v_n + v_{ISI} < -h_0) \end{aligned}$$

where v_{ISI} is a discrete random variable that represents residual ISI, and v_n is a normally distributed continuous random variable representing voltage noise.

As noted in [6], the probability mass function of v_{ISI} can be derived by convolving the individual distributions of ISI cursors. Since ISI is independent of noise, the bit error rate can be computed as the following:

$$\begin{aligned} BER &= \sum_x P(v_{ISI} = x \cap v_n < -h_0 - x) \\ &= \sum_x \left[P(v_{ISI} = x) \cdot P(v_n < -h_0 - x)\right] \\ &= \sum_x \left[p_{ISI}(x) \cdot F_n(-h_0 - x)\right] \end{aligned} \tag{2.9}$$

where p_{ISI} is the PMF of the ISI and F_n is the CDF of the noise distribution. For an ideal, noise-limited channel that has no residual ISI post-equalization, (2.9) simplifies to

$$BER_{symbol,ideal} = F_n(-h_0) \tag{2.10}$$

Now, consider the proposed MLSE algorithm, which is a sequence detection equalizer. The bit error rate can generally be split into 2 components: the probability of all overlapping windows decoding the incorrect bit, and the probability that some windows decode incorrect bits and the conflict resolution mechanism chooses the wrong bit. Let e represent an incorrect bit, and let C represent the conflict resolution function. Then, the BER can be expressed as

$$\begin{aligned} BER = {} & P\left(\bigcap_{i=1}^{N}[W_{N+1-i}(i) = e]\right) \\ & + P\left(\bigcup_{i=1}^{N}[W_{N+1-i}(i) = e] \cap \bigcup_{i=1}^{N}[W_{N+1-i}(i) \neq e] \cap C(W(1), \ldots, W(N)) = e\right) \end{aligned}$$

The first term is a error term that only depends on the window length and number of taps, whereas the second term indicates the effectiveness of a particular conflict resolution mechanism. Another perspective is that given some number of taps and window length, the first term represents the lower-bound BER for all variants. This lower-bound limit on the BER serves as an initial metric of whether the window length and/or the number of taps should be increased to meet the desired bit error rate.

Note that each decoded window is some function f of the consecutive noise samples, which are assumed to be independent and identically distributed according to a Gaussian distribution:

$$W(i) = f(v_n[i - L + 1], v_n[i - L + 2], \ldots, v_n[i])$$

This results in a dependence between events of overlapping decoded windows, except for the trivial case when $L = 1$. Given such dependencies involving Gaussian random variables, BER equations for these MLSE algorithms often lack closed-form expressions. To tackle this issue, a statistical analysis framework was created as a means for the user to codify the error statistics of different equalizers, thereby allowing machines to handle all the computations.

Since this framework must support the modeling of diverse equalization schemes, a few key features are implemented. First, real-world, continuous distributions like thermal noise are discretized or sampled into different bins at regular intervals, and then normalized to have a summed probability of 1. This "binning" results in a probability mass function (PMF), which can be used in-place of the continuous PDF to ease computation, especially for operations that involve multiple distributions like convolution or conditioning.

To control the range and resolution of the discretization, two parameters are introduced, respectively: the number of standard deviations (n_σ), and the number of bins per standard deviation (b_σ). Users can choose values based on their application (specifically, the BER of

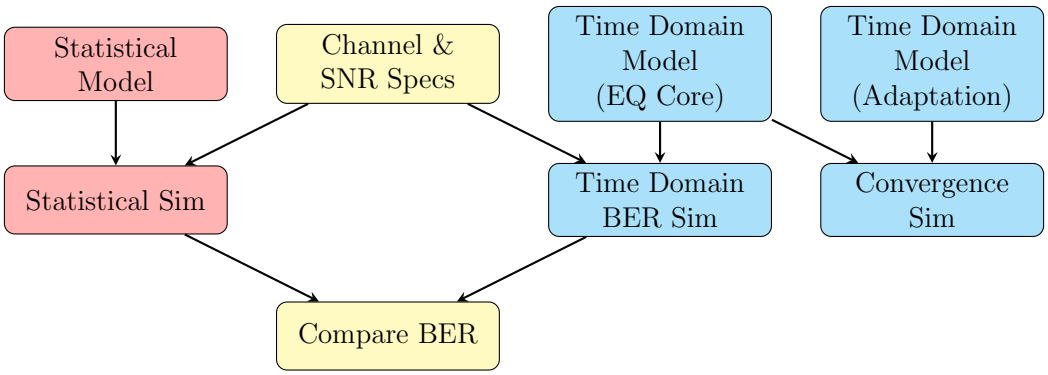


Figure 2.4: Framework flow for statistical (red) and time-domain (cyan) verification

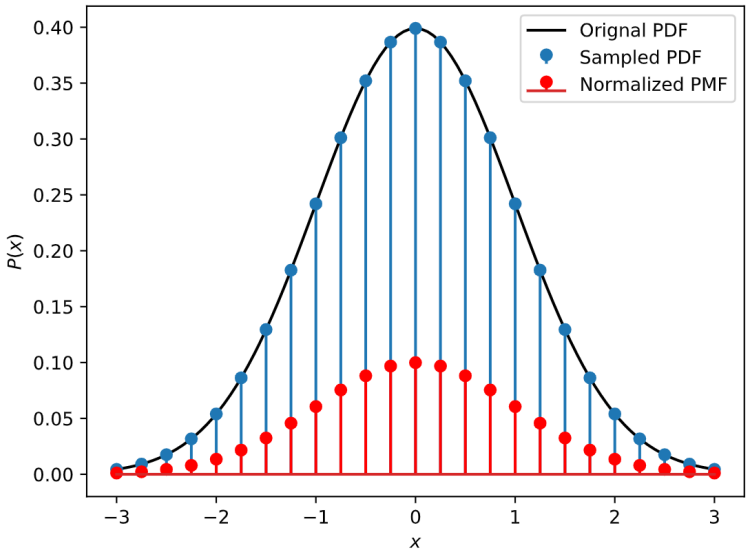


Figure 2.5: Discretization of a Gaussian distribution for $n_\sigma = 3, b_\sigma = 4$

interest). For example, to capture error rates of $\sim 10^{-12}$, n_σ should at least be 7, which results in a one-sided tail probability of 1.28×10^{-12} for Gaussian distributions. Increasing n_σ and b_σ improve the accuracy of the statistical results at the cost of analysis runtime. To the first order, if one is interested in how L random variables affect the probability of an event (as is the case with window decoding), an approach that iterates through all possible combinations of variable values results in a runtime $\sim (2n_\sigma b_\sigma)^L$. Practically, choosing n_σ, b_σ to be around 10-15 was a reasonable compromise between statistical accuracy and runtime for low ($< 10^{-12}$) BER rate applications.

Second, utility functions are implemented to deal with conditional distribution functions. For instance, consider the probability expression $P(X > Y \cap X > Z)$, where X, Y, and Z

are arbitrary, independent, discretized random variables. A brute-force approach to this calculation would involve a 3-dimensional iteration over the domains of each variable $\mathcal{F}_X$, $\mathcal{F}_Y$, and $\mathcal{F}_Z$:

$$P(X > Y \cap X > Z) = \sum_{x \in \mathcal{F}_X} \sum_{y \in \mathcal{F}_Y} \sum_{z \in \mathcal{F}_Z} \left[P(X = x)P(Y = y)P(Z = z) \begin{cases} 1 & \text{if } x > y \cap x > z \\ 0 & \text{otherwise} \end{cases} \right]$$

If X, Y, and Z have B_X, B_Y, and B_Z bins, respectively, then the above would take $\sim B_X B_Y B_Z$ time to compute. However, the runtime can be significantly reduced by applying conditionality. If we define X' to be the distribution of X conditioned on the event that $X > Y$, then the probability can be rewritten as

$$P(X > Y \cap X > Z) = P(X > Y) \cdot P(X > Z | X > Y) = P(X > Y) \cdot P(X' > Z)$$

The individual components can be computed as follows, where $\mathcal{F}_{X'}$ represents the domain of X':

$$P(X > Y) = \sum_{x \in \mathcal{F}_X} [P(X = x) \cdot P(Y < x)] = \sum_{x \in \mathcal{F}_X} [P(X = x) \cdot F_Y(x)]$$
$$P(X' = x) = \frac{P(X = x) \cdot F_Y(x)}{P(X > Y)}$$
$$P(X' > Z) = \sum_{x \in \mathcal{F}_{X'}} [P(X' = x) \cdot P(Z < x)]$$

Note the common term between $P(X > Y)$ and the probability mass function of X'. If we define this common term as a "weight" function $W_{X'}$, we can re-define the above equations:

$$W_{X'}(x) \equiv P(X = x) \cdot F_Y(x) \tag{2.11}$$
$$P(X > Y) = \sum_{x \in \mathcal{F}_X} W_{X'}(x) = \sum_{x \in \mathcal{F}_{X'}} W_{X'}(x) \tag{2.12}$$
$$P(X' = x) = \frac{W_{X'}(x)}{\sum_{x \in \mathcal{F}_{X'}} W_{X'}(x)} \tag{2.13}$$

In other words, $W_{X'}$ represents an unnormalized "probability" distribution of X', whose sum is the conditioned event $P(X < Y)$. If the CDF of Y (F_Y) is pre-computed, the overall runtime of this approach is on the order of B_X, which is a significant improvement over the initial approach.

In addition to the statistical modeling, the analysis framework supports time-domain simulation. Time-domain analysis achieves two core purposes. Because statistical models of equalizers are more equation-driven and prone to modeling errors, time-domain behavioral models serve as a reference to sanity check that the statistical model yields equivalent error rates across different values of SNR. However, since the minimum BER detectable by a

time-domain simulation scales inversely linearly with the number of simulation cycles, time-domain models typically are only useful to estimate error rates as low as $\sim 10^{-6}$. The second advantage of time-domain support is to verify closed-loop adaptation schemes, as each equalizer will need to have its own dedicated adaptation engine that should account for any circuit-induced non-linearities.

2.3 Explored Architectures

The feedforward MLSE algorithm would be proposed as an energy-efficient alternative to the DFE in the receiver equalization. To manage the MLSE's exponentially growing complexity, the design space was limited to one-tap post-cursor equalizers. Then, different window lengths and conflict resolution mechanisms were considered to develop different MLSE designs.

For each design, its error statistics were computed assuming a one-tap postcursor channel with main cursor $h_0 = 1$ and first post-cursor $h_1 = \alpha$. The noise variance σ_n was varied to generate a BER vs. SNR curve, where SNR is calculated as $\left(\frac{h_0}{\sigma_n}\right)^2$. The proposed design's BER vs. SNR curves were then compared with those of the DFE for different values of α to determine viability from an error statistical perspective.

The effect of error propagation was ignored in the statistical modeling of the DFE as it has minimal impact on the BER at low error rates [7]. Neglecting error propagation and residual ISI, the DFE error rate is simply $F_n(-h_0) = F_n(-1)$. Note that this is just the ideal BER for symbol-by-symbol equalizers from (2.10) because DFE does not enhance noise.

2.3.1 Window Length 1

Error terms in (2.5) can be substituted with the definitions from (2.1):

$$W(n) = \underset{x\in\{0,1\}^2}{\arg\min} \begin{cases} (v_r[n] - (1+\alpha))^2 & \text{if } x = [1,1] \\ (v_r[n] - (1-\alpha))^2 & \text{if } x = [0,1] \\ (v_r[n] - (-1+\alpha))^2 & \text{if } x = [1,0] \\ (v_r[n] - (-1-\alpha))^2 & \text{if } x = [0,0] \end{cases} \tag{2.14}$$

$$= \underset{x\in\{0,1\}^2}{\arg\min} \begin{cases} \alpha - (1+\alpha)v_r[n] & \text{if } x = [1,1] \\ -\alpha - (1-\alpha)v_r[n] & \text{if } x = [0,1] \\ -\alpha + (1-\alpha)v_r[n] & \text{if } x = [1,0] \\ \alpha + (1+\alpha)v_r[n] & \text{if } x = [0,0] \end{cases} \tag{2.15}$$

Expanding and removing common terms in the arg min arguments allows one to simplify (2.14) to (2.15). Then, one can inspect conditions in which one sequence results in a smaller

error than another. For example, consider the comparison between $e_{11}[n]^2$ and $e_{01}[n]^2$:

$$\begin{aligned} e_{11}[n]^2 &\overset{?}{<} e_{01}[n]^2 \\ \alpha - (1+\alpha)v_r[n] &\overset{?}{<} -\alpha - (1-\alpha)v_r[n] \\ 2\alpha &\overset{?}{<} 2\alpha v_r[n] \\ v_r[n] &\overset{?}{>} 1 \end{aligned}$$

One can apply the same approach to other sequences to further simplify the window function:

$$W(n) = \begin{cases} [1,1] & \text{if } v_r[n] > 1 \\ [0,1] & \text{if } 0 < v_r[n] < 1 \\ [1,0] & \text{if } -1 < v_r[n] < 0 \\ [0,0] & \text{if } v_r[n] < -1 \end{cases} \tag{2.16}$$

Another way to arrive at (2.16) is by graphically visualizing equidistant lines between the 4 noiseless expected states, as shown in Figure 2.6. Since the MLSE attempts to minimize the Euclidean distance, the equidistant lines represent thresholds between different sequences.

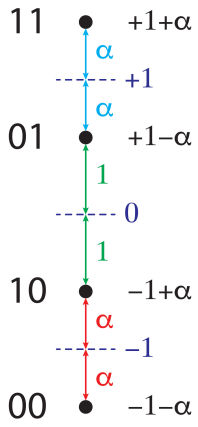


Figure 2.6: Window length 1 decoding with +1/0/-1 thresholds (dashed)

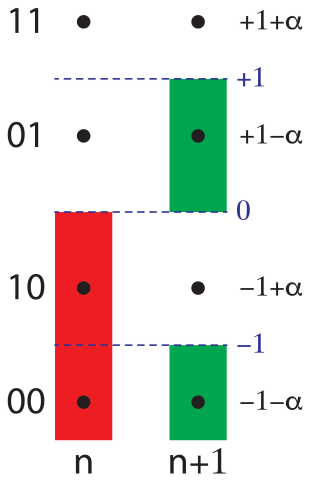


Figure 2.7: Decoded window conditions for $W(n)$ (red) and $W(n+1)$ (green) for lower-bound BER

As mentioned earlier, the error statistics depend on the exact conflict resolution mechanism. However, a lower-bound BER for all window length 1 designs can be calculated to sanity check whether a window length 1 equalizer is viable to begin with. The lower-bound BER can be computed by considering the ideal case that the conflict resolution mechanism

always decides the correct bit when a conflict between overlapping windows arises. Thus, the BER in this case is simply the probability that all windows decode the incorrect bit.

Without loss of generality, assume that at sample n, the transmitted data symbol $d_t[n]$ is 1. Then, the lower-bound BER (as illustrated in Figure 2.7) can be expressed as

$$\begin{aligned} BER &= P\left(W_2(n) = 0 \cap W_1(n+1) = 0\right) \\ &= P\left(W_2(n) = 0\right) \cdot P\left(W_1(n+1) = 0\right) \\ &= P(v_r[n] < 0) \cdot P(v_r[n+1] < -1 \cup 0 < v_r[n+1] < 1) \end{aligned} \tag{2.17}$$

Recall that with for a window length of 1, decoding the overlapping windows is independent. This expression simplifies to[1]

$$BER \approx \frac{1}{4}[F_n(-1-\alpha) + F_n(-1+\alpha)][2F_n(-\alpha) + F_n(2-\alpha) - F_n(1-\alpha)] \tag{2.18}$$

As shown in Figure 2.8, the lower-bound BER of the window length 1 MLSE is higher than that of the DFE across a wide range of α values, which correspond to different channel losses. The DFE would out-perform any window length 1 architecture, as any realizable window length 1 architecture would have greater error probabilities than the lower-bound model. To find a viable alternative to the DFE, longer window lengths should be considered.

2.3.2 Window Length 2

Applying the same graphical approach as in the previous section, one can reduce the window decoding function for window length 2 architectures to the following set of lines, as shown in Figure 2.9. Since the decoding is a function of two adjacent voltage samples, the plot shows the current sample on the x-axis and the previous sample on the y-axis. The black dots represent ideal voltage states for each possible data pattern and form a subset of the space $(\pm 1 \pm \alpha, \pm 1 \pm \alpha)$. The dashed lines represent equidistant lines between the ideal states, where the exact linear equation for each threshold is expressed below:

$$\begin{aligned} &① : v_r[n] = \alpha \\ &② : v_r[n] = -\alpha \\ &③ : v_r[n-1] = 1 \\ &④ : v_r[n-1] = -1 \\ &⑤ : v_r[n] = v_r[n-1] \\ &⑥ : v_r[n] = -\frac{1-\alpha}{\alpha} v_r[n-1] + 1 \\ &⑦ : v_r[n] = -\frac{1-\alpha}{\alpha} v_r[n-1] - 1 \\ &⑧ : v_r[n] = -\frac{1}{\alpha} v_r[n-1] \end{aligned}$$

[1]Refer to Appendix A.1 for the full derivation.

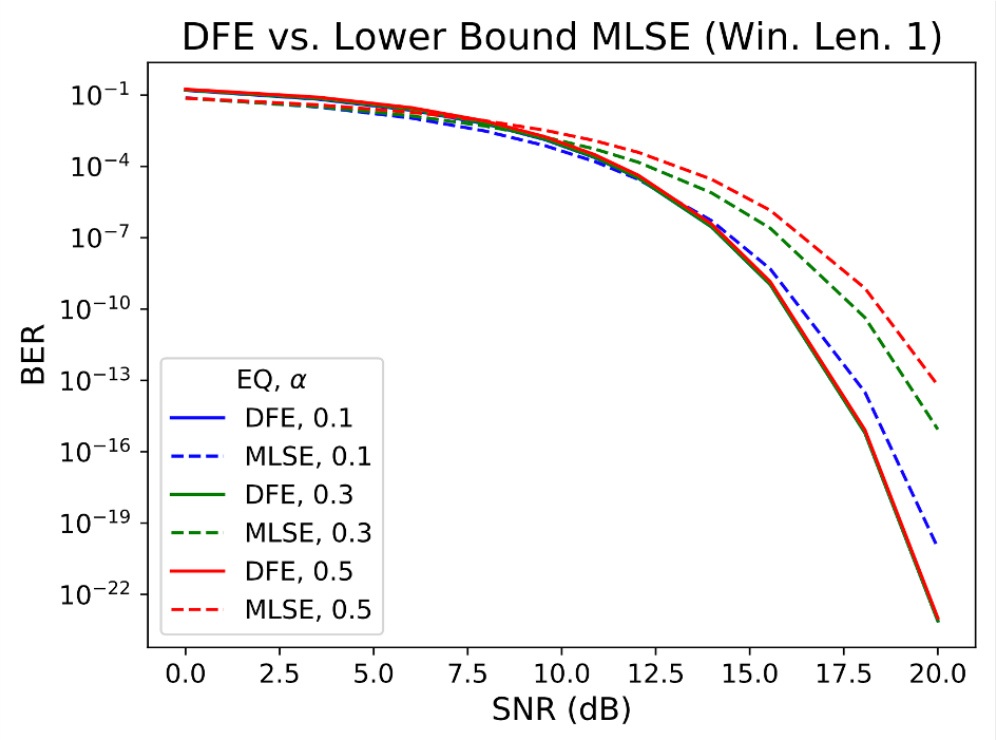


Figure 2.8: Error statistics comparison between DFE and lower-bound MLSE (window length 1)

Computing the lower-bound error statistics of the window length 2, however, is trickier than those of the window length 1 due to the dependence in decoded windows. For a 1-tap window length 2 architecture, there are 3 overlapping windows for $d_t[n]$:

$$\begin{aligned} W(n) &= f(v_r[n], v_r[n-1]) = g(d_t[n], d_t[n-1], d_t[n-2], v_n[n], v_n[n-1]) \\ W(n+1) &= f(v_r[n+1], v_r[n]) = g(d_t[n+1], d_t[n], d_t[n-1], v_n[n+1], v_n[n]) \\ W(n+2) &= f(v_r[n+2], v_r[n+1]) = g(d_t[n+2], d_t[n+1], d_t[n], v_n[n+2], v_n[n+1]) \end{aligned}$$

A brute-force approach would require 9 nested loops—5 for the transmitted data $d_t[n-4], ..., d_t[n]$ and 4 for the noise $v_n[n-3], ..., v_n[n]$. For a discretized Gaussian distribution of B bins, the total number of iterations would be $2^5 B^4$, which results in 100B to 1T iterations for typical number of bins. To reduce computational complexity, conditioning is applied to the noise distributions by partitioning the window decoding space, as illustrated in Figure 2.10. If the decoding is broken up at horizontal and vertical lines of threshold $0, \pm\alpha, \pm 1$, each rectangular region has at most 2 possible sequences that can be decoded.

Let the one-dimensional partitioning be represented by the following set of intervals $A \in \{(-\infty, -1), (-1, -\alpha), (-\alpha, 0), (0, \alpha), (\alpha, 1), (1, \infty)\}$, where each tuple contains the lower and upper bounds of the interval. Suppose again without loss of generality that the transmitted

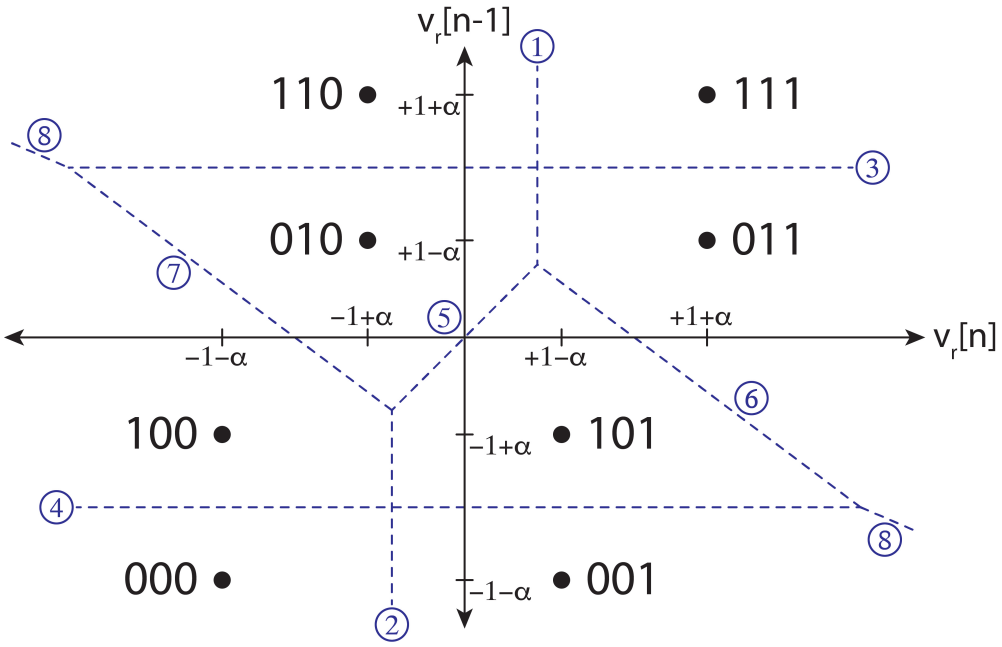


Figure 2.9: Window length 2 decoding with labeled thresholds (dashed)

data symbol $d_t[n]$ is 1. As the transmitted data spans 5 UI across all overlapping windows, the 5-bit transmitted sequence d_x can be expressed as

$$d_x = [d_t[n-2], d_t[n-1], d_t[n], d_t[n+1], d_t[n+2]] = [x_1, x_2, 1, x_3, x_4]$$

where $x = [x_1, x_2, x_3, x_4]$ represents a random vector of neighboring transmitted bits.

