

**A MAXIMUM LIKELIHOOD SEQUENCE EQUALIZING  
ARCHITECTURE USING VITERBI ALGORITHM FOR ADC-BASED  
SERIAL LINK**

An Undergraduate Research Scholars Thesis

by

ARSHAD KAMRUZ ZAMAN

Submitted to the Undergraduate Research Scholars program at  
Texas A&M University  
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor:

Dr. Samuel Palermo

May 2018

Major: Electrical Engineering

# TABLE OF CONTENTS

	Page
ABSTRACT.....	1
ACKNOWLEDGMENTS .....	2
NOMENCLATURE .....	3
CHAPTER	
I.    INTRODUCTION .....	4
Current Concerns with ADC Equalizer .....	5
Maximum Likelihood Sequence Estimator (MLSE) .....	7
II.   METHODS .....	9
Theoretical Framework of Viterbi Algorithm.....	9
Modeling Viterbi Algorithm.....	15
III.  RESULTS .....	20
IV.  CONCLUSION.....	24
REFERENCES .....	25

## **ABSTRACT**

A Maximum Likelihood Sequence Equalizing Architecture Using Viterbi Algorithm for ADC-Based Serial Link

Arshad Kamruz Zaman  
Department of Electrical & Computer Engineering  
Texas A&M University

Research Advisor: Dr. Samuel Palermo  
Department of Electrical & Computer Engineering  
TEXAS A&M UNIVERSITY

Channel impairments in high data rates make Analog-to-digital (ADC) serial link a very attractive choice in terms of bandwidth efficient modulation; however, power limitation of these receivers make the ADC front-end design rather challenging [3]. By replacing traditional symbol-by-symbol digital equalizer with a maximum likelihood sequence estimator (MLSE) receiver, in ADC serial link, we can produce a more optimal equalizing architecture in terms of noise, and simplify the complexity of the design in the analog front-end [7]. MLSE architecture is implemented using the Viterbi algorithm, in Matlab, and the parameters for the analog front-end circuits were defined by plotting the bit error rate (BER) as a function of different SNRs. Comparing the BER between the traditionally used MMSE equalizer and MLSE receiver BER was found to be lower for same SNR. Although using the Viterbi algorithm to determine the original signal sequence may make MLSE computationally challenging, the simplicity of analog front-end and lower BER makes this an effective choice for high bandwidth transmission in a digital-heavy receiver.

## **ACKNOWLEDGMENTS**

I would like to thank Dr. Palermo for providing me with this opportunity to research with his team, and I want to acknowledge the Ph.D. students, Shengchang Cai, Shiva Kiran, Shangfeng Qiu, and Yuanming Zhu for their guidance and support throughout the course of this research.

I want to appreciate my friends, colleagues, the department faculty, and staff for making my time at Texas A&M University a great experience. Finally, a special thanks to my family members, my mother, father, and little brother for their encouragements and always being there for me.

## NOMENCLATURE

ADC	Analog-to-Digital
BER	Bit Error Rate
CTLE	Continuous time linear equalizer
DFE	Decision Feedback Equalizer
DSP	Digital Signal Processing
FFE	Fast Forward Equalizer
ISI	Inter-Symbol Interference
MLSE	Maximum Likelihood Sequence Estimator
NRZ	Non-Return-to-Zero
PAM	Pulse Amplitude Modulation
SAR	Successive Approximation
SNR	Signal-to-Noise Ratio
CMOS	Complementary metal–oxide–semiconductor

# CHAPTER I

## INTRODUCTION

As the data rates of wireline communication link increase, channel impairments (i.e. skin effect, noise, dielectric loss, fiber dispersion) and inter-symbol interference become more of an issue [2]. Channel impairment in the following system can be caused by multipath propagation. Inter-symbol interference (ISI) is the distortion of a signal where one or multiple symbols (symbol are represented as bits) interferes with subsequent symbol causing the original symbol to appear differently from its intended appearance [1]. Fig. 1 gives a demonstration of how ISI graph appears during a multi-bit transmission. As shown in Fig. 1, when the sinusoidal pulse reaches the amplitude, the transmitted bit should read to be “1”, and when the pulse is at zero the transmitted bit is read to be “0”; since ISI’s job is to convolve the bits that are next to each other, the sinusoidal curve may not behave accordingly [9]. For example, in Fig. 1, although