The lower-bound BER can be rewritten as a summation across all possible interval and data pattern combinations:

$$\begin{aligned} BER &= P\left(W_3(n) = 0 \cap W_2(n+1) = 0 \cap W_1(n+2) = 0\right) \\ &= \frac{1}{16} \sum_{x \in \{0,1\}^4} \sum_{(y_{min}, y_{max}) \in A^4} p_{r,1} p_{r,2} p_{r,3} p_{r,4} p_{w,0,0} p_{w,1,0} p_{w,2,0} \end{aligned} \tag{2.19}$$

For $i \in \{1, 2, 3, 4\}$, $p_{r,i}$ represents the probability that the received voltage $v_r[n+i-2]$ falls in the given interval $(y_{min,i}, y_{max,i})$:

$$p_{r,i} = P(v_r[n+i-2] \in (y_{min,i}, y_{max,i}) | d_t[n+i-2] = d_{x,i+1}, d_t[n+i-3] = d_{x,i})$$

For $j \in \{0, 1, 2\}$, $p_{w,j,b}$ represents the probability that $W_{3-j}(n+j)$ (the overlapping bit in a decoded window) is the binary value $b \in \{0, 1\}$:

$$\begin{aligned} p_{w,j,b} = P(W_{3-j}(n+j) = b | & d_t[n+j] = d_{x,j+3}, d_t[n+j-1] = d_{x,j+2}, d_t[n+j-2] = d_{x,j+1}, \\ & v_r[n+j] \in (y_{min,j+2}, y_{max,j+2}), \\ & v_r[n+j-1] \in (y_{min,j+1}, y_{max,j+1})) \end{aligned}$$

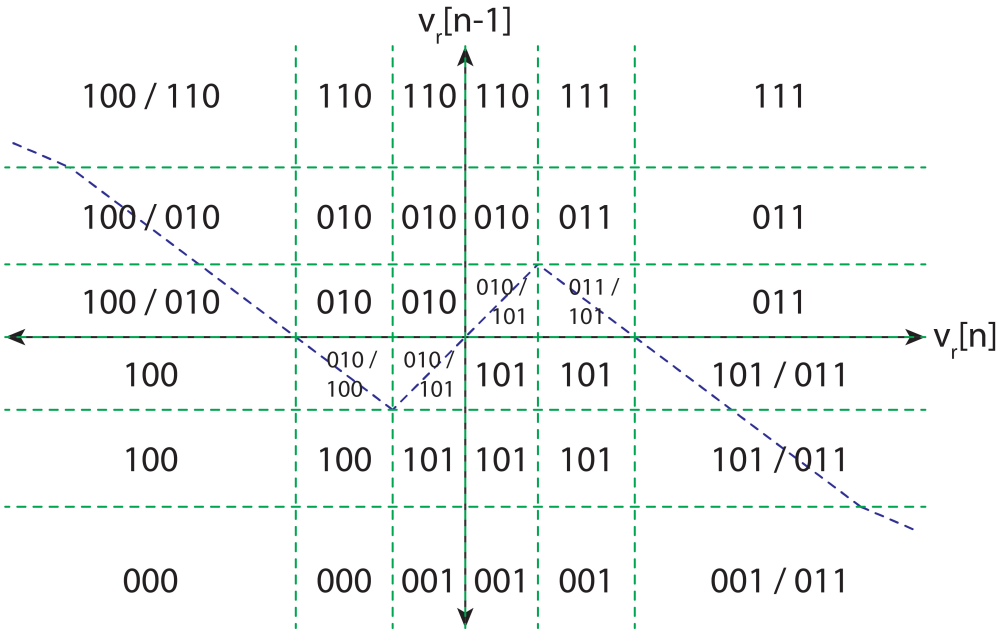


Figure 2.10: A partition of the window length 2 decoding into horizontal and vertical lines (green, dashed)

Since d_t is treated as a constant, $v_r[n+i-2] = v_n[n+i-2] + v_{ideal}$, where v_{ideal} is a direct function of the transmitted bits ($\pm 1 \pm \alpha$). (y_{min}, y_{max}) are also treated as constants, so $p_{r,i}$ are all independent probabilities that condition the distribution of the received voltage according to the inequality $y_{min,i} \leq v_r[n+i-2] \leq y_{max,i}$. Once the conditional distributions of v_r are computed, they can be used to determine $p_{w,j,b}$.

Due to the partitioning strategy, computing $p_{w,j,b}$ is fairly simple. Recall that $p_{w,j,b}$ is related to the decoding of (3-j)th bit in window $W(n+j)$, which depends on $v_r[n+j]$ and $v_r[n+j-1]$. Based on which interval is currently being operated on, there are the following 3 scenarios:

1. Only 1 sequence can be decoded. If the bit of interest is b, then $p_{w,j,b} = 1$ and the received voltage distributions remain unchanged. Otherwise, $p_{w,j,b} = 0$, zeroing out the whole term in the (2.19) summation.

2. 2 sequences can be decoded, but the bit of interest is same for both sequences. Then, this reduces to the above scenario where $p_{w,j,b}$ is either 1 or 0.

3. 2 sequences can be decoded, and the bit of interest is different for the sequences. In this case, this region is linearly split between the two possible sequences by an inequality in the form of $v_r[n+j] < A \cdot v_r[n+j-1] + B$. Using the weight function approach from (2.11) (2.12) (2.13), $p_{w,j,b}$ can be computed with conditional distributions on the received voltages.

However, $p_{w,j,b}$ are not all independent of each other. $p_{w,0,b}$ and $p_{w,2,b}$ are independent since the former depends on $v_r[n]$ and $v_r[n-1]$ while the latter depends on $v_r[n+2]$ and $v_r[n+1]$. $p_{w,1,b}$ shares a dependence with the other 2 terms because it shares a dependence of $v_r[n+1]$ with $p_{w,2,b}$ and a dependence of $v_r[n]$ with $p_{w,0,b}$. By first computing $p_{w,0,b}$ and $p_{w,2,b}$, the resulting conditioned distributions of $v_r[n]$ and $v_r[n+1]$ can then be directly applied to $p_{w,1,b}$.

Compared to the DFE, "lower-bound" window length 2 MLSE has better or comparable error rates for most values of α (Figure 2.11). Thus, window length 2 architectures could be promising alternatives to the DFE, but the key challenge is to develop a conflict resolution mechanism that is simple to implement in hardware while still retaining good error statistics.

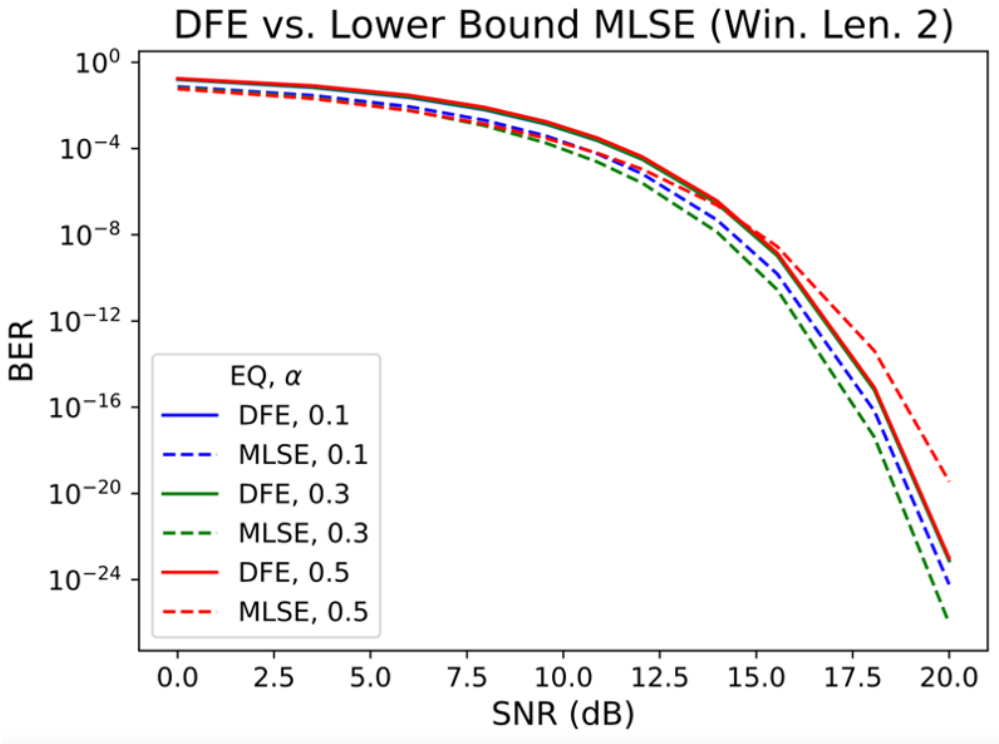


Figure 2.11: Error statistics comparison between DFE and lower-bound MLSE (window length 2)

2.3.2.1 Majority Vote

Note that for window length 2, there are an odd number of overlapping windows that give information about the same bit. In the event of a conflict, the majority vote is the simplest approach since it only requires a few additional logic gates to resolve the conflict.

The error statistics of the majority vote design can be computed similarly to the lower bound case. Whereas the lower bound version required all windows to give the wrong bit,

the majority vote design would error if at least 2 windows gave the wrong bit. In other words, the BER of the majority vote is

$$\begin{aligned} BER &= P\left(W_3(n)=0 \cap W_2(n+1)=0 \cap W_1(n+2)=0\right) \\ &+ P\left(W_3(n)=1 \cap W_2(n+1)=0 \cap W_1(n+2)=0\right) \\ &+ P\left(W_3(n)=0 \cap W_2(n+1)=1 \cap W_1(n+2)=0\right) \\ &+ P\left(W_3(n)=0 \cap W_2(n+1)=0 \cap W_1(n+2)=1\right) \\ &= \frac{1}{16} \sum_{x\in\{0,1\}^4} \sum_{(y_{min},y_{max})\in A^4} p_{r,1}p_{r,2}p_{r,3}p_{r,4}\big(p_{w,0,0}p_{w,1,0}p_{w,2,0} + p_{w,0,1}p_{w,1,0}p_{w,2,0} \\ &\qquad + p_{w,0,0}p_{w,1,1}p_{w,2,0} + p_{w,0,0}p_{w,1,0}p_{w,2,1}\big) \end{aligned} \tag{2.20}$$

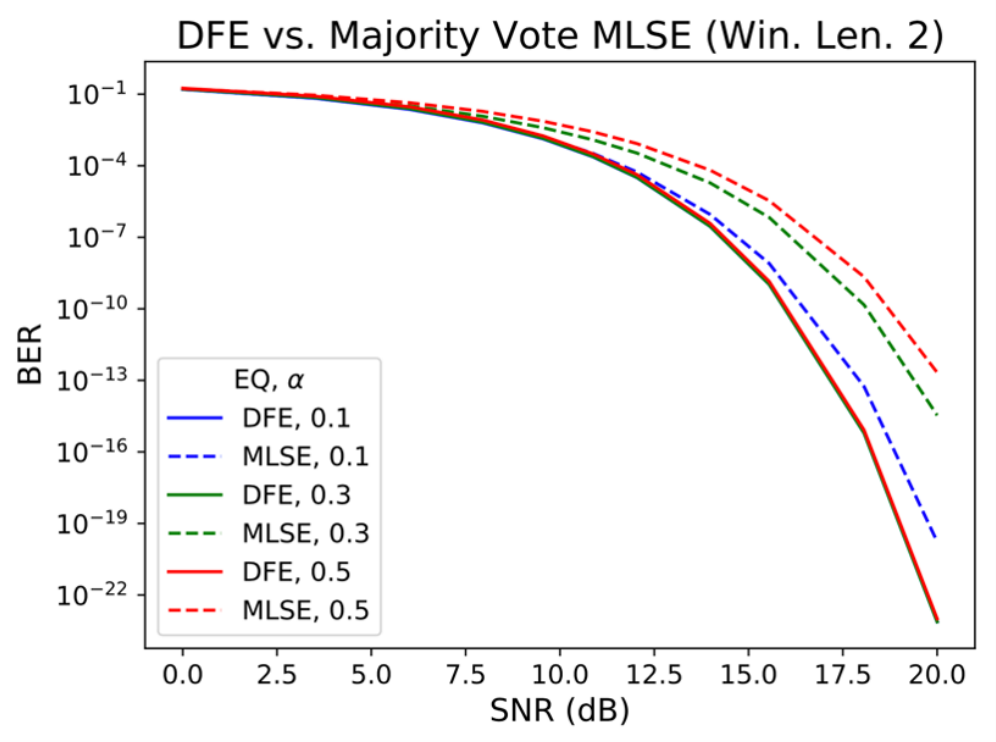


Figure 2.12: Error statistics comparison between DFE and majority vote MLSE (window length 2)

As shown in Figure 2.12, the majority vote design yields a higher BER than the DFE across all values of α. The main disadvantage of the majority vote method is that all overlapping windows are weighted equally. However, lower loss channels typically have $\alpha < 0.5$, so each voltage sample at the receiver naturally retains more information about the currently transmitted bit than the previous bit. As α decreases toward 0, the earliest window $W(n)$ would contain most information about the current bit to be resolved,

while the latest window $W(n+2)$ contains least information and is more error-prone. In fact, of the 4 error cases in the majority vote analysis, the the dominant error term is $P\left(W_3(n)=1 \cap W_2(n+1)=0 \cap W_1(n+2)=0\right)$.

2.3.2.2 Single Window

The above phenomenon suggests that for lower loss channels, the earlier windows should be weighed more heavily than later windows when resolving conflicts. As a result, the proposed "single window" design resolves conflicts by always using the bit decoded by the earliest window. Doing so entirely removes interaction between overlapping windows, which simplifies both the statistical analysis and complexity of this equalizer. Since overlapping windows are no longer considered, each window only needs to decode its last bit, not the entire sequence. In other words, the window decoding reduces from Figure 2.9 to Figure 2.13.

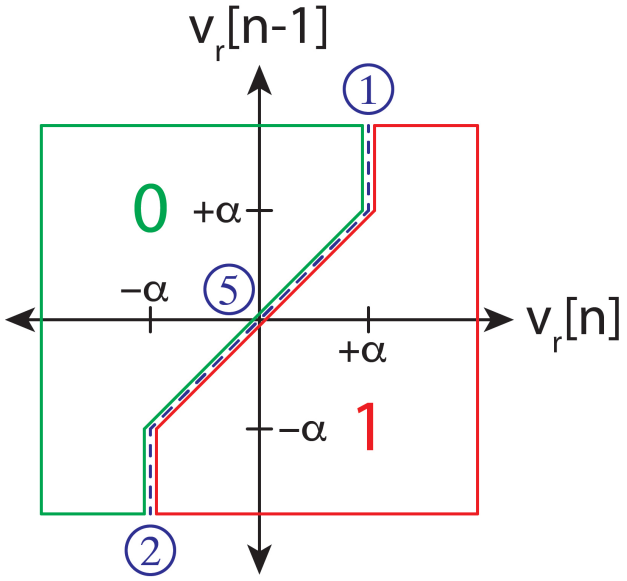


Figure 2.13: Single window decoding for window length 2. A single bit is decoded: 1 if the received samples are in the red region, 0 if the green region

Suppose we define a slice function to mimic a comparator or slicer function:

$$\text{slice}(x) \equiv \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

Then, the 3-line piecewise window decoding function $W_{sing}(n)$ can be mathematically

represented as

$$W_{sing}(n) = \begin{cases} 1 & \text{if } v_r[n] > h_0 \\ 0 & \text{if } v_r[n] < -h_0 \\ \text{slice}(v_r[n] - v_r[n-1]) & \text{otherwise} \end{cases} \tag{2.22}$$

This function can be realized by using minimal circuitry consisting of a few comparators, adders, and logic gates shown in Figure 2.14. The $\pm\alpha$ slicer thresholds are similar to those of the 1-tap loop-unrolled DFE, but an additional 2-tap FIR path is added. As a result, the adaptation algorithm of $\pm\alpha$ thresholds is reused from that of the 1-tap loop-unrolled DFE, where the data thresholds are interpolated from an additional error slicer that adapts for $1 \pm \alpha$ data levels (dLevs) based on 11/01 pattern filtering. The adaptation scheme will be discussed in more detail in Section 4.3.1.

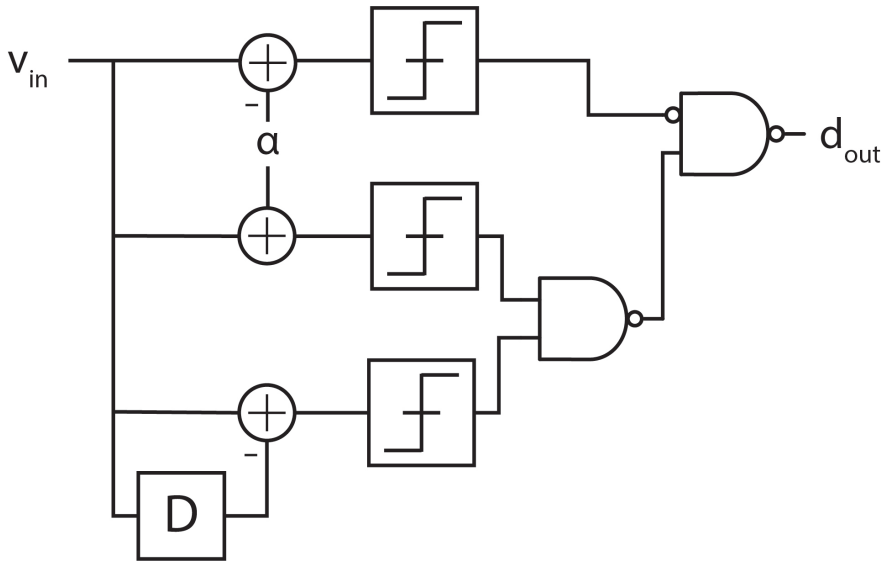


Figure 2.14: Single window length 2 block diagram

With the reduced decoding function and lack of interaction between overlapping windows, the bit error rate can be computed as the following (again assuming the transmitted bit is 1):

$$\begin{aligned} BER &= P(W_3(n) = 0) = P(v_r[n] < -\alpha \cup (-\alpha < v_r[n] < \alpha \cap v_r[n-1] > v_r[n])) \\ &= \frac{1}{4} \sum_{x \in \{0,1\}^2} [P(v_r[n] < -\alpha | d_t[n-2] = x_1, d_t[n-1] = x_2, d_t[n] = 1) \\ &\qquad + P(-\alpha < v_r[n] < \alpha \cap v_r[n-1] > v_r[n] | d_t[n-2] = x_1, \\ &\qquad\qquad d_t[n-1] = x_2, d_t[n] = 1)] \end{aligned} \tag{2.23}$$

As shown in Figure 2.15, this design achieves comparable error rates to the DFE for $\alpha \leq 0.3$ but falls off for larger values of α.

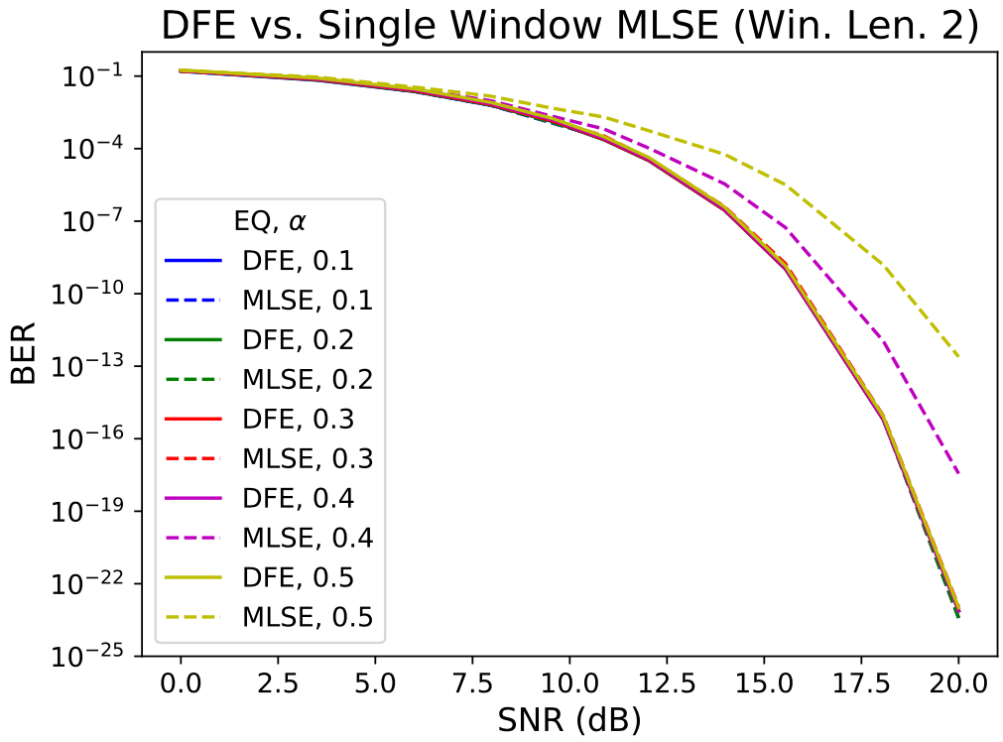


Figure 2.15: Error statistics comparison between DFE and single window MLSE (window length 2)

2.3.3 Window Length 3

The window decoding function showed a drastic increase in complexity from window length 1 to window length 2 architectures. Given the MLSE's known tendency for exponentially increasing complexity, window length 3 architectures seemed infeasible at first. However, with the discovery of the single window architecture, which could greatly simplify the decoding function, the single window approach was extended to window length 3 to determine whether the error statistics would improve for α values beyond 0.3.