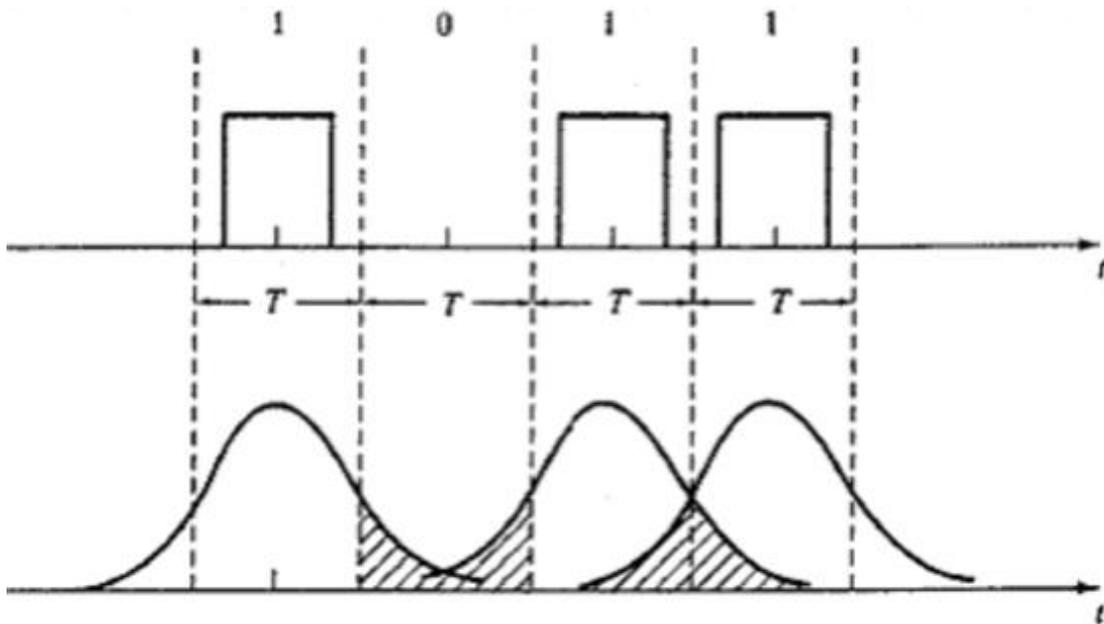


Fig. 1. Inter-symbol interference (ISI) of a transmitted signal.

the sinusoidal curve immediately after the 3<sup>rd</sup> transmitted bit should be at zero, the interference from 3<sup>rd</sup> and 4<sup>th</sup> bit made the curve appeared to be somewhere in between the amplitude and zero, which hinders the users from realizing the original magnitude of the transmitted bits. Since every single bit are affected by ISI, converting it back to the originally transmitted bits can be very challenging; analog-to-digital (ADC) converter is of interest in order to provide for these impairments in a digital domain [1]. ADC based serial link receiver allows more complex and flexible digital signal processing using which complex digital equalization and more bandwidth-efficient modulation scheme can be performed [3]. Another advantage of using digital equalizer is its robust tolerance to variation and better scalability with process technology.

### **Current Concerns with ADC Equalizer**

One of the key issues with ADC based receiver is the amount of power these receivers usually consumes. This issue can be minimized using successive approximation (SAR) ADCs instead of the traditional flash ADC's in the respective systems [2]. SAR ADC converts an analog waveform to a digital form using a binary search, where it constrains an input from a continuous set of values and derives a discrete set as output. A flash ADC, in comparison, converts analog waveform into a digital representation using a linear voltage ladder with a comparator at each step of the ladder; these comparator directly converts analog to digital symbols by comparing the input voltage to the successive reference voltage [10]. While performing this conversion, it is very important to take noise into account as well. In a communication system, noise is an unwanted random disturbance and error that distorts the transmitted signal. While a signal is being transmitted through a channel, a receiver must account for noises to properly trace back to the original signal. Although effective in SAR ADC, symbol-by-symbol digital equalizers are not most optimal in terms of noise [2].

MLSE architecture will be a better approach because, in theory, it is the most optimal equalizer in presence of noise [7]. In a conventional ADC serial link, represented by Fig. 2, the input signal initially travels through a channel. Channel refers to the way signal flows throughout the organizations [10]. ISI and noises are usually introduced during this section of the block diagram. Continuous time linear equalizer (CTLE) is used to improve the link performance; CTLE circuit works as a filter by attenuating low-frequency signal component, amplifies the component around the minimum rate at which a signal can be sampled without introducing errors (Nyquist frequency), and removes higher frequencies [2]. Fast Forward Equalizer (FFE) works by modifying the amplitude of the bits surrounding transitions while keeping the transmitted power constant. As discussed previously ADC is used to convert the analog signal to a digital domain, than a decision feedback equalizer (DFE) is used to produce an estimate of the channel output. Digital signal processing (DSP) is used to improve the accuracy and reliability of the overall serial link by using DFE and FFE [5]. Due to the ability to perform in presence of noise, the sequence estimator can reduce these requirements of the front end circuit blocks, as it statistically performs equalization in the digital domain, and make the design of analog front-end to be less complicated [6].

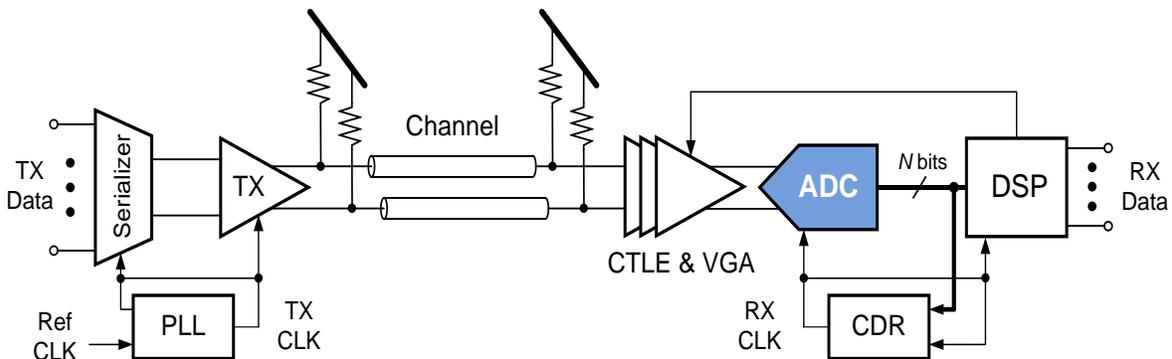


Fig. 2. ADC Based Serial Link.

## Maximum Likelihood Sequence Estimator (MLSE)

MLSE works by taking inter-symbol interference (ISI) into consideration during demodulation, rather than eliminating it using the transmitter/receiver filter [7]. Instead of minimizing the bit error probability directly, MLSE focuses more on minimizing the sequence error probability, and by doing so it also contains the noise power at the output. This equalization is performed in the DSP using a PAM-4 receiver (which will be explained later in the paper). During demodulation receiver sees  $M = 2^N$  possible signals due to the  $N$  transmitted bits, MLSE makes a decision on which of the  $M$  signal was transmitted by assessing the signal's probability of sequence error [7]. MLSE can do this by determining the orthogonal function of all possible signals "M", and projecting the received signal  $r(t) = s_i(t) + w(t)$ , where  $w(t)$  is noise, to generate observables. Finally, MLSE assumes the bit sequence to be equally probable and determines the decision rule to minimize the sequence probability error. However, this algorithm, just by itself, is infeasible because of the number  $M$ , which usually has a very large value. For example, if the length of the transmitted bit is extremely small (usually not the case)  $N = 100$  bits, the receiver will see  $M = 2^{100}$  possibilities, which is equivalent to  $10^{31}$ . Analyzing  $10^{31}$  possible signals using this method will be extremely trivial if not impossible; this method is made practical using Viterbi Algorithm [7].

Viterbi Algorithm uses state and branch metrics, which are determined using the present output and the bit value of the matched filter, and the previous values of the considered bit pattern [8]. Due to these considerations, the system will have memory of the ISI terms, which is used to compute the branch metrics [7]. Viterbi then uses a finite state diagram for each of the branches to determine a trellis graph, which yields in branch metrics [7]. The determination of the signal being transmitted is made by finding the best path through the trellis [9]. The best path

is determined by assessing the signals with the shortest distance depending on the transmitted sequence, channel impairments, and noise [7]. Finally, the sequence of the transmitted bits can be found by tracing back the trellis graph of the best state metrics.

## CHAPTER II

### METHODS

#### Theoretical Framework of Viterbi Algorithm

In ADC, ISI causes bandwidth limitation, which in previous models, have been combated by using transmitter/receiver filter. A drawback to these filter is it enhances the noise power at its output and degrades performance [7]. Unlike traditional method, MLSE deals with the ISI and noises as it is demodulating in the DSP. Receivers register  $N$  bits and sees  $M$  possible sequence of signals where  $M = 2^N$  [7]. Viterbi algorithm is then used to determine maximum likelihood sequence by figuring out the decision rule and the distance of the trellis path of each of these signals.