The window decoding function for the single window is shown below:

$$W_{sing}(n) = \begin{cases} 1 & \text{if } v_r[n] > h_1 \\ 0 & \text{if } v_r[n] < -h_1 \\ f(v_r) & \text{otherwise} \end{cases}$$

Note that the general form is the same as the window length 2 version (2.22). However, the "otherwise" condition is now replaced by a much more complex function f, as shown below:

$$f(v_r) = \begin{cases} f_1(v_r) & \text{if } (a_{(-\infty,-h_0)} \cap c_{>0}) \cup (a_{(h_0,\infty)} \cap c_{(2h_1,\infty)}) \cup (a_{(-h_0,h_0)} \cap b_{(h_1,\infty)}) \\ f_2(v_r) & \text{if } (a_{(h_0,\infty)} \cap c_{(-\infty,0)}) \cup (a_{(-\infty,-h_0)} \cap c_{(-infty,-2h_1)}) \cup (a_{(-h_0,h_0)} \cap b_{(-\infty,-h_1)}) \\ f_3(v_r) & \text{if } (a_{(h_0,\infty)} \cap c_{(0,2h_1)}) \\ f_4(v_r) & \text{if } (a_{(-\infty,-h_0)} \cap c_{(-2h_1,0)}) \\ f_5(v_r) & \text{if } (a_{(-h_0,h_0)} \cap b_{(-h_1,h_1)}) \end{cases}$$

$$f_1(v_r) = \text{slice}((1-\alpha)v_r[n] - v_r[n-1] + h_1)$$
$$f_2(v_r) = \text{slice}((1-\alpha)v_r[n] - v_r[n-1] - h_1)$$
$$f_3(v_r) = \text{slice}((1-\alpha)(v_r[n] - v_r[n-1]) + v_r[n-2] - h_1)$$
$$f_4(v_r) = \text{slice}((1-\alpha)(v_r[n] - v_r[n-1]) + v_r[n-2] + h_1)$$
$$f_5(v_r) = \text{slice}(v_r[n] - v_r[n-1] + v_r[n-2])$$

$$a_{(x,y)} \equiv v_r[n-2] \in (x,y)$$
$$b_{(x,y)} \equiv \alpha v_r[n-1] + (1-\alpha)v_r[n-2] \in (x,y)$$
$$c_{(x,y)} \equiv \alpha v_r[n-1] + v_r[n-2] \in (x,y)$$

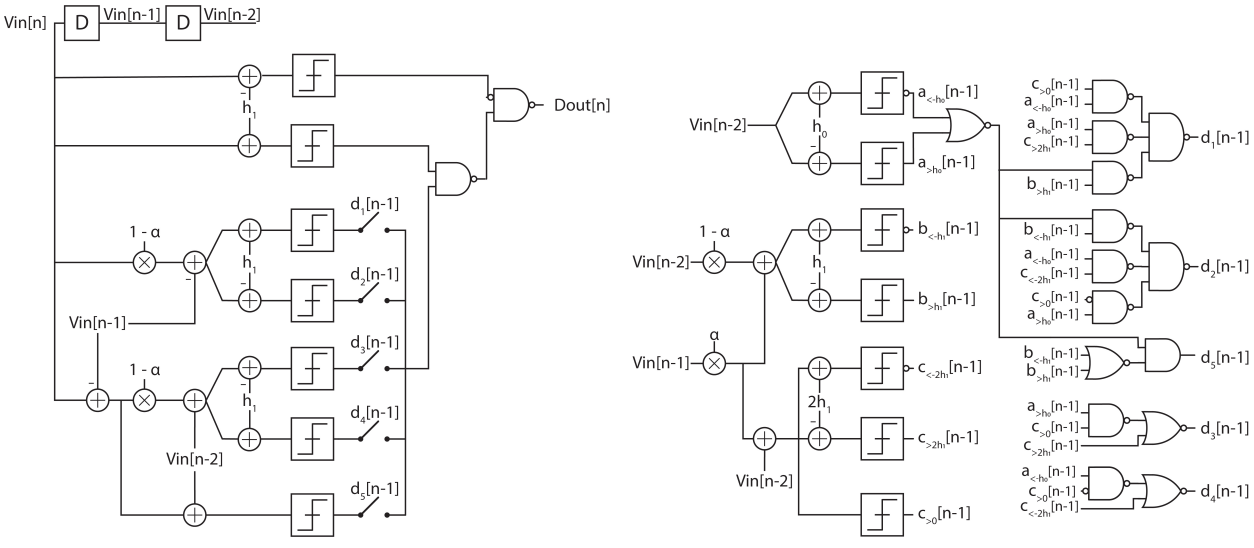


Figure 2.16: Single window length 3 block diagram

The equivalent block diagram is depicted in Figure 2.16. Compared to the window length 2 version, the new decoding function requires $>$ 4x comparators, which tend to be the most power-hungry.

The error statistics of the window length 3 variant is shown in Figure 2.17. This MLSE design now achieves similar BER as the DFE for $\alpha < 0.5$. However, this comes at a significant design complexity tradeoff.

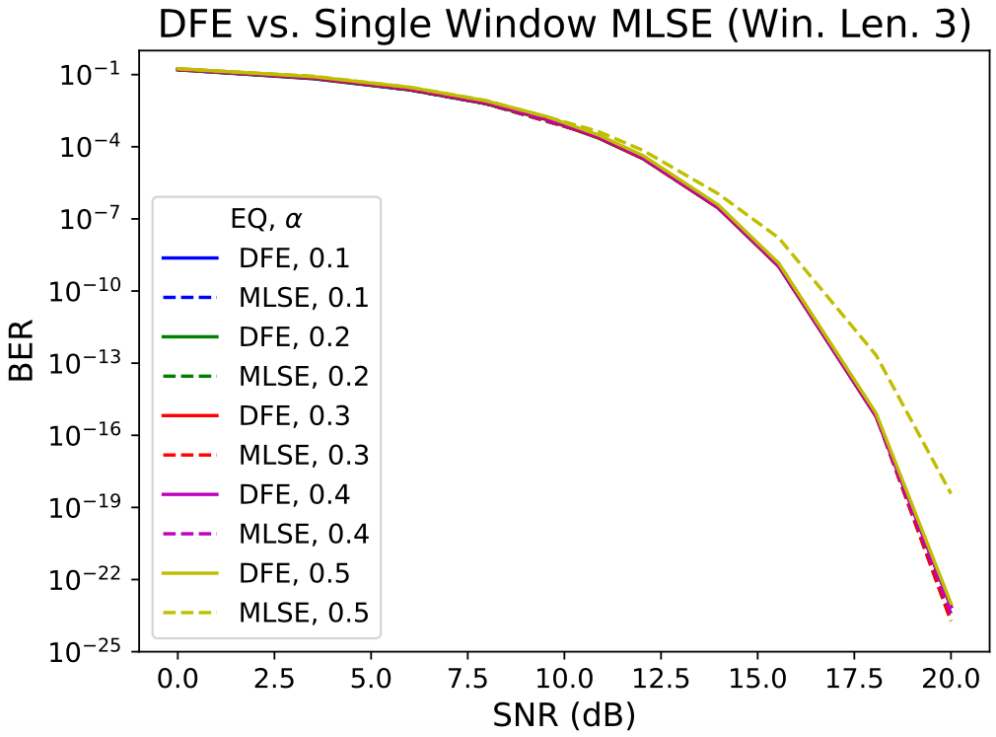


Figure 2.17: Error statistics comparison between DFE and single window MLSE (window length 3)

2.4 Equalizer Comparison and Conclusions

The MLSE algorithm is a powerful equalization algorithm that can theoretically achieve excellent error statistics. On the other hand, applying MLSE for ultra-high-throughput, energy-efficient links requires a simplification of the algorithm and removal of feedback loops, both of which degrade the statistical performance. Several variants of the feedforward MLSE were explored for different window lengths and conflict resolution mechanisms. Of these, the most promising designs were the single window architectures.

For given target channels and bit error rates, the energy-delay tradeoffs are compared across three equalizer designs: 1-tap loop-unrolled DFE, single window MLSE for length 2,

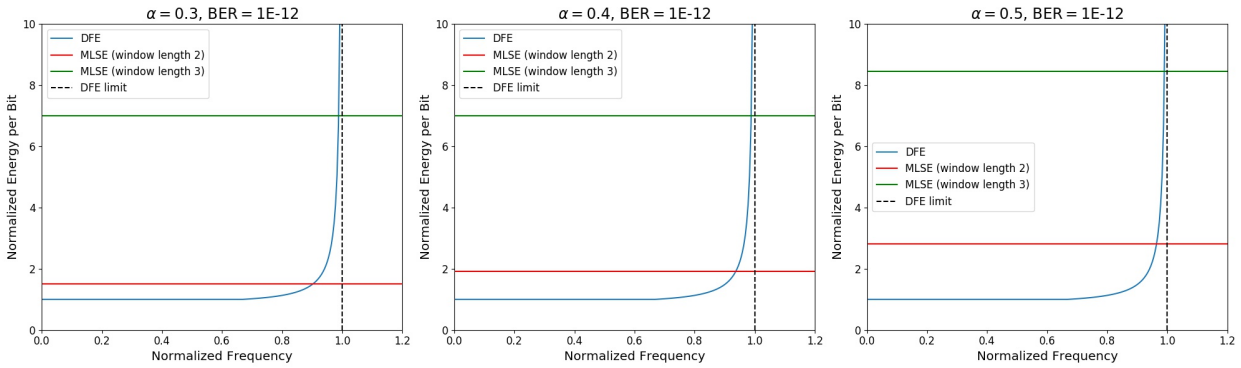


Figure 2.18: Equalizer tradeoffs

and single window MLSE for length 3 (Figure 2.18). The DFE's energy efficiency is modeled to asymptotically reach a maximum frequency limit, which is set by the critical path of the feedback loop and is technology-dependent. At high frequencies, self-loading effects dominate the overall power, resulting in the asymptotical behavior. The impact of self-loading on power consumption for similar circuits is analyzed in more detail in [8]. In contrast, the feedforward MLSE equalizers can be parallelized as needed and are approximated to be constant over frequencies.

Let C_{DFE}, $C_{MLSE,2}$, and $C_{MLSE,3}$ represent the base complexity of the three designs without pushing the design to its limits. In other words, C_{DFE} is normalized to 1 as in Figure 2.18. Assuming that comparators are the dominant source of power consumption, $C_{MLSE,2} \approx 1.5$ and $C_{MLSE,3} \approx 7$.

Then, one can compute the SNR required to achieve the target BER. In particular, the SNR requirements of the MLSE designs can be computed relative to the DFE SNR reference. For error rates of 10^{-12}, the MLSE designs may need anywhere from 0 to 3 dB higher SNR compared to the DFE. Assuming noise is dominated by kT/C noise, MLSE designs would need to be upsized to meet the required SNR. The additional power penalty due to noise consideratoins can be computed as follows:

$$P \propto C \propto \frac{1}{v_n^2} \propto SNR$$

$$N_{MLSE,2} \equiv \frac{P_{MLSE,2}}{P_{DFE}} = \frac{SNR_{MLSE,2}}{SNR_{DFE}} \tag{2.24}$$

$$N_{MLSE,3} \equiv \frac{P_{MLSE,3}}{P_{DFE}} = \frac{SNR_{MLSE,3}}{SNR_{DFE}} \tag{2.25}$$

The nominal power consumption of the MLSE designs are

$$P_{MLSE,2} = C_{MLSE,2}N_{MLSE,2} = 1.5 \times \frac{SNR_{MLSE,2}}{SNR_{DFE}}$$
$$P_{MLSE,3} = C_{MLSE,3}N_{MLSE,3} = 7 \times \frac{SNR_{MLSE,3}}{SNR_{DFE}}$$

Based on the energy-delay tradeoffs, the window length 2 is a better choice than the window length 3 across all cases. Although the window length 3 has superior error statistics for higher loss channels, its base complexity offsets much of its SNR advantages when it comes to evaluating the energy efficiency. This simple comparison, however, assumes that the receiver power is mainly dominated by the equalizer. If the MLSE is not the dominant source of power, the window length 3 could be more attractive in higher loss channels, as the entire receiver would usually need to be upsized to meet the SNR requirement. Furthermore, the asymptotic behavior of the DFE energy efficiency indicates that past its maximum achievable data rate, the feedforward MLSE could be key to implementing the wireline receiver.

Chapter 3

Design Techniques for a 160 Gb/s 1-Tap MLSE Datapath

3.1 Overview

A 160 Gb/s NRZ wireline receiver was designed in a 16 nm FinFET process to evaluate the energy efficiency and future promise of the feedforward MLSE equalizer. As discussed in the last chapter, the single window length 2 MLSE design is chosen for its lower power consumption over the window length 3 version.

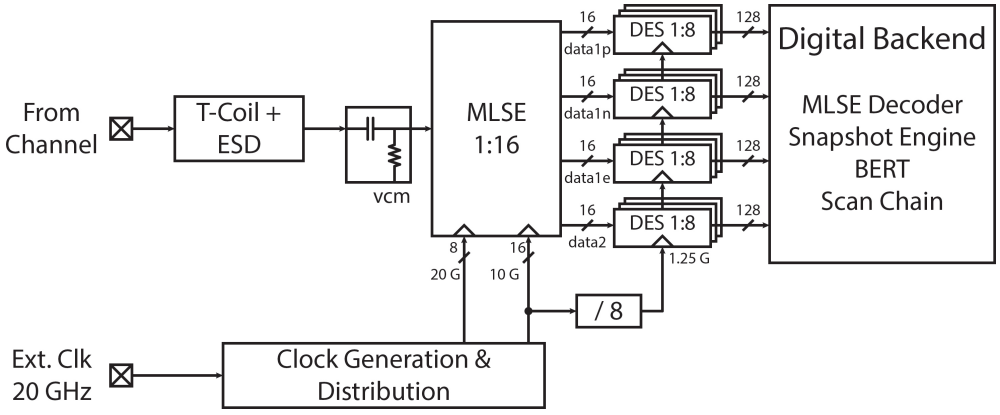


Figure 3.1: RX MLSE block diagram

The receiver block diagram is shown in Figure 3.1. To further reduce power, the equalization is primarily implemented in the analog and mixed-signal (AMS) domain, removing the need for a high-speed ADC. To achieve the desired throughput, the MLSE equalizer is implemented as 16 time-interleaved slices, each effectively working at 10 GS/s. Recall from Figure 2.14 that one slice consists of 3 comparator outputs ("data1p", "data1n", "data2" in Figure 3.1), which are passed into some digital logic for decoding. As is the case with the

1-tap loop-unrolled DFE, an additional error slicer must be added to adapt the $1 \pm \alpha$ dLev. For the MLSE, this is added as a 4th comparator with output "data1e". These signals are then deserialized by an additional factor of 8 and connected to the synthesized logic, which consists of the MLSE decoder and testing modules. A custom-digital decoder operating at 10 GS/s would halve the number of deserializers required, as each slice would only need to output 2 signals—one data and one error for datapath calibration. In addition, the decoder latency would be reduced from ∼128 UI to ∼16 UI. However, latency was not critical to this design, and having all comparator outputs simultaneously available for off-chip testing procedures increased visibility and debug capability for the test chip.

Although parallelism eases the implementation of the feedforward MLSE equalizer, the frontend circuits must still operate at full bit rate and can be the bottleneck to link performance. Inductive peaking circuits have thus been used in SerDes frontend designs to improve bandwidth and return loss performance [9, 10]. In particular, this work's passive frontend was designed by Kunmo Kim and features a T-coil design, whose detailed analysis can be found in [11, 12].

As data rates approach the limits of the process node, device and routing parasitic effects can significantly degrade the link performance. Building such a high-speed link requires numerous design iterations, from device sizing to floorplanning. To improve designer productivity, the receiver was designed using a generator framework called Berkeley Analog Generator (BAG) [13, 14]. Once a user codifies a circuit's design process into a configurable generator, this design methodology enables an agile workflow through the automatic creation and post-layout characterization of parameterized instances. Furthermore, complex design rules in advanced process nodes require greater physical design effort to pass design rule checking (DRC). Generated instances will often be DRC-clean with little to no manual changes, making BAG attractive for advanced-node IC designs.

Developing a robust circuit generator, however, can take significant time. If one does not yet exist for the desired circuit, designers must consider the upfront effort of generator development. As a result, the generator-based design flow is most effective for circuits which must be critically optimized for power performance area (PPA) and thus is used extensively in the proposed receiver.

3.2 Analog Datapath Architecture

A 16-way interleaved MLSE equalizer requires a 1:16 frontend deserializer. This deserializer is implemented in 2 stages to reduce loading at the input. Instead of equal 4x deserialization for each stage, the first stage interleaving ratio is set as 8 because distributing octa-rate (20 GHz) clocks would be more energy-efficient than quad-rate (40 GHz) clocks in the given technology node.

The MLSE slice contains 2 parallel integrating paths. The first path consists of a gain stage, which drives the $\pm\alpha$-threshold data comparators. Because the error slicer on this path must compare its input against $1 + \alpha$, the gain stage must have enough linear range to drive

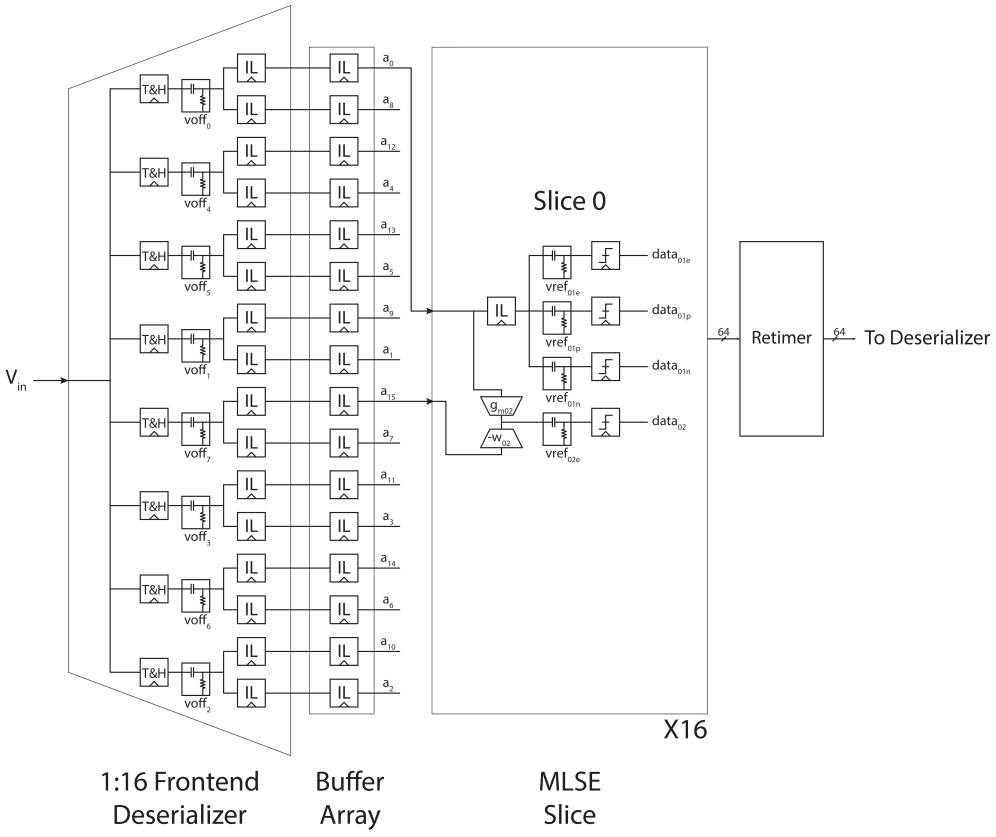


Figure 3.2: MLSE analog datapath block diagram

large signals. The second path consists of the 2-tap FIR, where the previous analog sample is subtracted from the current sample. The output linearity constraints on this FIR/summer path are much more relaxed, as the slicer compares against differential zero.

Due to the 2-tap FIR structure, the deserialized 16-way signals must be shuffled around the slices in the array. The 16 slices are physically tiled in a specific order to minimize the longest routing distance, but the routing tends to increase linearly with number of interleaved ways. The routing parasitics increase the load capacitance driven by the 8:16 deserializer. To manage the larger fanout, an additional gain stage is inserted between the deserializer and MLSE slice as an analog buffer.

Because routing parasitics can often dominate load capacitances and limit circuit performance, the datapath design and floorplaning must be iterated with post-layout simulations. The entire datapath was written as a BAG generator, which allowed for quick design iteration as parameters like device sizing and wire spacing were updated. The design flow is shown in Figure 3.3. First, load capacitances are estimated for each stage of the signal chain. The unit cells are independently optimized with these loading constraints, and the top-level datapath is generated based on each unit cell design. A post-layout transient simulation of

the datapath can be used to extract total load capacitances seen by each stage, which are compared with the original estimates. If there is a difference between the assumed and actual load conditions, the unit cells are re-designed with the new loads and the design process repeats. Once this loop converges and the top-level load capacitances match the unit cell load specifications, a final sign-off check is run and the datapath design is complete.

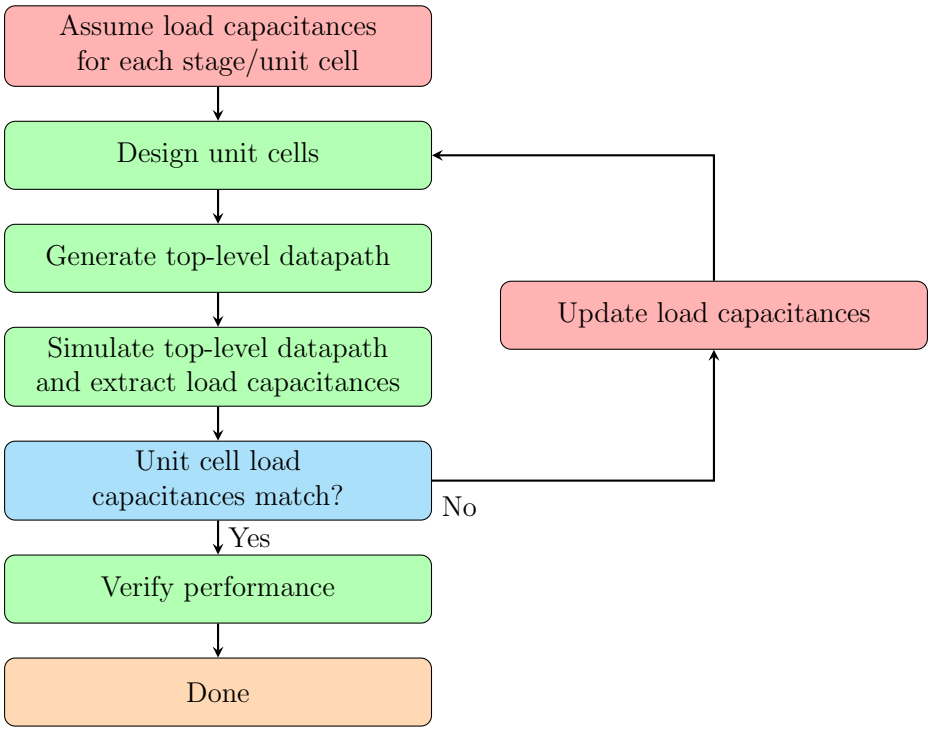


Figure 3.3: Datapath design flow

Various receiver designs have employed current integration techniques to reduce power consumption by $\sim$ 3x [8, 15–22]. As discussed in following sections, gain and delay stages of the datapath are implemented using integrated latches (abbreviated as IL in Figure 3.2).

3.3 Current Integrating Circuits

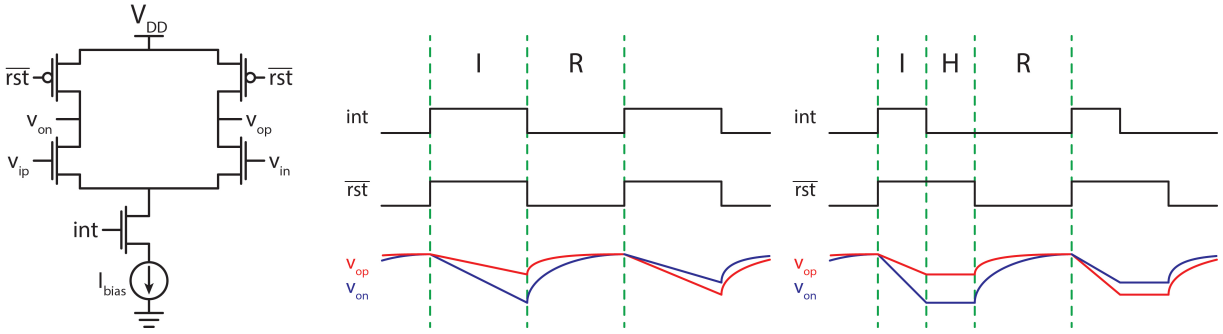
Traditional current integration techniques involve 2 phases of operation (Figure 3.4b). During the first phase, the amplifier integrates current off of the output nodes, generating an output differential voltage with a voltage gain A_v:

$$A_v = \frac{g_m T_{int}}{C_L} \tag{3.1}$$

where g_m is the transconductance of the input devices, T_{int} is the integration time, and C_L is the load capacitance at the output nodes. In the second phase, the circuit resets the outputs to some common mode level (usually V_{DD} or ground, dependeing on the input device type) to clear the previous sample in preparation for the next cycle. The settling error associated with these reset devices ε_r is

$$\varepsilon_r = \exp\left(-\frac{T_{rst}}{R_{sw}C_L}\right) \tag{3.2}$$

where T_{rst} is the reset time and R_{sw} is the switch resistance of the reset devices.



(a) Current integrator (b) Integrate-reset timing (c) Integrate-hold-reset timing

Figure 3.4: Current integration operation

Since the amplifier output is not held during the operation, the following stage must quickly sample the voltage before it resets to differential zero. In fact, current integration in receiver equalizers has primarily been applied to the FFE and/or DFE summer circuit, whose output is sliced by a regenerative comparator with a narrow sampling aperture. A 2-stage current integration scheme is proposed in [18], which requires an additional analog "dynamic latch" circuit between the stages to provide a stable input for the second integrator.

With 4 stages of current integration in this work's datapath, 3 dynamic latches would need to be inserted in the signal chain, resulting in higher power and additional circuit non-idealities. To avoid this, a current integration operation proposed by Eric Chang introduces a hold phase (Figure 3.4c), where the integrating amplifier retains the amplified output value for some time until the resetting begins. While the current stage is in the hold phase, the next stage integrates and amplifies the held voltage, allowing for an easier cascading of the current integration stages. The integrate-reset operation typically assigns equal durations for the integration and reset phases, which results in the phases being controlled by a single 50% duty cycle clock. With the integrate-hold-reset scheme, however, the clock period must now be divided into 3 different phases. This results in shorter integration and/or reset pulses, which tend to increase clocking power. As this amplifier now "latches" onto and retains the input during the hold phase, it will be henceforth referred to as the integrating latch.

3.3.1 Intersymbol Interference Design Considerations

The integrating latch can generate ISI, which must be accounted for in the overall link analysis. The ISI comes from two independent sources in the integration and reset phases. By nature of integration, the latch low-pass filters the input during the integration window. As discussed in [17], this results in frequency-dependent gain that could introduce 3.9 dB of loss at Nyquist frequency if the integration window is 1 UI wide. One can control the amount of ISI by controlling the integration period, which would also modulate the gain. Nevertheless, the integration non-idealities only translate to ISI for the first stage samplers. Later stages have inputs which are already sampled and in discrete time, so any integration window effects only impact the gain of that stage.

At low enough speeds, the integration window is primarily set by the width of the integration pulse. For higher rates, the effective integration period is further affected by the limited bandwidth at the tail node. During integration, the tail node must drop at least a V_{th} below the input common mode for the input devices to turn on. Then, the tail node must be pulled up back within a threshold of the input common mode for the input devices to turn back off, signifying the end of the integration window. Tail node bandwidth limited designs can increase the delay and transition times of the tail voltage, which disperse and time-shift the ideal boxcar integration window. For frontend samplers that rely on current integration, controlling the effective tail node bandwidth is key to mitigating ISI.

During the reset phase, the differential output voltage exponentially approaches zero. Any residual output left at the end of the reset phase is defined as the reset error and is simply the settling error from (3.2). Larger reset devices will reduce the reset error but introduce additional parasitic device capacitance to the output, thereby decreasing the integration gain. Unlike the ISI from integration, the reset error affects all stages of current integration, where an Nth postcursor ISI tap is effectively introduced if the latch is operating at $1/N$ of the full data rate. With the MLSE slices operating at 16x interleaving, the reset error would mainly manifest as 16th postcursor tap ISI, which is not readily handled by the equalizer. Thus, the reset error was kept to a minimum and its effect was negligible on the overall link performance.

The ISI effects can be observed by running an impulse sensitivity function (ISF) characterization, where the integrating latch input is driven by an impulse signal and the held output amplitude is measured. This process is repeated for different arrival times to generate the ISF. The equivalent impulse response can then be obtained by reversing the time axis. An example impulse response waveform is provided in Figure 3.5 for a latch with a total operation time of 16 UI. The main output impulse is at UI 0, with some nonzero width due to the integration width of the latch. An attenuated version of this impulse is shown at UI 16, which corresponds to the reset error of this latch.

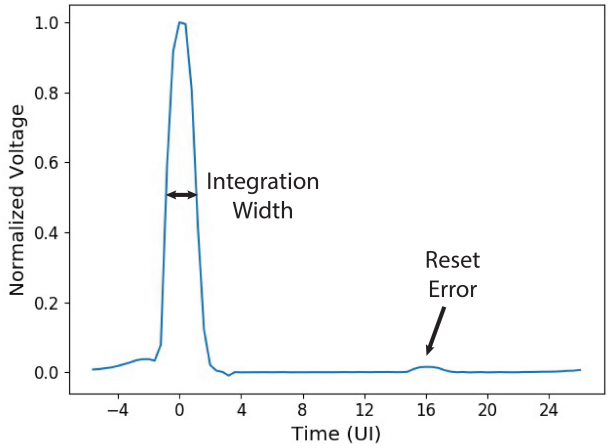


Figure 3.5: Impulse response of integrating amp showing different sources of ISI

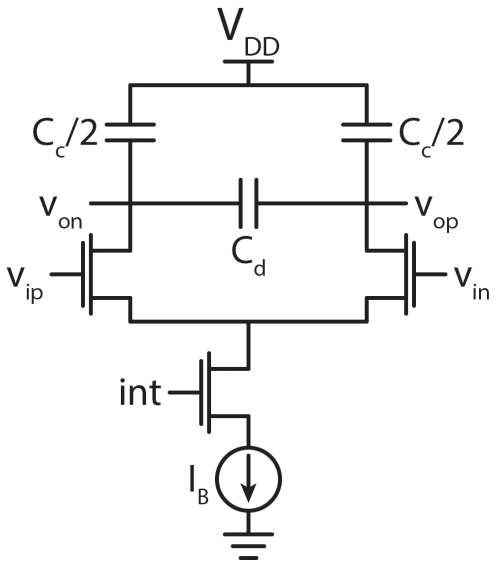


Figure 3.6: Latch model during integration

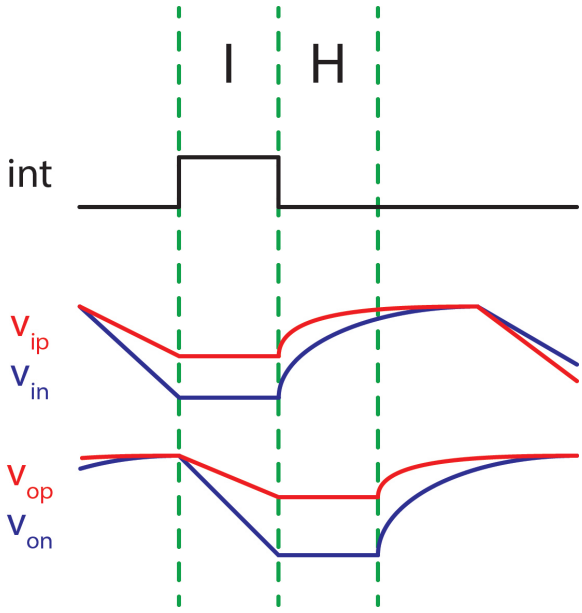


Figure 3.7: Waveforms for current-integrating input

3.3.2 Output Linearity Design Considerations

During the integration phase, the output common mode decreases linearly with integration time. If the common mode drops too low, the input devices fall into triode, degrading the latch gain and linearity as the device output resistance r_o begins to modulate with the input amplitude.

Suppose the parasitic load capacitance at the output nodes are modeled by common mode (C_c) and differential mode (C_d) components, as shown in Figure 3.6. For integration time t_{int}, the drop in common mode voltage Δv_{oc} is

$$\Delta v_{oc} = \frac{I_B t_{int}}{C_c} \tag{3.3}$$

The expected voltage gain A_v if the input devices remain in saturation is

$$A_v = \frac{g_m t_{int}}{\frac{C_c}{2} + 2C_d} = 2\frac{g_m t_{int}}{C_c + 4C_d} \tag{3.4}$$

where g_m is the transconductance of the input devices. The ratio of the common mode drop to voltage gain is set by design and load parameters:

$$\frac{\Delta v_{oc}}{A_v} = \frac{I_B}{2g_m}\frac{C_c + 4C_d}{C_c} \tag{3.5}$$

We define V^* to represent the current efficiency of the transistor with drain current I_d and transconductance g_m:

$$V^* \equiv \frac{2I_d}{g_m} \tag{3.6}$$

Note that V^* is mathematically equivalent to the overdrive voltage V_{ov} for long channel devices. Although this equivalence doesn't apply for short channel devices, the general V_{ov} trade-offs with respect to g_m linearity, f_T, etc. can be applied to V^*. Applying the V^* definition to (3.5) yields

$$\frac{\Delta v_{oc}}{A_v} = V^*\left(\frac{1}{2}\frac{C_c + 4C_d}{C_c}\right) = k_c V^* \tag{3.7}$$

where $k_c = \frac{1}{2}\frac{C_c+4C_d}{C_c}$ represents some ratio of common mode to differential capacitance. At its minimum, $k_c = 0.5$ when $C_d = 0$.

Applying square law intuition, suppose to the first order that the input device will enter the triode region when $V_{gd} > V_{th}$. For an input differential voltage v_{id} and common mode voltage v_{ic}, the output node must not fall more than V_{th} under the input node:

$$V_{DD} - \Delta v_{oc} - A_v\frac{v_{id}}{2} \geq v_{ic} + \frac{v_{id}}{2} - V_{th} \tag{3.8}$$

Based on the relationship between gain and common mode drop as derived in (3.7), (3.8) can be rewritten as a maximum A_v constraint between for some V^*:

$$\begin{aligned} V_{DD} - k_c A_v V^* - A_v\frac{v_{id}}{2} &\geq v_{ic} + \frac{v_{id}}{2} - V_{th} \\ A_v\left(k_c V^* + \frac{v_{id}}{2}\right) &\leq V_{DD} - v_{ic} - \frac{v_{id}}{2} + V_{th} \\ A_v &\leq \frac{V_{DD} - v_{ic} - \frac{v_{id}}{2} + V_{th}}{k_c V^* + \frac{v_{id}}{2}} \end{aligned} \tag{3.9}$$

The above mainly applies while the current stage is in integration phase and the previous stage is in the hold phase. However, when the current stage is in the hold phase, the previous stage is resetting its outputs, which causes the current stage's inputs to rise up to V_{DD}. As the devices must still remain in saturation during the hold phase, a stricter condition is derived:

$$\begin{aligned} V_{DD} - \Delta v_{oc} - A_v \frac{v_{id}}{2} &\geq V_{DD} - V_{th} \\ V_{DD} - k_c A_v V^* - A_v \frac{v_{id}}{2} &\geq V_{DD} - V_{th} \\ A_v \left(k_c V^* + \frac{v_{id}}{2} \right) &\leq V_{th} \\ A_v &\leq \frac{V_{th}}{k_c V^* + \frac{v_{id}}{2}} \end{aligned} \tag{3.10}$$

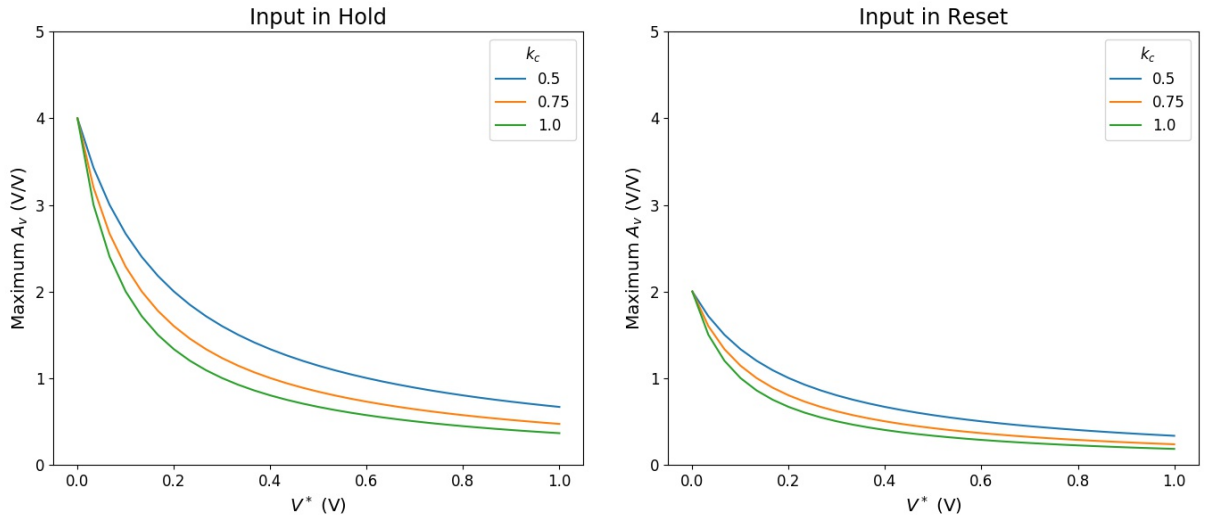


Figure 3.8: Maximum voltage gain (A_v) vs. current efficiency (V^*) tradeoffs for stable input (left) and input resetting to V_{DD} (right); assuming $V_{DD} = 1$ V, $v_{ic} = 0.7$ V, $V_{th} = 0.2$ V, maximum $v_{id} = 0.2$ V

With example numbers described in Figure 3.8, the maximum voltage gain for a given V^* can be ∼2x worse when the input is resetting to V_{DD}, a common occurrence for cascaded current integration stages. Furthermore, picking small V^* to enable larger A_v leaves the latch more susceptible to g_m degradation and non-linearity.

There are several methods to mitigate the gain limitations without compromising linearity. Based on equations (3.9) and (3.10), the maximum gain can be increased by using higher threshold flavors and reducing the differential capacitance (relative to the common mode capacitance). Increasing the space between the differential output wires decreases any

differential parasitic capacitance, but it comes with diminishing returns as the capacitance is inversely proportional to the spacing. Alternatively, adding a shield wire between the differential wires can greatly suppress C_d at the cost of higher total load capacitance. Common mode boosting/restoration circuits can also be added to avoid this issue altogether [15, 18]. Such techniques must be clocked for integrate-hold-reset latches and thus increase clocking power. For this work, the gain limitation effects were reduced by using higher threshold input devices and shielding between the output wires.

3.3.3 Integrating Latch Design Flow

Recall from Figure 3.3 that top-level design iterations require individual unit cell design flows. As most unit cells are based on integration latches, the following methodology will apply to most integrating latches in the signal chain. The one exception is the T&H, which is covered in Section 3.4.

The general integrating latch design flow is shown in Figure 3.9. The design iterations are primarily on meeting linearity, gain, and reset target specifications. Note that because these latches are not used as frontend samplers, the integration-related ISI is irrelvant. First, we design the latch in a way that the linearity is limited not by the output resistance r_o of the input devices, but rather their transconductance g_m. As mentioned in the earlier section, this can be done by increasing the threshold of the input devices and minimizing any differential capacitance. This allows the input linear range constraint to be independent of gain and reset error. The input linear range is primarily set by the bias current density I_{bias}/W_n, which determines V^* (or for long-channel devices, V_{ov}). An initial guess for the bias current density is computed based on transistor I-V characteristics, then any simulation-based design iterations are executed to ensure that the linearity specification is met. Next, a separate design loop is carried out for the gain and reset error specifications. Hand calculations can be made using (3.1) and (3.2), where C_L is modified to include self-loading:

$$C_L = C_{L,ext} + C_{dp}W_p + C_{dn}W_n \tag{3.11}$$

$C_{L,ext}$ represents the external load capacitance, and C_{dp} and C_{dn} represent unit drain capacitances for PMOS and NMOS, respectively. To account for any modeling errors due to circuit non-idealities, the design is iterated based on simulation results. As the reset error and gain are correlated due to their shared dependence on the load capacitance, they are iterated together. Once the loop is converged, the integrating latch design flow is complete.

3.4 8-Way Interleaved T&H

The T&H's are driven by multiple phases of 20 GHz clocks, which correspond to a clock period of 8 UI. Each T&H thus samples the input every 8 UI, with 1 UI of integration, 3 UI of hold, and 4 UI of reset (Figure 3.10b). The integration period must be approximately 1 UI to sample the input continuous-time signal without incurring too much integration ISI.

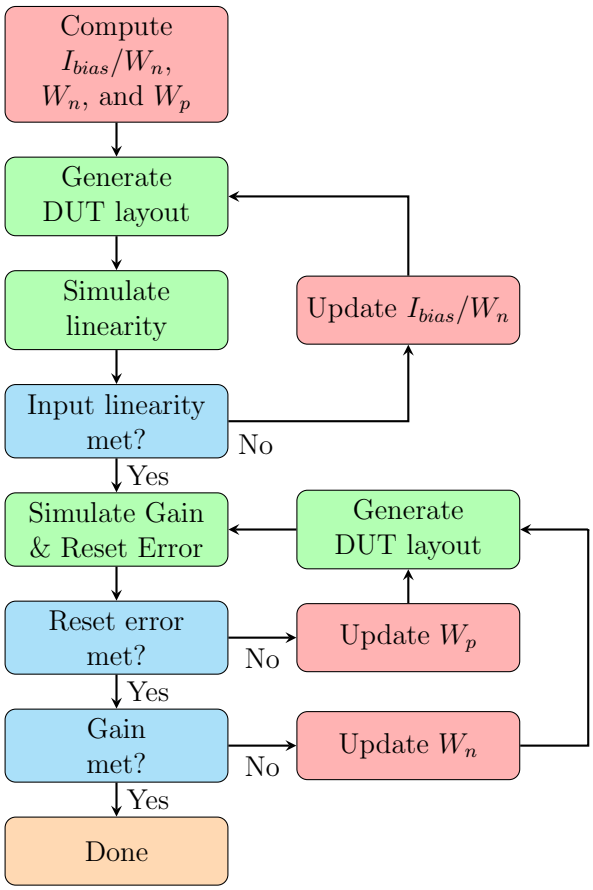


Figure 3.9: Current integrating latch design flow

The 4 UI reset period is chosen to accommodate $< 5\%$ reset error without aggressively sizing the reset devices, and its 50% duty cycle pulse simplifies the reset network to single PMOS devices. The 1 UI integration "pulse" is generated by AND'ing two phases of the 8-phase clocks, as shown in Figure 3.10a. As mentioned earlier, the tail node bandwidth is critical in the design of this frontend sampler. The tail bias current source is removed to increase the pull-down speed on the tail node, while a NMOS pull-up switch is added to the tail node to increase the pull-up speed.

Because the integration window is only 6.25 ps (1 UI), the highest achievable gain while satisfying the maximum capacitive loading allowed by the T-coil was around 0.9 V/V. Although the T&H will likely attenuate the input signal, it was found that passive sampler alternatives would face similar bandwidth constraints that would result in 2-3x lower gain. With such a short integration time, the common mode drop in the output was also small enough that the T&H wouldn't face output linearity issues mentioned in Section 3.3.2. AC-coupling high-pass filters were added after the T&H outputs to lower the high common mode

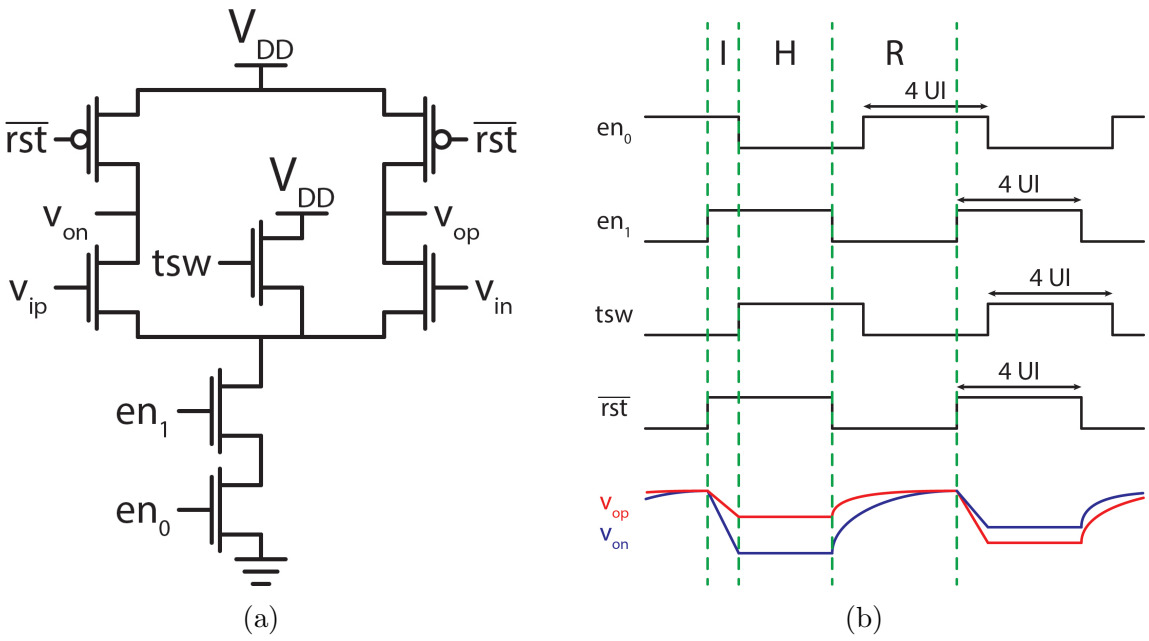


Figure 3.10: T&H topology (a) and timing (b)

levels for the following stage, as well as to provide any offset correction (Figure 3.2).