For a better understanding of the theoretical framework of Viterbi Algorithm, an example is considered. In this example, the impulse response ( $h(t)$ ) is shown using Fig. 3(a), where  $T_B$  is represented as the bit interval. Impulse response in a dynamic system is usually considered to be reactions to external changes. Fig. 3(b) displays a signal with a delayed copy of 4(a) as a function of delay, it also includes the ISI terms which will be considered by Viterbi as decoding is being done [7]. As Shown in 4b,  $h_1$  and  $h_2$  taps represent the ISI caused by bits prior to the bit which is being analyzed; therefore, if  $N$  is the total number of transmitted bits and  $k$ th bit is currently being analyzed,  $h_1$  and  $h_2$  is due to  $b_{i, k-1}$  and  $b_{i, k-2}$  bits in the sequence. The subscript “ $i$ ” represents the specific bit sequence [7]. In this model, memory length is two bits, since the system needs to remember the previous two bits, which are  $b_{i, k-1}$  and  $b_{i, k-2}$ , to provide for ISI. A state diagram approach is used to hold the value of these two bits [7]. Memory length ( $M_L$ ) can

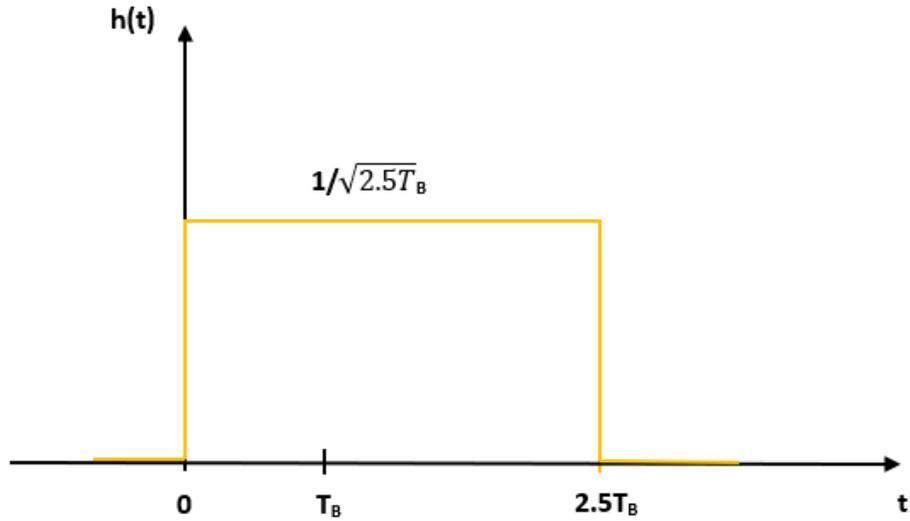


Fig. 3(a). Viterbi Impulse Response.

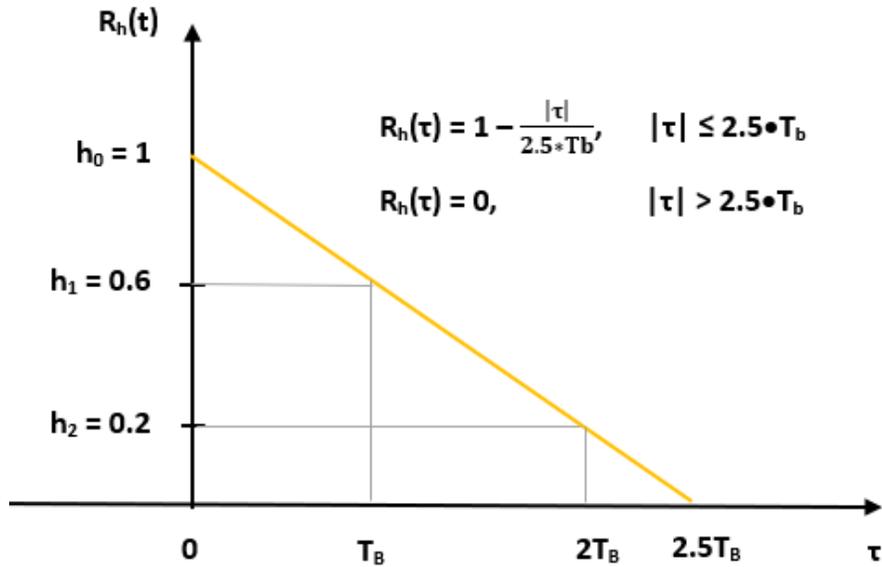


Fig. 3(b). Autocorrelated Impulse Response.

be defined using Eq. 1.1, where  $L$  is the total number of bits considered to analyze a single bit in the sequence.