Because the T&H must operate near the speed limits of the process, each design parameter (device sizing and biasing) could critically impact the performance of the circuit, with a many-to-many correspondence from design parameters to performance specs. For example, sizing up the tail pull-down switches would increase gain and linearity but also the integration-induced ISI. Increasing the input common mode would share similar trends, albeit to different degrees. Upsizing the tail pull-up switch would have the opposite effect as the tail pull-down switch upsizing, and increasing the reset devices would mitigate reset-induced ISI while degrading the gain. Furthermore, simple models to map design parameters to specs often deviate greatly from simulation results at these high rates, as any parasitic effects become even more pronounced. Instead of solving this complex optimization through manual design iteration, an automated design optimization flow was developed in BAG (Figure 3.11).

First, the user specifies both the design space and target specs. The design space is defined by allowed sizes of different devices and the range of each bias parameter. For the T&H, the target specs would be to minimize power under certain gain, ISI, and linearity conditions. Then, every design in the user-defined design space would be created through the circuit generator and characterized automatically. All measurement results would be cached into a singular characterization database. To deal with large design spaces which would require an unrealistic number of DUTs to be generated and characterized, the user may set coarse steps for each design parameter to lower the sweep resolution. Then, the

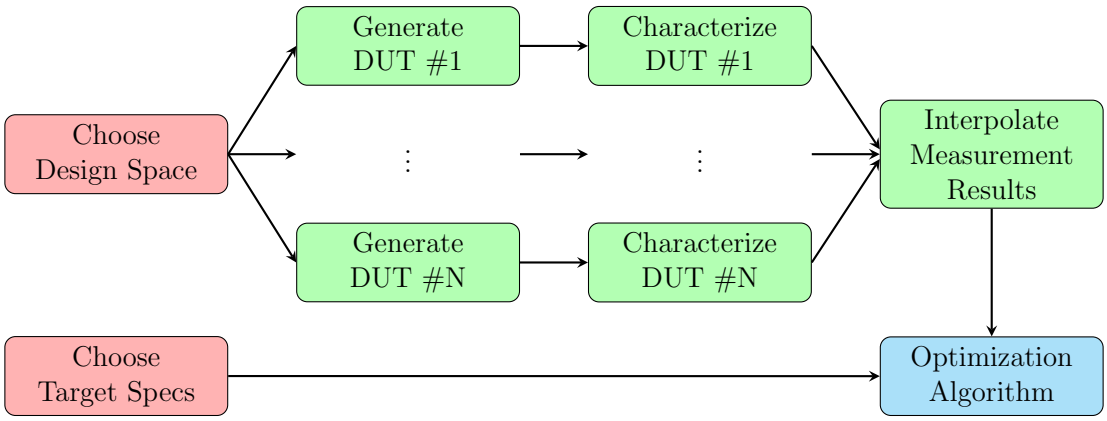


Figure 3.11: BAG optimization flow

database can be queried with interpolation between characterization points. Finally, the user target specs are passed into the optimization engine, which uses the characterization database and computes the final design by calling the SciPy optimize Python package. The target spec to be minimized or maximized may not be convex in design space, and many local optima may exist. To increase the chances of achieving the global optimum, the engine invokes multiple optimization calls with random seeds, and then picks the best design among the covered local optima. Of these steps, the characterization database generation tends to take the longest. However, for a fixed design space, the database only needs to be created once, and can be reused for every subsequent design optimization.

3.5 8:16 Deserializer

The second stage of the frontend deserializer consists of 16 time-interleaved integrating latches, where each pair is driven by one T&H. Because the input to this stage is now at the deserialized rate of 20 GS/s, the tail current source is included for bias control without being bottlenecked by the tail node bandwidth (Figure 3.12a). The tail pull-up switch is also removed for similar reasons. The 8:16 deserializer is clocked by 10 GHz clocks (period of 16 UI), with each unit cell sampling the input every 16 UI over a 4 UI integrate – 4 UI hold – 8 UI reset sequence (Figure 3.12b). Although the T&H hold phase is only 3 UI wide, having the 8:16 deserializer's phases all be multiples of 4 UI simplifies to clock distribution. Since each unit cell only needs quadrature phases, the 8:16 deserializer can be grouped into 4 commonly clocked "quadrants", each containing 4 unit cells. These unit cells integrate during 3 UI of the T&H's hold phase and the first UI of its reset phase. This reduces the average input amplitude seen by the unit cell over its integration window. Phase alignment between the 20 GHz and 10 GHz clocks is critical to ensure that the 8:16 deserializer is integrating while the T&H's are in the hold phase.

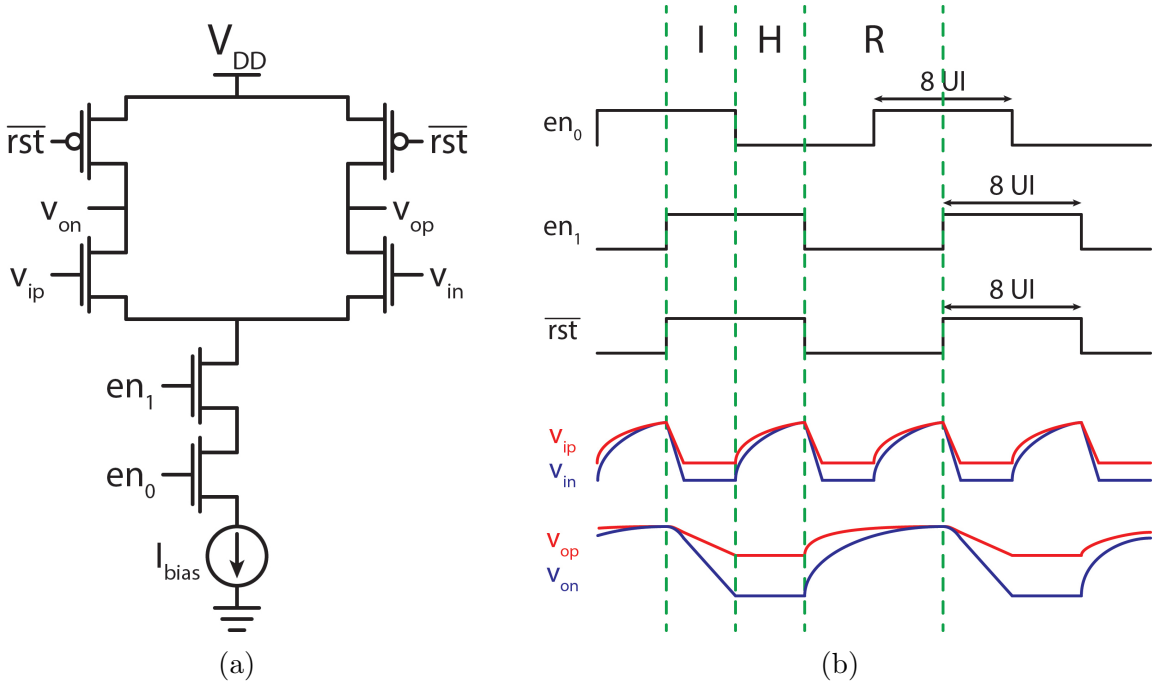


Figure 3.12: Integrating 8:16 deserializer unit cell topology (a) and timing (b)

The 8:16 deserializer unit cell samples every other output of the T&H. If the latch samples even data from the T&H, any odd data that makes it to the latch output would translate to additional ISI around the 8th postcursor. Even if the T&H's odd data capacitively coupled onto the latch output, it would be cleared during the latch's reset phase. With 8 UI of reset time, this effect is usually negligible.

3.6 Integrating Latched Summer

As depicted in Figure 3.2, the MLSE slice contains 2 parallel integrating paths. The first path consists of the same integrating latch topology as the 8:16 deserializer (Figure 3.12a) and serves delay-match with the second path while providing more gain for the $\pm\alpha$-threshold comparator decisions. The second stage includes an integrating latched summer, which computes the $v_r[n] - v_r[n-1]$ term (2.22) needed for the window length 2 MLSE equalizer.

The latched summer is made of two integrating latches, each driven with a pair of differential inputs (Figure 3.13a). The two inputs v_{i0} and v_{i1} arrive at the summer through very different paths, as seen in Figure 3.14. Despite how symmetric the floorplanning may be, the two signal paths will have different net gain due to process variation. Furthermore, the latched summer waveforms show that due to the 1 UI shift between v_{i1} and v_{i0}, v_{i0} has a smaller effective magnitude than v_{i1} when both inputs are being amplified in the summer, which introduces more gain mismatch. Let A_{v0} be the total gain from $v_{in,rx}$ (the input of the

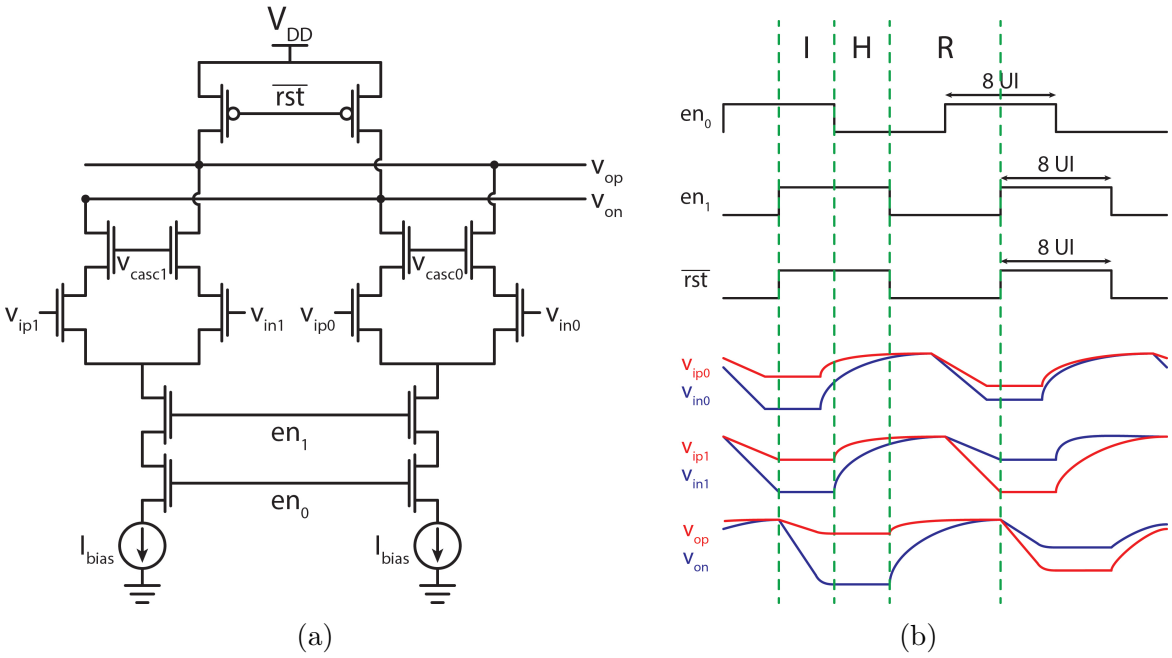


Figure 3.13: Integrating latched summer topology (a) and timing (b)

receiver chain) to summer input v_{i0}, and let A_{v1} be the total gain from $v_{in,rx}$ to v_{i1}. To deal with gain mismatch, if the latch driving v_{i0} has transconductance g_m, then the latch driving v_{i1} must have some gain-calibrated transconductance w such that the net gain is equal on both branches, i.e.

$$A_{v0}g_m = A_{v1}w \Rightarrow w = g_m\frac{A_{v0}}{A_{v1}} \tag{3.12}$$

As a result, the latches in the summer have an additional cascode device, which is used to precisely equalize gain between the two paths. Cascode gate biasing has been shown to be an effective way to control gain without compromising linearity [18].

3.7 Backend Sampler

After the integrating gain and summer stages in the MLSE slice, the signals must be sliced and resolved to CMOS levels for further signal processing in the digital domain. While the summer output ideally requires no scalar subtraction or voltage threshold, the other paths require non-zero thresholds for proper operation. The slicer paths corresponding to "data1p" and "data1n" in Figure 3.2 need $\pm\alpha$ thresholds to implement the window length 2 algorithm (refer to Figure 2.14), and the error slicer path "data1e" must compare against data levels for equalization adaptation. All slicers are also offset-calibrated to improve the link margin. The offset correction is jointly applied with any decision thresholds through AC-coupling

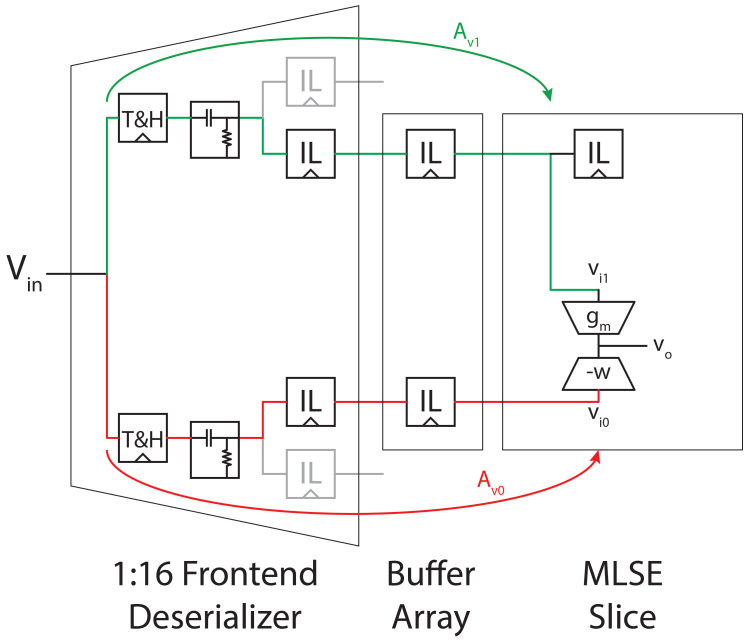


Figure 3.14: Summer input path tracing between v_{i1} (green) and v_{i0} (red)

high pass filters, which can be biased as such to provide differential DC voltages at the input of the comparators [18].

A StrongArm (SA)-based flip flop is used as the comparator (Figure 3.15) for its high sensitivity. On the rising edge of the clock, StrongArm latch senses the input and regenerates rail-to-rail outputs. However, the StrongArm exhibits a return-to-zero (RZ) behavior as its outputs reset to V_{DD} while clock is low. A set-reset (SR) latch is thus used to capture and hold the StrongArm latch decision for the entire clock period. Inverters are added between the SA and SR latches to mitigate any hysteresis from the SR latch loading. PMOS devices are also added in the SA latch to pre-charge all intermediate nodes to V_{DD}, further reducing any hysteresis (Figure 3.16). The symmetric SR latch proposed in [23] is used for its lower latency.

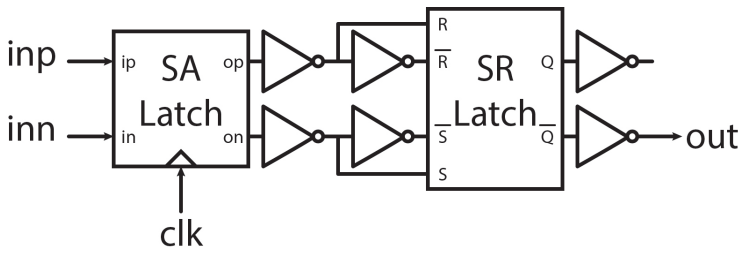


Figure 3.15: StrongArm flop

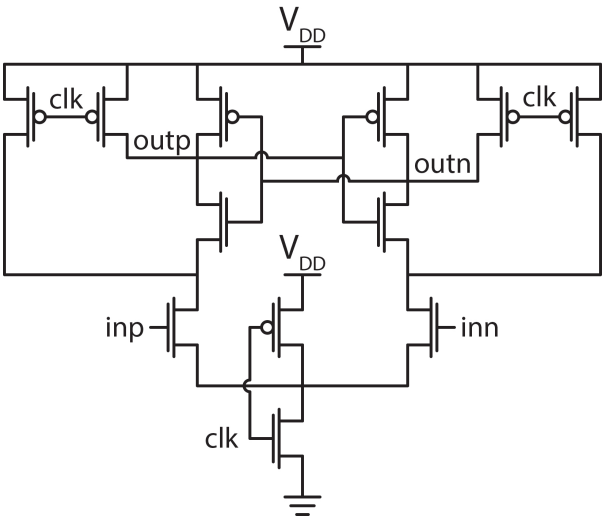


Figure 3.16: StrongArm latch

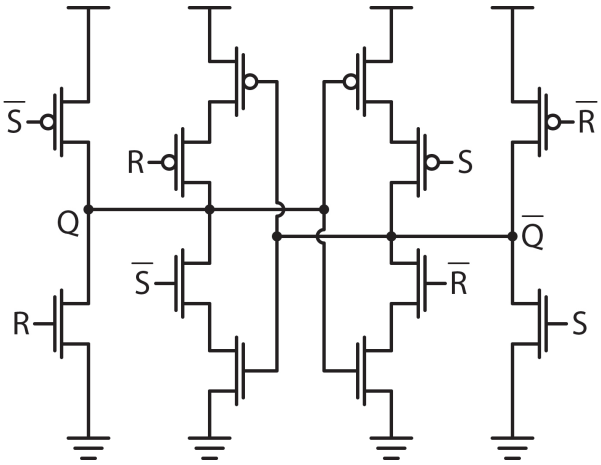


Figure 3.17: Symmetric SR latch

3.8 Backend Deserializer

The time-interleaved 10 GS/s comparator outputs are aligned to 10 GHz clock phases from 0° to 337.5°, in steps of 22.5°. They must be retimed and deserialized to a 1.25 GS/s data stream that can be sampled off of a single-phase digital clock. To maximize setup and hold margins, the retiming is implemented in two stages: coarse and fine. The coarse retimer aligns 4 quadrature (90°-spaced) outputs to the same clock edge (Figure 3.18). First, using a D-type latch, the 0° and 90° outputs are each delayed half a clock period and aligned to 180° and 270° phases, respectively. Then, all quadrature signals are latched by a single 10 GHz phase that aligns them to 270°. The coarse retiming waveforms are shown for a single quadrature group in Figure 3.19.

After the coarse retiming, all data is aligned to clock phases between 270° and 337.5°, with an overlap of 13 UI or 81.25 ps. This allows for the fine retimer to simply sample all coarse outputs with a single phase clock, with a maximum timing margin of $\sim \pm 40$ ps (Figure 3.20). Instead of latching all coarse outputs with a 10 GHz clock, the fine retimer is implemented in the first stage of the deserializer, which samples the input with 5 GHz clocks, thereby reducing clocking power.

The backend 1:8 deserializer is implemented using a 3-stage tree structure, where each stage provides a deserialization ratio of 2 and operates at half of its input data rate (Figure 3.21). Compared to the shift register topology, the tree structure consumes less power as it avoids the use of the baud-rate clock frequency, but must meet tighter timing constraints. The required 5 GHz, 2.5 GHz, and 1.25 GHz divided clocks are generated through a series of C^{2}MOS divide-by-2 stages (Figure 3.22). With a large (64) number of deserializers,

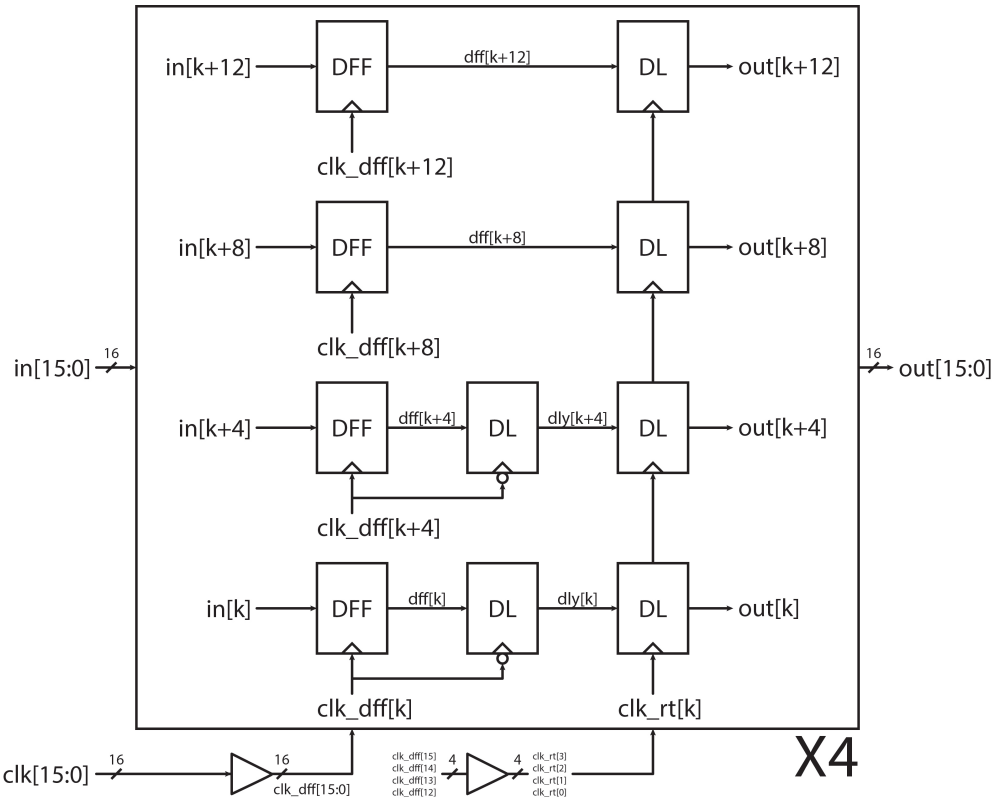


Figure 3.18: Coarse retimer

distribution of the divided clocks adds significant clock skew, which must be accounted for when verifying timing arcs. To avoid hold time violations, the clock divider chain features optional inverters to introduce a 180° phase shift to a divided clock, which can increase the timing margin. Each deserializer stage is composed of 1:2 demultiplexers, which latch the even and odd input data at clk and $\overline{\text{clk}}$ phases, respectively (Figure 3.23). Then, the odd path is delayed by a half clock period using a second D-latch such that the outputs are aligned to the same clock phase. Most demux stages are implemented using latches to reduce power consumption, but the final stage is composed of edge-triggered FFs to relax the timing constraints for the synthesized digital backend.

3.9 Simulation Results and Conclusions

Serial links operating at 100+ GBaud/s rates demand a highly parallelized datapath to meet the throughput requirements. Highly interleaved structures often come with routing overhead, which must be carefully managed during the circuit design process. On the other hand, frontend circuits as the T&H must still operate at the full rate, making their design most critical for the link performance. For both challenges, the generator-based flow enables

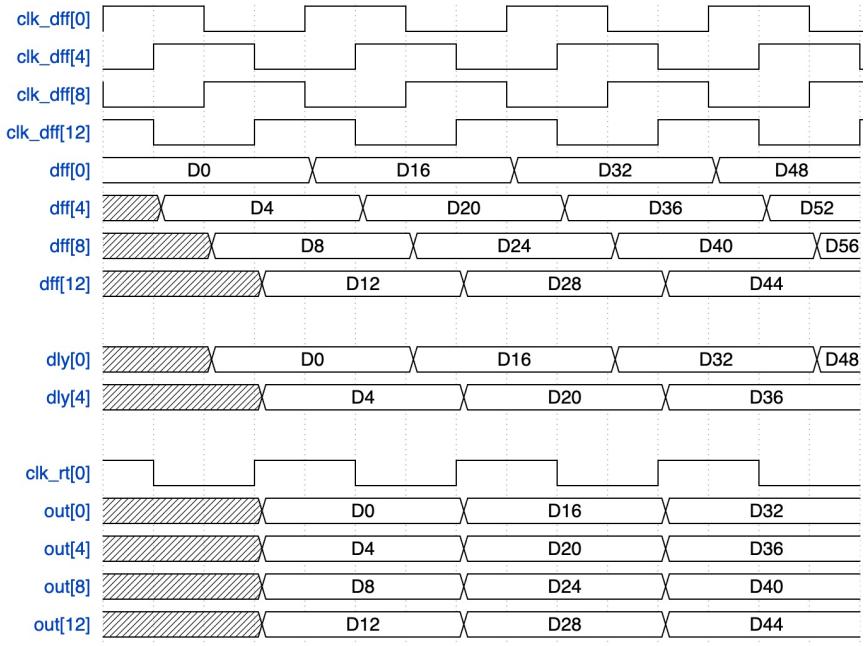


Figure 3.19: Coarse retiming

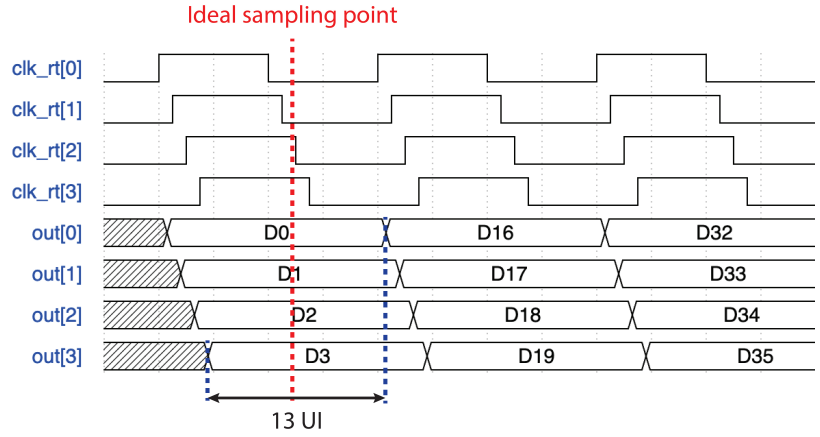


Figure 3.20: Fine retiming

the engineer to quickly re-evaluate circuit design modifications, or automate the design loop altogether through codified optimization strategies.

A 160 Gb/s wireline receiver design is presented in this chapter, with current integration techniques to achieve energy-efficient MLSE equalization. The conventional integrate-reset scheme is extended to a integrate-hold-reset latch scheme to enable cascaded integration stages for analog signal processing. With these design techniques, the receiver datapath is

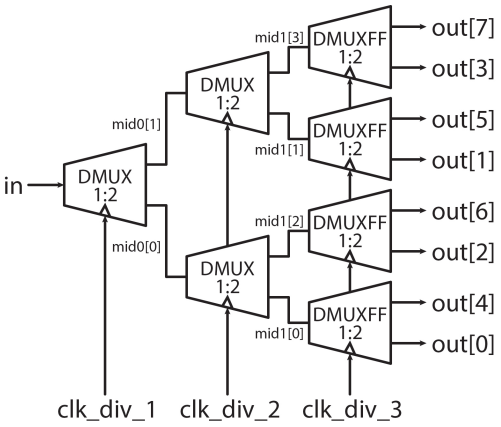


Figure 3.21: Backend 1:8 deserializer

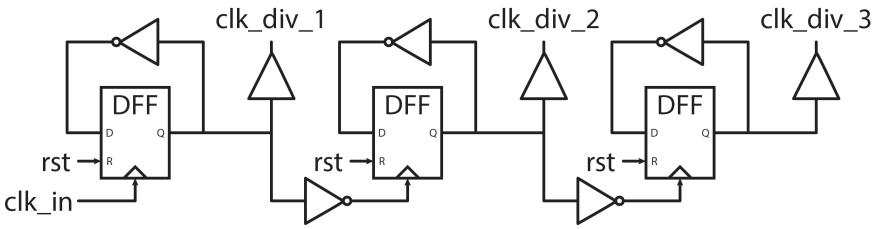


Figure 3.22: Clock divider chain

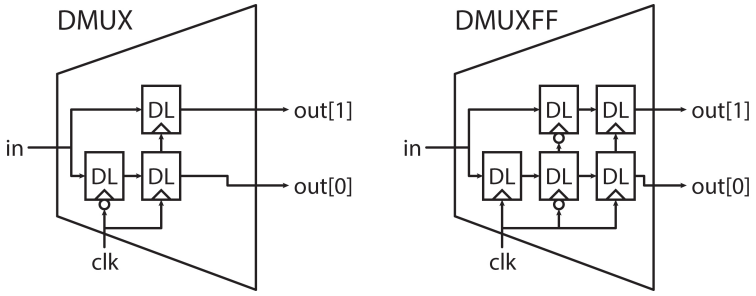


Figure 3.23: 1:2 demultiplexer: latch (left) and flip flop (right)

simulated to consume 115 mW, or an energy efficiency of 0.72 pJ/bit, in a 16 nm FinFET process. The datapath is designed to achieve a total voltage gain of 2.5 V/V is achieved from the T&H to comparator inputs, and the total integrated noise is 2 mV RMS when referred to the input of the T&H.

The generated datapath is shown in Figure 3.24 and spans 240 µm by 150 µm.

Block	Power (mW)
T&H	20
8:16 Deserializer	10
MLSE EQ	20
Comparator + Retimer	40
Backend Deserializer	25
Total	115

Table 3.1: Datapath power (simulated) breakdown

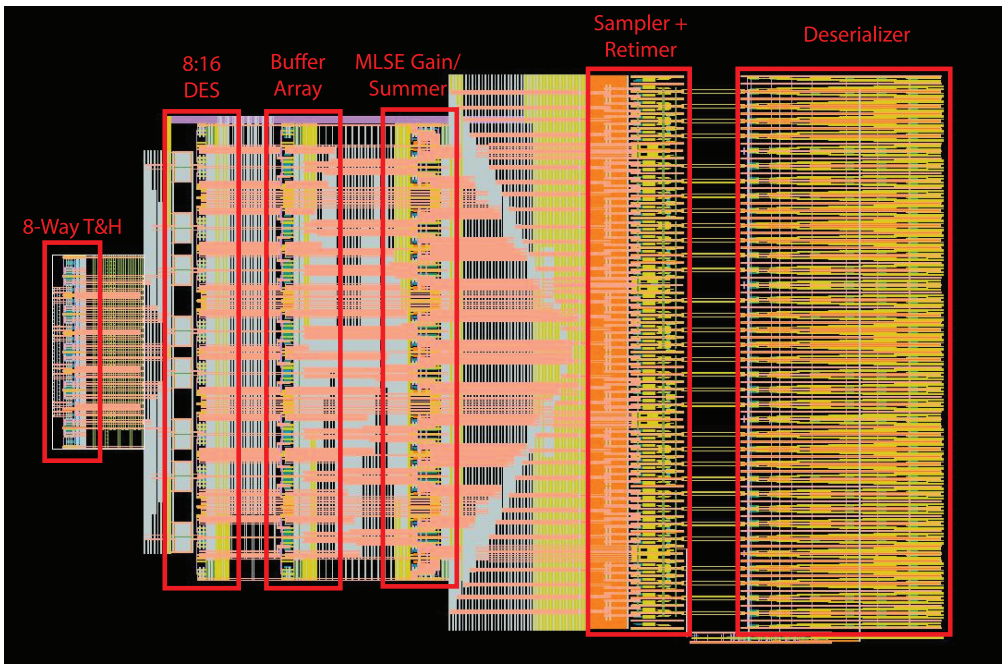


Figure 3.24: Generated MLSE analog datapath layout

Chapter 4

160 Gb/s Receiver Integration

4.1 Receiver Design

4.1.1 Clocking

The receiver datapath requires multi-phase 20 GHz and 10 GHz clocks for operation. The clocking path was designed by Yi-Hsuan Shih and Wahid Rahman. An externally sourced differential 20 GHz clock is buffered and fed into an 8-phase injection-locked ring oscillator, whose implementation is based on the quadrature clock generator described in [24]. Phase alignment is critical for the octature 20 GHz clocks, which are needed by the datapath's T&H; phase errors could worsen the ISI. Thus, digitally controlled delay lines (DCDLs) are added to independently tune each phase of the 20 GHz clocks. To minimize the adverse effect of phase error on the receiver performance, the resolution of the DCDL was designed to be 50 fs ($< 1\%$ of the UI). The tuning range was set to 3.2 ps to account for $\pm 3\sigma$ of delay variation.

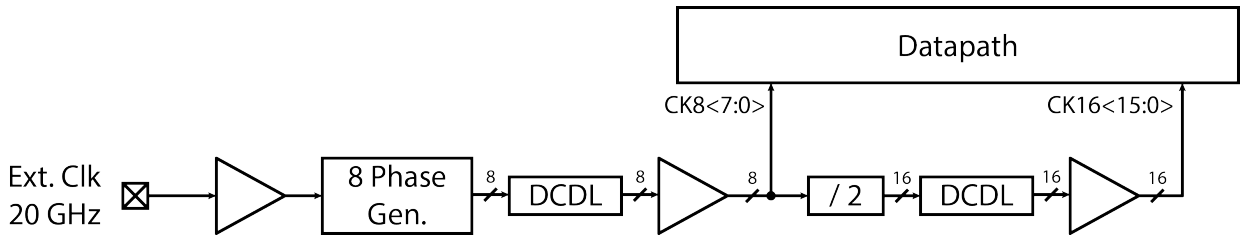


Figure 4.1: Receiver clocking path

The 8 phases of 20 GHz clocks are then fed into 4 parallel clock dividers, each of which takes differential pair of phases and generates quadrature phases at half the frequency (Figure 4.2). Each divider consists of two differential C^2MOS latches, whose circuit is shown in Figure 4.3. The dividers output 16 phases of 10 GHz clocks, which must be phase-aligned to the 20 GHz clocks. Recall from Figure 3.12b that the 8:16 deserializer should be integrating while

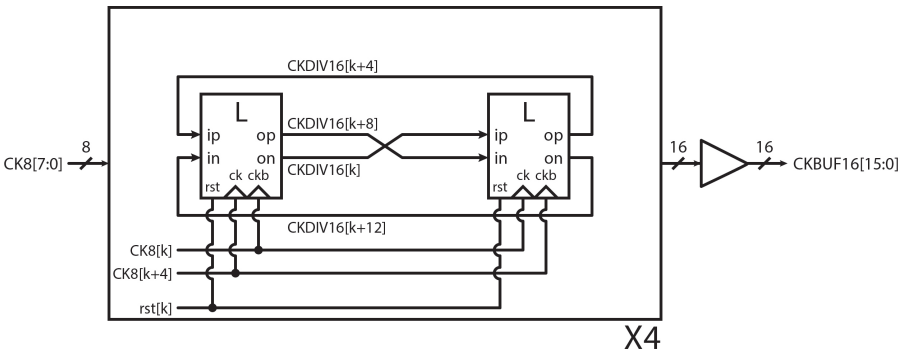


Figure 4.2: 20 GHz to 10 GHz clock dividers

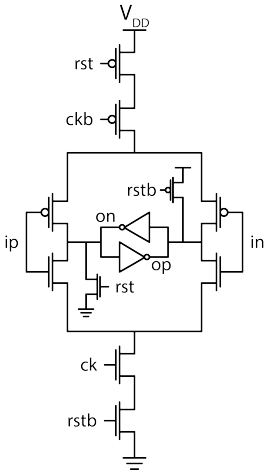


Figure 4.3: Differential C^2MOS latches

the T&H is holding its output. The 10 GHz clock phases can be corrected through a phase rotator, which provides coarse delay tuning, and another DCDL, which provides fine tuning.

4.2 On-Chip Testing Features

One of the challenges of building high-speed links is the lack of direct visibility into the receiver. Attempting to create probing circuitry on any high-speed signal path could detrimentally affect the performance due to the additional capacitive loading. In addition, routing such signals on a package or test board could incur significant loss or degrade signal integrity at frequencies above 1 GHz if not carefully designed. Aside from probing DC bias nodes, a low-speed scan chain is often the primary way of getting any data-dependent debug information off the chip.

On-chip testing capability is especially critical for functionality that cannot be implemente off-chip for runtime reasons. For example, consider the BER measurement. To measure bit error rates of 10^{-12}, more than 10^{12} bits would need to be captured and checked for errors. A 160 Gb/s transceiver requires 6.25 to test 10^{12} bits, plus some additional I/O time to send the error count off-chip. An off-chip implementation for this would require many ($\sim 10^9$) snapshots, and the measurement runtime would be severely limited by the I/O latency. If 10 snapshots can be taken off-chip every second, testing 10^{12} bits would take 2-3 years. This disparity in runtime only increases as more bits are needed to accurately measure a BER of 10^{-12}.

The digital backend logic is shown in Figure 4.4. To account for accidental polarity swaps or endianness inconsistencies between the analog and digital blocks, the deserialized MLSE outputs first go through a bits formatter block, where each signal can optionally be

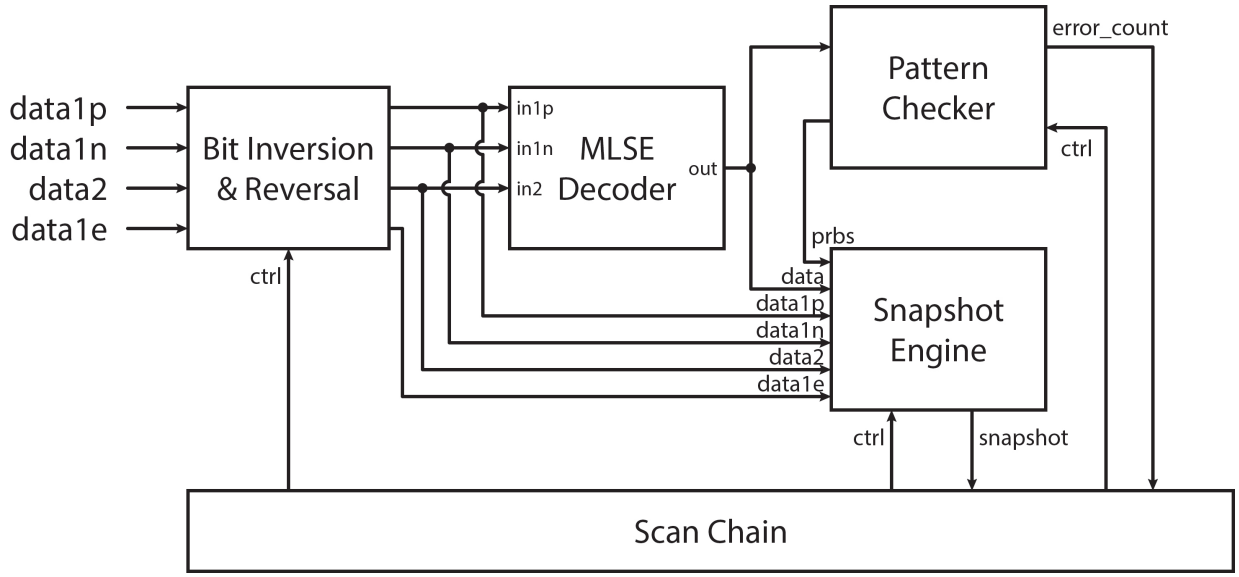


Figure 4.4: Digital backend block diagram

inverted or bit-reversed based on chicken bits. This block is followed by the MLSE decoding logic, whose output can then be verified in the pattern checker. To maximize visibility, all intermediate data signals are collected by the snapshot engine to enable off-chip transmission.

4.2.1 Snapshot Engine

The snapshot engine is controlled by a Moore finite state machine (FSM) shown in Figure 4.5. Starting in the idle state, the FSM waits for the enable control signal to be asserted. Once the enable signal goes high, the module takes each input data and stores it into a bank of flip flops. This occurs for D consecutive cycles, where D is the snapshot depth. With a digital backend word width of 128 bits and snapshot depth of $D = 8$, a total of 1024 consecutive bits are saved per data signal, or 6144 total bits from the PRBS, decoded data, and 4 slicer outputs. Then, the FSM reaches the done state and raises valid high to signify that the data snapshot can be read. The engine will freeze the snapshot until enable is de-asserted and then re-asserted, at which point the engine will record another snapshot. An event-triggered implementation operation allows for safe data transfer off-chip while avoiding metastability and data loss issues commonly found in asynchronous clock domain crossing.

4.2.2 Pattern Checker

The pattern checker computes the number of errors in the data sequence by comparing it against the reference PRBS test pattern (Figure 4.6). Three different PRBS run lengths (7, 15, and 31) are implemented in this work, although any other set of PRBS sequences can

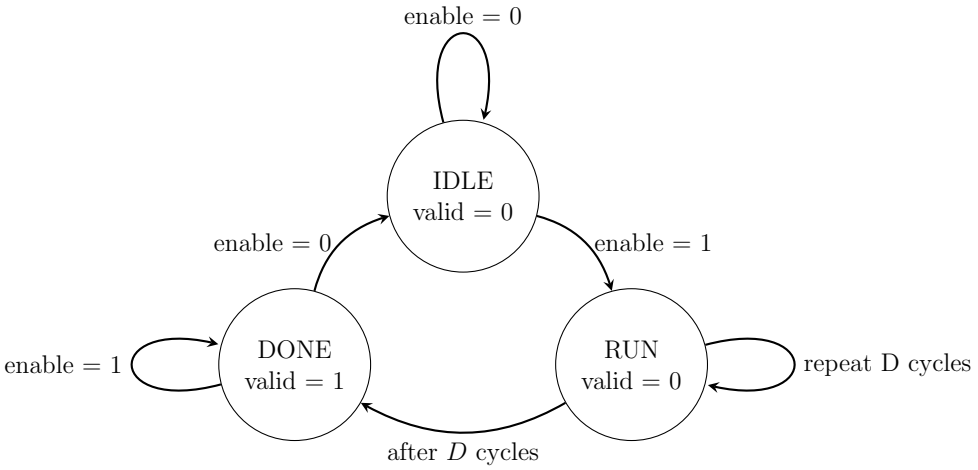


Figure 4.5: Snapshot engine FSM

easily be configured by the Chisel generator. First, the PRBS generators are synchronized to the input data stream by directly seeding the LFSR state with the input. Then, these modules are switched to the free-running state. The selected PRBS test pattern is then compared against the input data stream via an XOR, and the number of errors are accumulated for a user-specified number of data words. A single cycle delay is introduced on the input data comparison path to account for the PRBS generator's inherent 1-cycle latency. The accumulator width w_{err} is determined by the minimum measurable BER:

$$BER_{min} = \frac{1}{2^{w_{err}}} \Rightarrow w_{err} = \lceil -\log_2 BER_{min} \rceil \tag{4.1}$$

After the measurement is complete, the error count is frozen and can be shifted off-chip via the scan chain. The counters are then reset for the next measurement.

For a parallelized digital backend of 128 bits, the PRBS generator must output the next 128 bits of the sequence every cycle. This is done by representing a one-step PRBS LFSR state update as a transition matrix, and then raising it to the 128th power to obtain the 128-step transition matrix [25].

4.3 Off-Chip Testing Software

Testing functionality with lower bandwidth requirements are written in a Python-based testing framework for ease of implementation. These include feedback loops that track DC or very-low frequency signals such that the I/O latency doesn't cause loop instability. As these loops require data and error samples, the snapshot engine is heavily used for off-chip testing software.

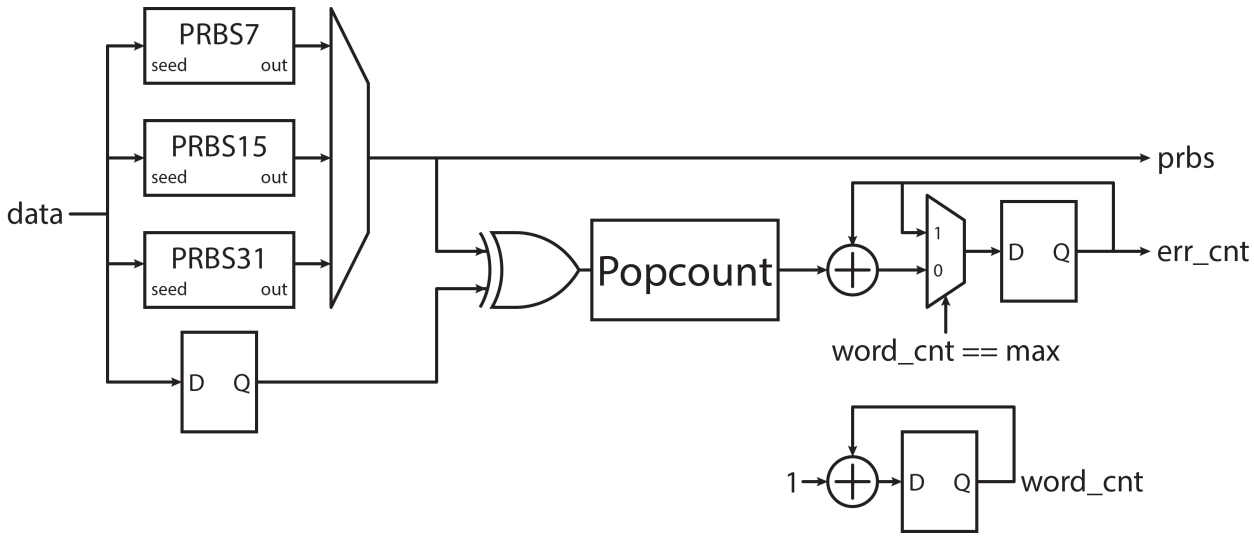


Figure 4.6: Pattern checker block diagram

4.3.1 Equalizer Adaptation

The MLSE has 3 core coefficients that must be adapted: vref1p, vref1n, and w2. Without any circuit mismatch, the comparator thresholds vref1p and vref1n should be set to $+\alpha$ and $-\alpha$, respectively, and the summer gain setting w2 should be adapted to a voltage gain of 1. However, as mentioned in Section 3.6, the analog frontend circuits will have variation which result in both gain and offset errors. These errors are not fully correlated across the interleaved ways. Adaptation loops must thus account for such mismatch when converging to the unique tap coefficients per interleaved slice of the MLSE. All loops are implemented using the sign-sign LMS [26] for its simplicity; the comparators in the analog datapath already apply the sign function to error and data signals.