Branch metrics can be derived by using Eq. 1.2, where  $r_k$  is the current output of the matched filter. After substituting the value for  $h_0$  and  $h_1$ , the branch metrics for this system will

become  $b_{i,k} \cdot r_k - 0.6 \cdot b_{i,k} \cdot b_{i,k-1} - 0.2 \cdot b_{i,k} \cdot b_{i,k-2}$ . Although  $b_k$  is expressed as 0 and 1, since  $b_k$  is transmitted as an impulse of strength, branch metrics uses -1 and +1 for computational purposes for PAM-2 modulation [7]. Table I provides a brief demonstration of how branch metrics is computed as different bits are considered for  $b_{i,k}$ ,  $b_{i,k-1}$ , and  $b_{i,k-2}$ .

$$M_L = L - 1 \quad (1.1)$$

$$\text{Branch Metrics} = b_{i,k} \cdot r_k - h_1 \cdot b_{i,k} \cdot b_{i,k-1} - h_2 \cdot b_{i,k} \cdot b_{i,k-2} \quad (1.2)$$

Table I. Branch Metric Computation.

$b_{i,k}$	$b_{i,k-1}$	$b_{i,k-2}$	Branch metrics
0 (-1)	0 (-1)	0 (-1)	$r_k \cdot (-1) - 0.6 \cdot (-1) \cdot (-1) - 0.2 \cdot (-1) \cdot (-1) = -r_k - 0.8$
0 (-1)	0 (-1)	1 (+1)	$r_k \cdot (-1) - 0.6 \cdot (-1) \cdot (-1) - 0.2 \cdot (-1) \cdot (+1) = -r_k - 0.4$
0 (-1)	1 (+1)	1 (+1)	$r_k \cdot (-1) - 0.6 \cdot (-1) \cdot (+1) - 0.2 \cdot (-1) \cdot (+1) = -r_k + 0.8$
1 (+1)	1 (+1)	1 (+1)	$r_k \cdot (+1) - 0.6 \cdot (+1) \cdot (+1) - 0.2 \cdot (+1) \cdot (+1) = +r_k - 0.8$

As mentioned above, a state diagram is used to draw trellis for this ISI example. Trellis is a visual representation of all the possible branches that can be taken from  $b_{i,1}$  to  $b_{i,N}$ . Viterbi chooses the shortest branch through the trellis [7]. In this framework, an initial state of the trellis was chosen to be 00, which must agree with both the transmitter and the receiver. Trellis becomes fully developed at  $M_L \cdot T_b$ ; therefore, the trellis is fully developed after  $2 \cdot T_b$  intervals. Computationally, if two path diverges at a state, it would take at least L interval before they can meet at a state [7]. As input bits are received, states of the trellis (in the state diagram) gets updated with the most significant bit being  $b_{i,k-1}$  and least significant bits being  $b_{i,k-2}$ . Since a PAM-4, four distinct pulse amplitude levels are used to carry information, demodulation scheme

was used instead of a traditional PAM-2 approach there are a total of four possibilities (00, 01, 10, 11) that are considered in the diagram [7]. Fig. 4(a) shows the difference between PAM-2 and PAM-4 demodulation scheme, and Fig. 4(b) demonstrates how next state in state diagram is found. As shown in Fig. 4(b), after each interval the value of  $b_{i, k-2}$  is updated with  $b_{i, k-1}$ , and  $b_{i, k-1}$  is updated with  $b_{i, k}$ . If  $b_{i, k}$  is 1, the dotted line is used to demonstrate transition to the next state, and if  $b_{i, k}$  is 0 solid line is used in the figure. Since each of these bits has two possible

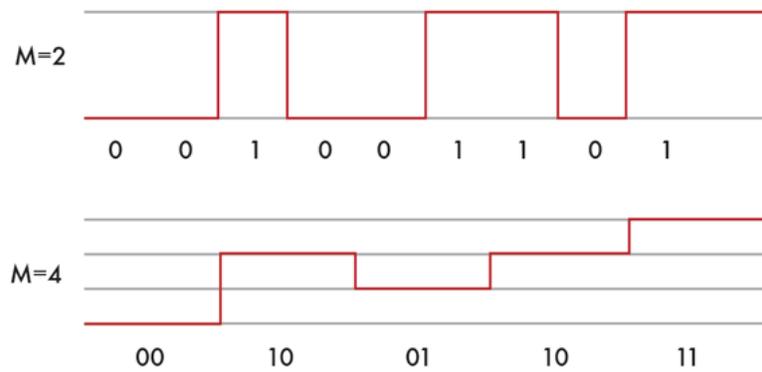


Fig. 4(a). PAM2 vs PAM-4 demodulation.