First, the $\pm\alpha$ threshold adaptations are considered. Given the similarities in the topology of the 1-tap MLSE and the 1-tap loop-unrolled DFE, the data level (dLev)-based adaptation algorithm proposed in [27] can be directly reused here. At the input of the MLSE slicers, the data levels should ideally be $\pm 1 \pm \alpha$, depending on the transmitted data pattern. Although α cannot be directly observed from the data pattern, data levels $1 + \alpha$ and $1 - \alpha$ can be observed when the transmitted data is a 11 and 01 pattern, respectively. $\pm\alpha$ levels can then be interpolated from the dLevs:

$$\alpha = \frac{dLev_{11} - dLev_{01}}{2} \tag{4.2}$$

$$-\alpha = \frac{dLev_{01} - dLev_{11}}{2} \tag{4.3}$$

This interpolation is done in the digital domain, and the resulting digital codes are sent to the voltage DACs to generate differential biases for the AC-coupling HPFs. Therefore,

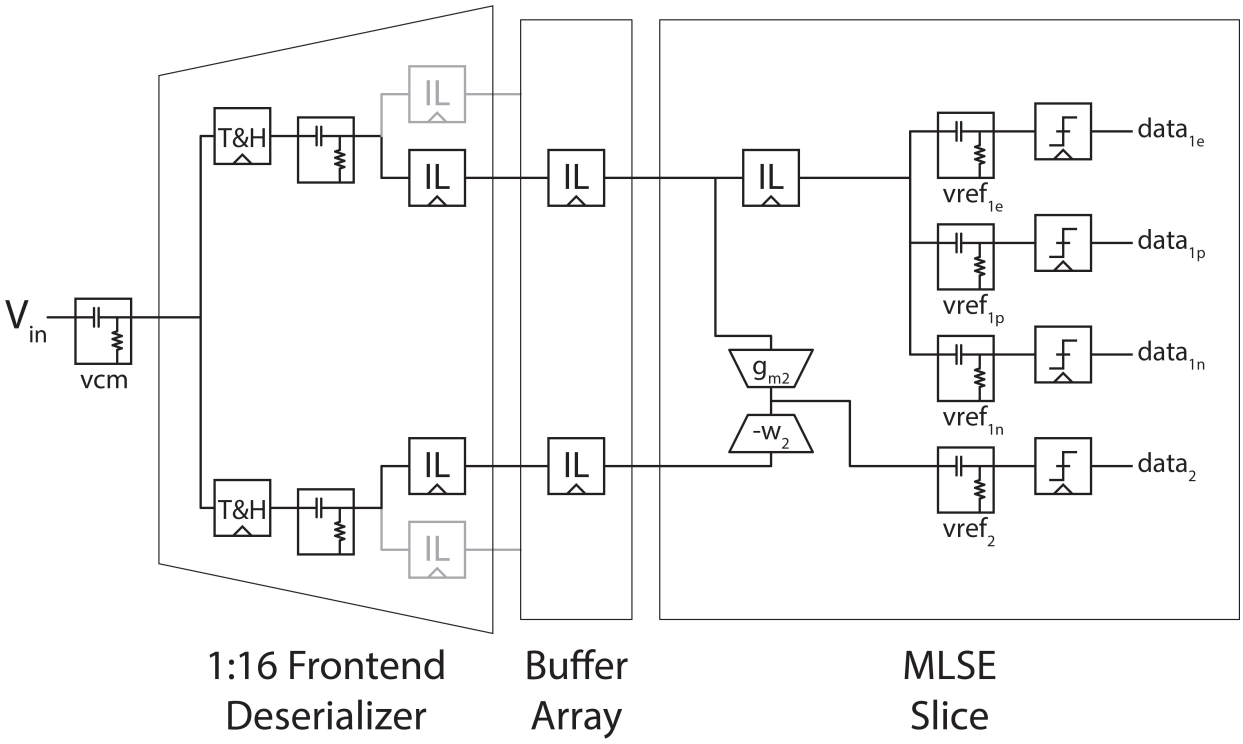


Figure 4.7: MLSE datapath (simplified)

the DAC linearity is critical to minimize errors from interpolation.

The error signal e for the dLev adaptation is defined as

$$e[n] = v_{in}[n] - dLev[n] \tag{4.4}$$

If $e[n] > 0$, then the current value of dLev is too low and must be increased. The inverse is also true, resulting in the following update equation for dLev:

$$dLev[n+1] = dLev[n] + \mu \cdot \text{sgn}(e[n]) \tag{4.5}$$

where μ is the update step size. Note that this adaptation scheme requires a dedicated error slicer to adapt dLev, as the existing slicers must continually be used for the datapath. Thus, an additional slicer with threshold vref1e is added, which serves as the dLev adaptation. This slicer is shared between the adaptation of both data levels to reduce hardware, so the algorithm time-multiplexes between adapting for the 11 and 01 thresholds. Note that by construction, vref1e will also track any offsets input-referred to the error comparator. However, the interpolated $\pm\alpha$ thresholds remove the error offset as the two data levels are subtracted from each other.

The pseudocode is shown in Algorithm 1. As the error samples are unsigned integers with each bit being 0 or 1, the 0s must be mapped to -1s and then summed. This is done

by counting the number of 1s (up) and 0s (dn), and then subtracting the 2 counts to obtain a signed error term. Then, this error term is reduced to an update step using a user-defined function $cntToDelta$.

Algorithm 1 $+/-\alpha$ adaptation using dLev interpolation

Require: $data$ and $data1e$ are bits with width W
Require: $dLev11$ and $dLev01$ are arrays of length N for independent slice adaptation
Require: $adaptDLev11$ is True if currently adapting $dLev11$, False if $dLev01$

$N \leftarrow 16$ ▷ Number of interleaved MLSE slices
$D \leftarrow 8$ ▷ Backend deserialization ratio
$S \leftarrow 8$ ▷ Snapshot depth
$W \leftarrow N \times D \times S$ ▷ Off-chip data width

if $adaptDLev11$ is True **then** ▷ Generate data pattern filter masks
    $mask \leftarrow ptrnFilter(data, 0b11)$
else
    $mask \leftarrow ptrnFilter(data, 0b01)$
end if
$up \leftarrow mask \ \& \ data1e$ ▷ Apply pattern filtering masks
$dn \leftarrow mask \ \& \ \sim data1e$

for $j \leftarrow 0$ *to* $N-1$ **do** ▷ Generate update step per way
    $cnt \leftarrow popCount(up[j :: N]) - popCount(dn[j :: N])$
    $\delta[j] \leftarrow cntToDelta(cnt)$
end for

if $adaptDLev11$ is True **then** ▷ Update $dLevs$
    $dLev11 \leftarrow dLev11 + \delta$
else
    $dLev01 \leftarrow dLev01 + \delta$
end if

$\alpha \leftarrow 0.5 \times (dLev11 - dLev01)$ ▷ Interpolate to get $\pm\alpha$
$vref1p \leftarrow \alpha$
$vref1n \leftarrow -\alpha$
$vref1e \leftarrow adaptDLev11 \ ? \ dLev11 \ : \ dLev01$

The integrated summer gain adaptation algorithm is shown in Algorithm 2. This adaptation follows the dual-loop adaptation scheme [27] where both the tap weight and dLev are adapted simultaneously. Like the $\pm\alpha$ adaptation scheme, note that the inputs of the summer should ideally have data levels of $\pm 1 \pm \alpha$. Because the previous sample is subtracted from

the current sample, consider the ideal summer outputs when the data pattern is a 111 or a 000:

$$v_{sum,111} = (1+\alpha) - (1+\alpha)w_2 \to 0 \tag{4.6}$$

$$v_{sum,000} = (-1-\alpha) - (-1-\alpha)w_2 \to 0 \tag{4.7}$$

With these data patterns, the data level should be differential zero. After pattern filtering, Table 4.1 describes how the coefficients should be updated based on the sliced summer output data2, as well as the current decoded bit data. This approach allows a closed-loop tracking of this slicer's input-referred offset using vref2. Note that a separate error slicer is not required for this path because the summer path captures a differential zero "data level" that can be used for adaptation purposes.

data	data2	w2	vref2
1	1	↑	↑
1	0	↓	↓
0	1	↓	↑
0	0	↑	↓

Table 4.1: Summer gain and offset update

4.3.2 Offset Calibration

Although the proposed adaptation loop cancels any offsets on vref1e and vref2, the vref1p and vref1n paths (corresponding to $\pm\alpha$ thresholds) must be calibrated separately. The offset calibration for these signals is done as a startup procedure, where the external data input to the receiver is set to a DC value. The frontend AC-coupling HPF at the input of the T&H's blocks any DC inputs, thereby providing a differential-zero input to the analog datapath. Then, any offsets are propagated to the comparators in the MLSE slices, where the vref signals can be adjusted to cancel offsets. The offset cancellation is implemented similarly to the vref1e adaptation update, with the exception that no pattern filtering occurs.

The equalizer adaptation and offset calibration loops are simulated with gain and offset mismatches across the interleaved MLSE slices. In the waveforms shown in Figure 4.8, offset calibration first occurs for 100 off-chip cycles of snapshots. Then, the equalizer adaptation loop is enabled, with 250 cycles of dLev11 adaptation followed by dLev01 adaptation. All equalizer coefficients are shown to converge to expected states while accounting for slice-to-slice variation.

4.3.3 Pulse Response Characterization

Without means to directly observe the channel characteristics, an on-chip scope allows for visibility into the effective channel seen by the receiver. In this work, the on-chip scope

Algorithm 2 Summer gain and offset adaptation

Require: $data$ and $data2$ are bits with width W
Require: $ref2$ and $w2$ are arrays of length N for independent slice adaptation

$N \leftarrow 16$ ▷ Number of interleaved MLSE slices
$D \leftarrow 8$ ▷ Backend deserialization ratio
$S \leftarrow 8$ ▷ Snapshot depth
$W \leftarrow N \times D \times S$ ▷ Off-chip data width

$mask111 \leftarrow ptrnFilter(data, 0b111)$ ▷ Generate data pattern filter masks
$mask000 \leftarrow ptrnFilter(data, 0b000)$
$mask \leftarrow mask111 \mid mask000$

$refUp \leftarrow mask \;\&\; data2$ ▷ Apply pattern filtering masks
$refDn \leftarrow mask \;\&\; \sim data2$
$gainUp \leftarrow mask \;\&\; \sim (data2 \oplus data)$ ▷ Apply data-error correlation
$gainDn \leftarrow mask \;\&\; (data2 \oplus data)$

for $j \leftarrow 0$ *to* $N-1$ **do** ▷ Generate update step per way
    $refCnt \leftarrow popCount(refUp[j :: N]) - popCount(refDn[j :: N])$
    $\delta_R[j] \leftarrow cntToDelta(refCnt)$
    $gainCnt \leftarrow popCount(gainUp[j :: N]) - popCount(gainDn[j :: N])$
    $\delta_G[j] \leftarrow cntToDelta(gainCnt)$
end for

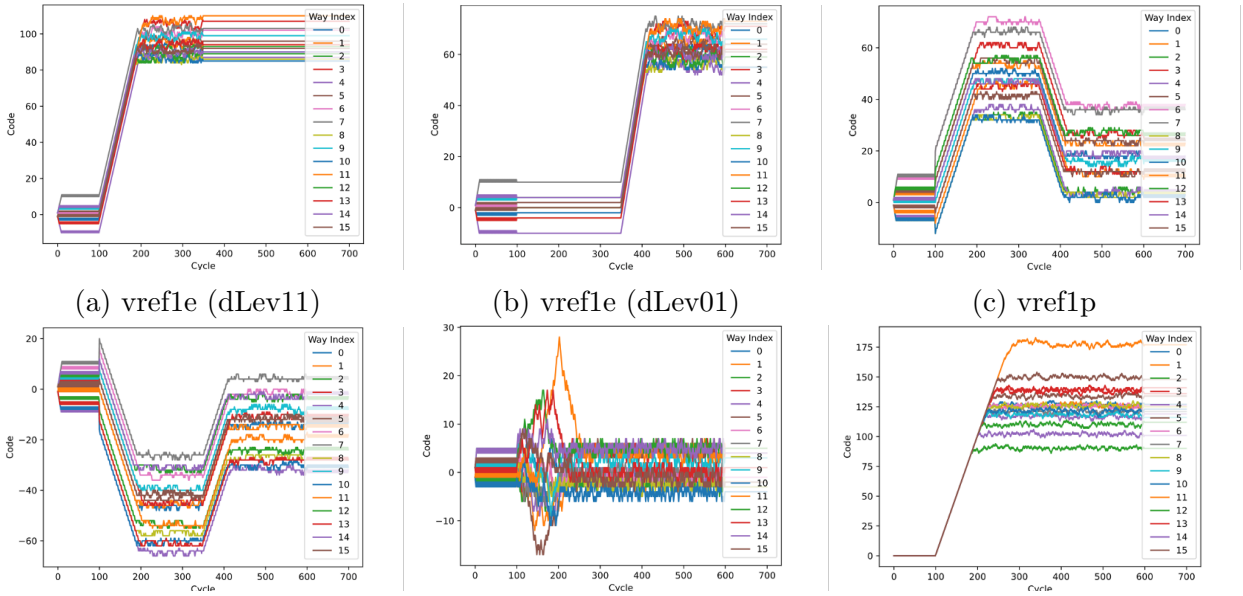
$vref2 \leftarrow vref2 + \delta_R$
$w2 \leftarrow w2 + \delta_G$

functionality reuses the same hardware as the core datapath, specifically through the vref1e error slicer path.

Recall from the earlier section that by pattern filtering for certain data patterns, one can find data levels that give information about the effective channel at the input of the error slicer. In the adaptation scheme, finding the first postcursor α was key, but this strategy can be extended to any other channel tap coefficient.

Suppose that the channel can be modeled as an FIR with M precursor taps, N postcursor taps, and 1 main tap. For some input data sequence x, whose values are ± 1 corresponding to ones and zeros, the channel output y is simply the convolution of the input sequence and the channel FIR:

$$y[n] = (h * x)[n] = \sum_{k=-\infty}^{\infty} h_k x[n-k] = \sum_{k=-M}^{N} h_k x[n-k] \tag{4.8}$$



(a) vref1e (dLev11) (b) vref1e (dLev01) (c) vref1p

(d) vref1n (e) vref2 (f) w2

Figure 4.8: Simulated equalizer adaptation and offset calibration waveforms

For a test sequence of all "zeros" are sent, the corresponding data level output $dLev_{ref}$ is

$$dLev_{ref} = -\sum_{k=-M}^{N} h_k \tag{4.9}$$

Let v_j represent a test data pattern with a single 1 at sample $n-j$, and "zeros" for all other samples. Then, the observed data level $dLev_j$ is

$$dLev_j = \sum_{k=-M}^{N} h_k v_j[n-k] = dLev_{ref} + 2h_j \tag{4.10}$$

Thus, the jth cursor tap coefficient can be computed by interpolating between $dLev_j$ and $dLev_{ref}$:

$$h_j = \frac{dLev_j - dLev_{ref}}{2} \tag{4.11}$$

4.4 Test Setup

The test-chip features an entire 160 Gb/s NRZ transceiver, whose transmitter is used to stimulate the data for receiver testing. Three dies were assembled on the same package.

Two dies were used to form an on-package loopback test configuration, with a channel length of 8.5 mm and loss of 3 dB at Nyquist frequency. The third die on the package enables a secondary, backup method of testing, where an external, lower-rate transmitter or signal generator can provide the data stimuli instead. The transceiver was designed to target the on-package loopback channel of 3 dB loss. The transmitter implemented a 2-tap FFE to cancel the first pre-cursor tap.

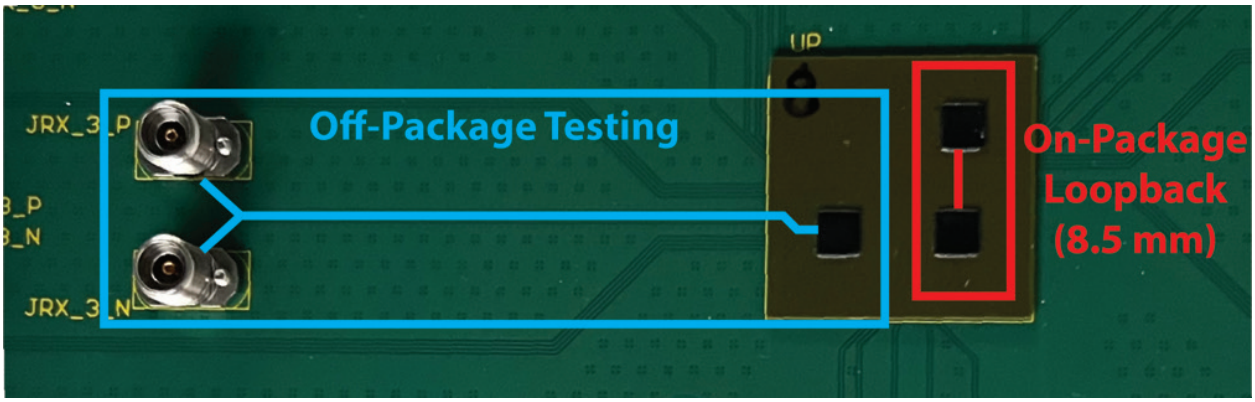


Figure 4.9: Testing options

The 20 GHz reference clock needed for the receiver's injection-locked ring oscillator is generated by an Anritsu MG3692B. The receiver's 1.25 GHz digital clock is monitored by an Infiniium DSO80204B to sanity check the functionality of the clocking path.

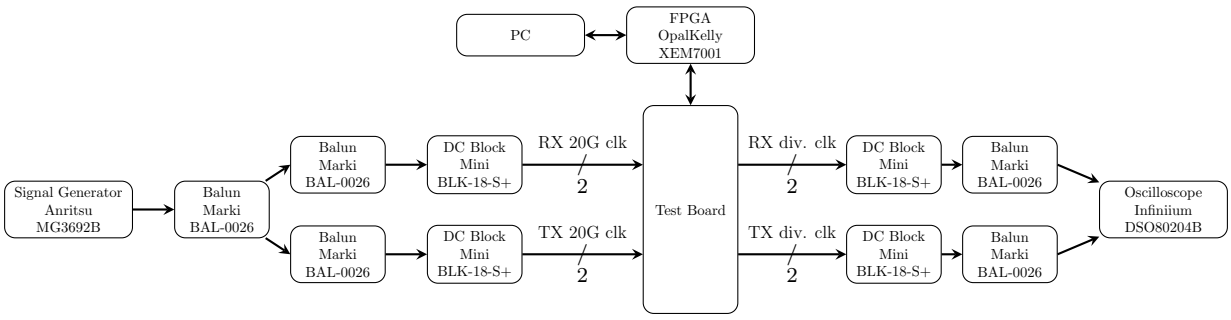


Figure 4.10: Test setup

4.5 Simulation and Measurement Results

The transceiver was taped out in a 16 nm FinFET process. The receiver occupies 0.58 mm^2 and consumes 332.6 mW, which includes the shared digital backend with the transmitter.

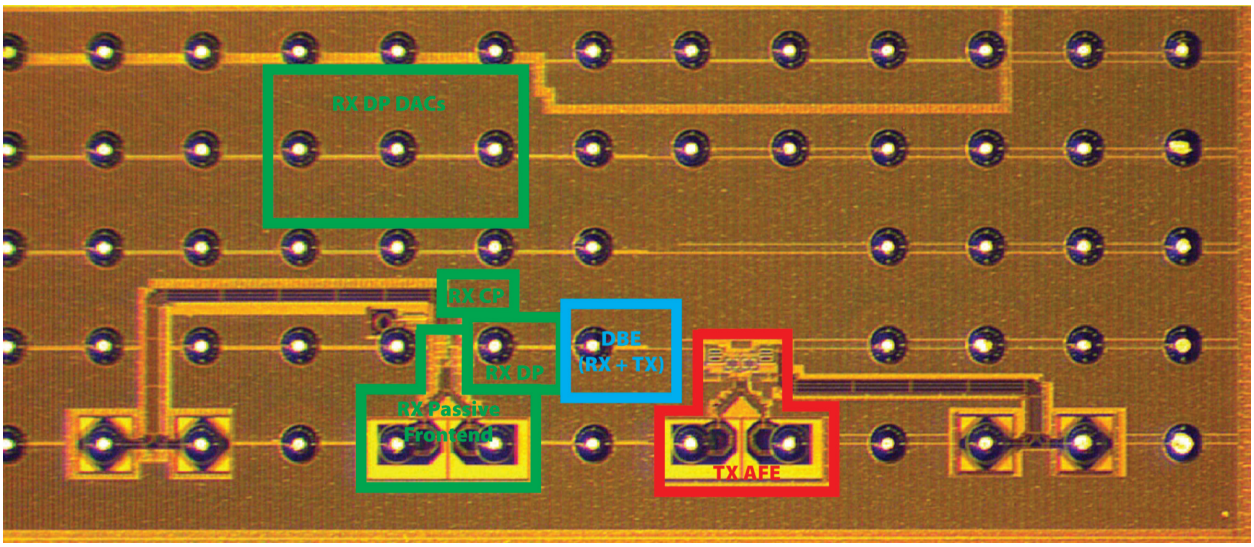


Figure 4.11: Die photo

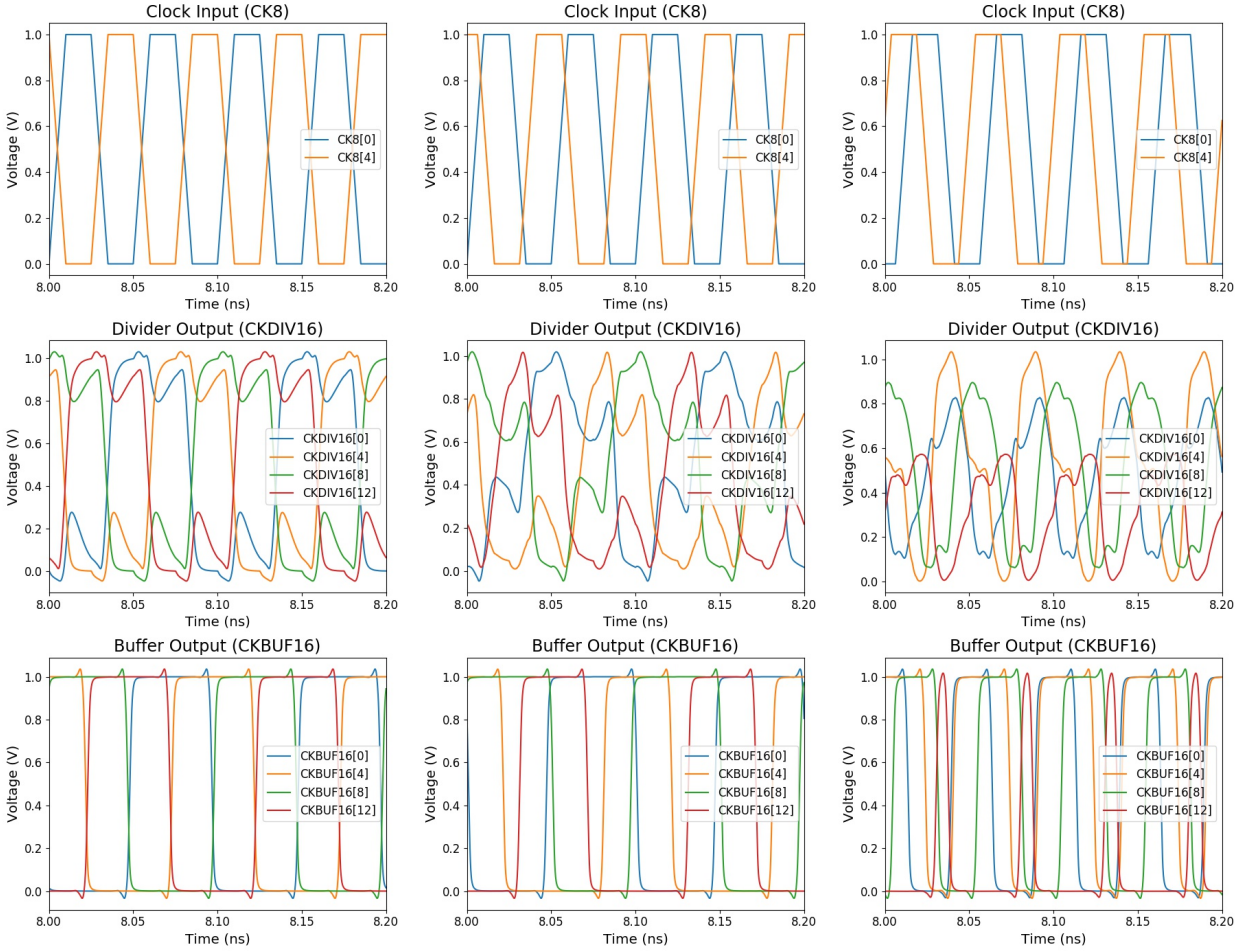
Block	Area (mm^2)	Power (mW)
Datapath (DP)	0.04	115
Clocking Path (CP)	0.01	200
Passive Frontend	0.09	0
Digital Backend (DBE)	0.05	17
DP DACs	0.39	0.6
Total	0.58	332.6

Table 4.2: Receiver area and power (simulated) breakdown

Unfortunately, the output 1.25 GHz digital clock could not be observed during post-silicon testing, which suggests a problem in the clocking path. After further debugging, it is suspected that the differential phases of the 20 GHz clock are picking up enough phase error to cause the 20 GHz to 10 GHz dividers to fail.

As shown in Figure 4.2, the complementary clock phases of the divider's C^2MOS latches are driven separately by differential phases of the 20 GHz clock. Divider simulation results indicate that with 20 GHz clock phases that are 180° out of phase, the divider outputs exhibit some kinks in transition (Figure 4.12a). This is due to the input clock's nonzero rise and fall times, which cause the non-transparent latch to be slightly on during the transition. This effect amplifies as the skew between the differential phases increases, which introduces phase and duty cycle errors after the divider outputs are buffered. With a 45° (or 6.25 ps) skew between the input clocks, the buffered divider outputs exhibit duty cycles of 54% and 40%, as well as a quadrature phase error of 32°(Figure 4.12b). At 90° of skew, the C^2MOS

latches are semi-transparent the entire clock period, causing distorted 20 GHz outputs with varying duty cycle and phase errors (Figure 4.12c). The skew on-chip would have likely been greater than 45° to cause this divider failure and was too large to be correctable by the DCDL (whose tuning range was 3.2 ps, or 23°). Post-layout simulations with the integrated clocking and datapath did not show any signs of such phase mismatch, but this seems to be a deterministic issue as all test chips exhibited the same failure.



(a) 0 skew (b) 45° (6.25 ps) skew (c) 90° (12.5 ps) skew

Figure 4.12: Simulated clock divider waveforms with varying skew between input differential clock phases

Although the performance of the proposed receiver could not be measured due to the aforementioned bug, Table 4.3 shows a performance comparison of this work's simulation results with state-of-the-art transceivers designed for short-reach serial links. This work attempts to achieve the highest data rate and $\sim$ 3x the Nyquist frequency of other works.

Reference	[28]	[29]	[30]	[31]	[32]	This Work[1]
Technology	16 nm FinFET	7 nm FinFET	7 nm FinFET	5 nm FinFET	28 nm CMOS	16 nm FinFET
Data Rate (Gb/s)	56	112	106.25	113	52	160
Modulation	NRZ	PAM-4	PAM-4	PAM-4	PAM-4	NRZ
Bandwidth (GHz)	28	28	26.56	28.25	13	80
Channel Loss (dB) @ BER	8 @ 1e-15	4 @ 6e-11 7 @ 1.3e-10	10 @ 1e-10	11.5 @ 1e-6 0.8 @ 1e-12	7.1 @ 1e-12	3 @ 1e-12
TX Equalization	None	5-tap FFE	3-tap FFE	6-tap FFE	-	2-tap FFE
RX Equalization	CTLE	CTLE	CTLE	CTLE	CTLE	1-tap MLSE
RX Power (mW)	-[2]	-[2]	69	-[2]	43.1	332.6
RX Energy Eff. (pJ/bit)	-[2]	-[2]	0.65	-[2]	0.83	2.08
Total Energy Eff. (pJ/bit)	2.25	1.7	1.55	1.55	0.83[3]	4.08

[1] Simulated results

[2] No RX-only power reported

[3] RX-only work

Table 4.3: Performance table

4.6 Conclusions

This chapter reports the implementation of a complete 160 Gb/s NRZ receiver designed in a 16 nm FinFET process. The analog MLSE datapath is integrated with the multi-phase sub-baud-rate clocks. Datapath calibration techniques are proposed to adapt the MLSE tap coefficients while handling gain and offset mismatch per each interleaved way of the equalizer. The receiver is simulated to operate with a 3 dB loss channel at an energy efficiency of 2.08 pJ per bit.

Chapter 5

Conclusion

5.1 Thesis Summary

The continual scaling of computing and communication networks causes an increasing demand for high-bandwidth, energy-efficient die-to-die links. These links must implement equalizers that improve error margins for noise-limited, low-loss channels while minimizing power consumption. As we look toward throughput rates of 100 GBaud/s and beyond, conventional wireline equalization techniques face bandwidth limitation or noise enhancement problems, resulting in power-hungry or infeasible designs. This necessitates the exploration of new equalization architectures which remove such circuit bottlenecks and enable a higher rate of operation.

The MLSE algorithm is chosen as the starting point for the equalizer architecture exploration, as it is considered the optimum receiver for band-limited channels with additive white Gaussian noise. Classical implementations of the MLSE equalizer such as the Viterbi algorithm introduce feedback to reduce computational complexity. Feedforward implementations of the MLSE algorithm have significant complexity overhead and must resolve conflicts in overlapping windows. Targeting a 1-tap postcursor channel with coefficients $h = [1, \alpha]$, various 1-tap postcursor MLSE-based designs were explored to simplify the algorithmic computations while managing the BER penalty. The single window MLSE architecture seemed to provide the best complexity-error statistics trade-off for low-loss channels, with the window length 2 version achieving similar error rates as the DFE for $\alpha \leq 0.3$ and the window length 3 version meeting the same benchmark up to $\alpha \leq 0.4$.

The proposed window length 2 feedforward MLSE was then designed for a 160 Gb/s NRZ receiver in a 16 nm FinFET process. By removing the feedback loop, the bottleneck from the equalizer was eliminated. However, this revealed challenges in the design of frontend circuits, primarily the T&H, as device parasitics dominated and could incur significant loss if not carefully designed. A T&H design flow codified in the Berkeley Analog Generator (BAG) framework enabled automatic design space exploration and design optimization. Current integrating latches reduce power consumption of the mixed-signal datapath, resulting in a

simulated datapath energy efficiency of 0.72 pJ/bit.

The receiver datapath is integrated with a multi-phase 20 GHz and 10 GHz generating clocking path. The digital backend incorporates crucial testing features which provide visibility into the analog datapath, and an off-chip testing framework applies independent per-way adaptation and calibration for the equalizer, minimizing error sources from process variation. The receiver is simulated to run at 160 Gb/s with a 2.08 pJ/bit efficiency while equalizing a 8.5 mm on-package channel with a loss of 3 dB.

5.2 Future Works

This work's receiver was primarily designed to target very low loss channels for die-to-die link applications. Hence, the receiver was constructed with only a 1-tap feedforward MLSE for equalization. Longer reach wireline transceivers often combine different equalization techniques to cover different sources of ISI (e.g., FFE and DFE for near-tap and CTLE for long-tail ISI). For channels with $>$ 10 dB of loss, combining the proposed feedforward MLSE equalizer with the receiver FFE and CTLE could prove to be a promising equalization strategy for transceivers operating at 100+ GBaud/s, where the DFE's feedback timing constraint cannot be met.

The proposed feedforward equalization scheme itself is a very simplified version of the original MLSE algorithm, with the primary aim to manage complexity given the challenges of designing AMS datapaths. As longer-reach standards transition to PAM-4, most wireline receivers are built as ADC-based links, where the bulk of the time-domain equalization is implemented in the digital backend. A DSP-based feedforward MLSE could support more taps or longer window length, achieving better error statistics with easier implementation.

Bibliography

[1] S. Mirabbasi, L. C. Fujino, and K. C. Smith. "Through the Looking Glass—The 2023 Edition: Trends in solid-state circuits from ISSCC". In: *IEEE Solid-State Circuits Magazine* 15.1 (2023), pp. 45–62. DOI: 10.1109/MSSC.2022.3222727.

[2] N. Tracy et al. *112 Gbps Electrical Interfaces – An OIF Update on CEI-112G*. https://www.oiforum.com/wp-content/uploads/00311c-OIF-112G-OFC-slides_ofc20_presentation.pdf. 2020.

[3] G. Moore. "Cramming More Components onto Integrated Circuits". In: *Electronics Magazine* 38.8 (1965), pp. 114–117.

[4] D. Das Sharma. *Universal Chiplet Interconnect express (UCIe): Building an open chiplet ecosystem*. Tech. rep. UCIe Consortium, Mar. 2022. URL: https://www.uciexpress.org/_files/ugd/0c1418_c5970a68ab214ffc97fab16d11581449.pdf.

[5] J. Proakis and M. Salehi. *Digital Communications*. 5th ed. McGraw-Hill, 2008.

[6] V. Stojanović. "Channel-Limited High-Speed Links: Modeling, Analysis and Design". PhD thesis. Department of Electrical Engineering, Stanford University, 2004. URL: http://www-vlsi.stanford.edu/people/alum/pdf/0409_Stojanovic_Link_Opt.pdf.

[7] Y. Lu. "Energy-Efficient Equalization Circuits for High-Speed Wireline Links". PhD thesis. EECS Department, University of California, Berkeley, Dec. 2016. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-178.html.

[8] C. Thakkar et al. "Design Techniques for a Mixed-Signal I/Q 32-Coefficient Rx-Feedforward Equalizer, 100-Coefficient Decision Feedback Equalizer in an 8 Gb/s 60 GHz 65 nm LP CMOS Receiver". In: *IEEE Journal of Solid-State Circuits* 49.11 (2014), pp. 2588–2607. DOI: 10.1109/JSSC.2014.2360917.

[9] J. Kim et al. "A 112 Gb/s PAM-4 56 Gb/s NRZ Reconfigurable Transmitter With Three-Tap FFE in 10-nm FinFET". In: *IEEE Journal of Solid-State Circuits* 54.1 (2019), pp. 29–42. DOI: 10.1109/JSSC.2018.2874040.

[10] J. Kim et al. "A 224-Gb/s DAC-Based PAM-4 Quarter-Rate Transmitter With 8-Tap FFE in 10-nm FinFET". In: *IEEE Journal of Solid-State Circuits* 57.1 (2022), pp. 6–20. DOI: 10.1109/JSSC.2021.3108969.

[11] J. Paramesh and D. J. Allstot. "Analysis of the Bridged T-Coil Circuit Using the Extra-Element Theorem". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 53.12 (2006), pp. 1408–1412. DOI: 10.1109/TCSII.2006.885971.

[12] B. Razavi. "The Design Of Broadband I/O Circuits [The Analog Mind]". In: *IEEE Solid-State Circuits Magazine* 13.2 (2021), pp. 6–15. DOI: 10.1109/MSSC.2021.3072299.

[13] E. Chang et al. "BAG2: A process-portable framework for generator-based AMS circuit design". In: *2018 IEEE Custom Integrated Circuits Conference (CICC)*. 2018, pp. 1–8. DOI: 10.1109/CICC.2018.8357061.

[14] J. Crossley et al. "BAG: A designer-oriented integrated framework for the development of AMS circuit generators". In: *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2013, pp. 74–81. DOI: 10.1109/ICCAD.2013.6691100.

[15] R. Bai, S. Palermo, and P. Y. Chiang. "2.5 A 0.25pJ/b 0.7V 16Gb/s 3-tap decision-feedback equalizer in 65nm CMOS". In: *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 2014, pp. 46–47. DOI: 10.1109/ISSCC.2014.6757331.

[16] E. Chang et al. "An Automated SerDes Frontend Generator Verified With a 16-nm Instance Achieving 15 Gb/s at 1.96 pJ/bit". In: *IEEE Solid-State Circuits Letters* 1.12 (2018), pp. 245–248. DOI: 10.1109/LSSC.2019.2911404.

[17] T. O. Dickson, J. F. Bulzacchelli, and D. J. Friedman. "A 12-Gb/s 11-mW half-rate sampled 5-tap decision feedback equalizer with current-integrating summers in 45-nm SOI CMOS technology". In: *2008 IEEE Symposium on VLSI Circuits*. 2008, pp. 58–59. DOI: 10.1109/VLSIC.2008.4585951.

[18] J. Han et al. "Design Techniques for a 60 Gb/s 173 mW Wireline Receiver Frontend in 65 nm CMOS Technology". In: *IEEE Journal of Solid-State Circuits* 51.4 (2016), pp. 871–880. DOI: 10.1109/JSSC.2016.2519389.

[19] J. Han et al. "Design Techniques for a 60-Gb/s 288-mW NRZ Transceiver With Adaptive Equalization and Baud-Rate Clock and Data Recovery in 65-nm CMOS Technology". In: *IEEE Journal of Solid-State Circuits* 52.12 (2017), pp. 3474–3485. DOI: 10.1109/JSSC.2017.2740268.

[20] K. Huang et al. "A 190mW 40Gbps SerDes transmitter and receiver chipset in 65nm CMOS technology". In: *2015 IEEE Custom Integrated Circuits Conference (CICC)*. 2015, pp. 1–4. DOI: 10.1109/CICC.2015.7338370.

[21] B. Kim et al. "A 10-Gb/s Compact Low-Power Serial I/O With DFE-IIR Equalization in 65-nm CMOS". In: *IEEE Journal of Solid-State Circuits* 44.12 (2009), pp. 3526–3538. DOI: 10.1109/JSSC.2009.2031015.

[22] M. Park et al. "A 7Gb/s 9.3mW 2-Tap Current-Integrating DFE Receiver". In: *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*. 2007, pp. 230–599. DOI: 10.1109/ISSCC.2007.373378.

[23] B. Nikolic et al. "Improved sense-amplifier-based flip-flop: design and measurements". In: *IEEE Journal of Solid-State Circuits* 35.6 (2000), pp. 876–884. DOI: 10.1109/4.845191.

[24] K. Kim et al. "A 2.6mW 370MHz-to-2.5GHz Open-Loop Quadrature Clock Generator". In: *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*. 2008, pp. 458–627. DOI: 10.1109/ISSCC.2008.4523255.

[25] J. J. O'Reilly. "Series-parallel generation of m-sequences". In: *Radio and Electronic Engineer* 45.4 (1975), pp. 171–176. DOI: 10.1049/ree.1975.0033.

[26] S. Dasgupta, C. Johnson, and A. Baksho. "Sign-sign LMS convergence with independent stochastic inputs". In: *IEEE Transactions on Information Theory* 36.1 (1990), pp. 197–201. DOI: 10.1109/18.50391.

[27] V. Stojanović et al. "Autonomous dual-mode (PAM2/4) serial link transceiver with adaptive equalization and data recovery". In: *IEEE Journal of Solid-State Circuits* 40.4 (2005), pp. 1012–1026. DOI: 10.1109/JSSC.2004.842863.

[28] M. Erett et al. "A 2.25pJ/bit Multi-lane Transceiver for Short Reach Intra-package and Inter-package Communication in 16nm FinFET". In: *2019 IEEE Custom Integrated Circuits Conference (CICC)*. 2019, pp. 1–8. DOI: 10.1109/CICC.2019.8780221.

[29] R. Yousry et al. "11.1 A 1.7pJ/b 112Gb/s XSR Transceiver for Intra-Package Communication in 7nm FinFET Technology". In: *2021 IEEE International Solid- State Circuits Conference (ISSCC)*. Vol. 64. 2021, pp. 180–182. DOI: 10.1109/ISSCC42613.2021.9365752.

[30] R. Shivnaraine et al. "11.2 A 26.5625-to-106.25Gb/s XSR SerDes with 1.55pJ/b Efficiency in 7nm CMOS". In: *2021 IEEE International Solid- State Circuits Conference (ISSCC)*. Vol. 64. 2021, pp. 181–183. DOI: 10.1109/ISSCC42613.2021.9365975.

[31] G. Gangasani et al. "A 1.6Tb/s Chiplet over XSR-MCM Channels using 113Gb/s PAM-4 Transceiver with Dynamic Receiver-Driven Adaptation of TX-FFE and Programmable Roaming Taps in 5nm CMOS". In: *2022 IEEE International Solid- State Circuits Conference (ISSCC)*. Vol. 65. 2022, pp. 122–124. DOI: 10.1109/ISSCC42614.2022.9731636.

[32] S. Park et al. "A 0.83pJ/b 52Gb/s PAM-4 Baud-Rate CDR with Pattern-Based Phase Detector for Short-Reach Applications". In: *2023 IEEE International Solid- State Circuits Conference (ISSCC)*. 2023, pp. 118–120. DOI: 10.1109/ISSCC42615.2023.10067541.

Appendix A

Detailed Error Statistical Analysis of MLSE Equalizers

A.1 MLSE Window Length 1: Lower Bound BER

Recall the that the lower bound BER for the window length 1 MLSE can be written as (2.17). $d_t[n]$ is assumed to be 1 without loss of generality.

We can first solve for $P(W_2(n) = 0)$, using the law of total probability to condition on $d_t[n-1]$:

$$\begin{aligned} P(W_2(n) = 0) &= P(v_r[n] < 0) \\ &= P(d_t[n-1] = 1)P(v_n[n] + (1+\alpha) < 0) \\ &+ P(d_t[n-1] = 0)P(v_n[n] + (1-\alpha) < 0) \\ &= \frac{1}{2}\left[P\left(v_n[n] + (1+\alpha) < 0\right) + P\left(v_n[n] + (1-\alpha) < 0\right)\right] \\ &= \frac{1}{2}[F_n(-1-\alpha) + F_n(-1+\alpha)] \end{aligned}$$

where F_n is the CDF of the noise distribution. Next, we can similarly solve for $P(W_1(n+1) = 0)$:

$$\begin{aligned} P(W_1(n+1) = 0) &= P(v_r[n+1] < -1 \cup 0 < v_r[n+1] < 1) \\ &= P(d_t[n+1] = 1)P(v_n[n+1] + (1+\alpha) < -1 \\ &\qquad \cup\, 0 < v_n[n+1] + (1+\alpha) < 1) \\ &+ P(d_t[n+1] = 0)P(v_n[n+1] + (-1+\alpha) < -1 \\ &\qquad \cup\, 0 < v_n[n+1] + (-1+\alpha) < 1) \end{aligned}$$

$$\begin{aligned} P(W_1(n+1)=0) &= \frac{1}{2}[P(v_n[n+1]+(1+\alpha)<-1\cup 0<v_n[n+1]+(1+\alpha)<1) \\ &\quad + P(v_n[n+1]+(-1+\alpha)<-1\cup 0<v_n[n+1]+(-1+\alpha)<1)] \\ &= \frac{1}{2}[2F_n(-\alpha)+F_n(-2-\alpha)+F_n(2-\alpha)-F_n(-1-\alpha)-F_n(1-\alpha)] \end{aligned}$$

If α is positive and standard deviation of the noise σ_n is sufficiently lower than 1, then $F_n(-\alpha) \gg F_n(-1-\alpha), F_n(-2-\alpha)$. Thus, $F_n(-1-\alpha)$ and $F_n(-2-\alpha)$ can be safely removed from the equation, resulting in

$$P(W_1(n+1)=0) \approx \frac{1}{2}[2F_n(-\alpha)+F_n(2-\alpha)-F_n(1-\alpha)]$$

Plugging these terms back into (2.17), we get

$$BER \approx \frac{1}{4}[F_n(-1-\alpha)+F_n(-1+\alpha)][2F_n(-\alpha)+F_n(2-\alpha)-F_n(1-\alpha)] \tag{A.1}$$