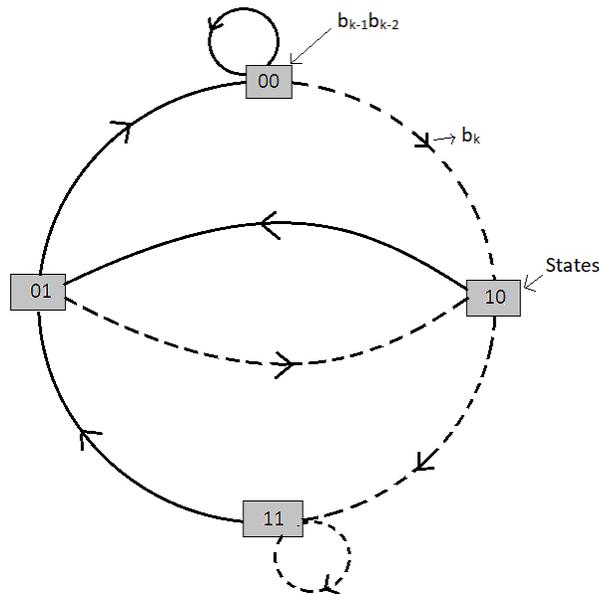


Fig. 4(b). State Diagram for Trellis.

characteristics, at every interval each state can only have two possible branches coming into it and two branches leaving it [7]. Viterbi ensures to minimize the number of those branches by taking distance into consideration. The distance of each branch is represented using received sequence,  $r_k$ , which is represented by Eq. 2.1. Furthermore,  $w_k$  is the corruption term which is also known as noise; the user must take noise variance ( $\sigma^2$ ) and signal to noise ratio (SNR) into considerations to account for noise [8]. SNR is a comparison of the level of background noise power with respect to the strength of the desired signal. SNR is computed using Eq. 2.2, where  $E_b$  (1 Joule for this model) is the energy per bit. Finally,  $y_k$  is a computed sequence of the branch metrics, which is found using Eq. 2.3.

$$r_k = y_k + w_k \quad 2.1$$

$$\text{SNR} = E_b / \sigma^2 \quad 2.2$$

$$y_k = b_k + 0.6 \cdot b_{k-1} + 0.2 \cdot b_{k-2} \quad 2.3$$

SNR was designed to be 16 dB for this example. A partial path metrics for all possible path is derived for this example using the theoretical framework. Branch metrics and distances were computed for first three bits as a demonstration. Although this example only demonstrates branch metrics with the initial state being 00, in reality (computational model discussed later) Viterbi considers branch metrics with the initial state being all 4 of the possible states [8]. A total of four path metrics are created after all intervals have been iterated. In these metrics, first two bits will be different from each other, and depend on the 4 states discussed earlier. As Viterbi Algorithm is used for the first three bits, it considers several different paths; Viterbi checks to see how many paths are going into each state during each interval (as discussed earlier, maximum number of paths going into each state is two). To minimize the total number of paths that can be taken from  $b_1$  to  $b_N$ , Viterbi ensures there is only one branch going into each state [7]. This is

done by comparing the distance of the two paths going into the states during each interval, and eliminating the paths with the longer distance [7].

As shown in Fig. 5, each state experiences the maximum number of path going into it during the third interval. For example, at  $3T_b$  state 01 had two paths going into it, one was taken by sequence 010, and the other was taken by sequence 011. The distance (branch metric) of the two possible paths at the interval was 0.884 and 1.284 respectively. Since the distance of the sequence 011 going into state 01 was larger than sequence 010, the path taken by this sequence (011) was eliminated, and 010 is considered to be the survivor path [7]. This method is used by each state to minimize the total number of paths, where each state corresponds to one path. As shown in Fig 5, towards the end of the third interval there are 4 path metrics remaining with a sequence of 000, 100, 010, 110, which have a distance of -0.301, 0.331, 3.017, and 1.249. Viterbi then compares these four distance to find the most optimal path metrics (survivor state metrics). Again, smaller distance has a higher probability of matching the original transmitted sequence;

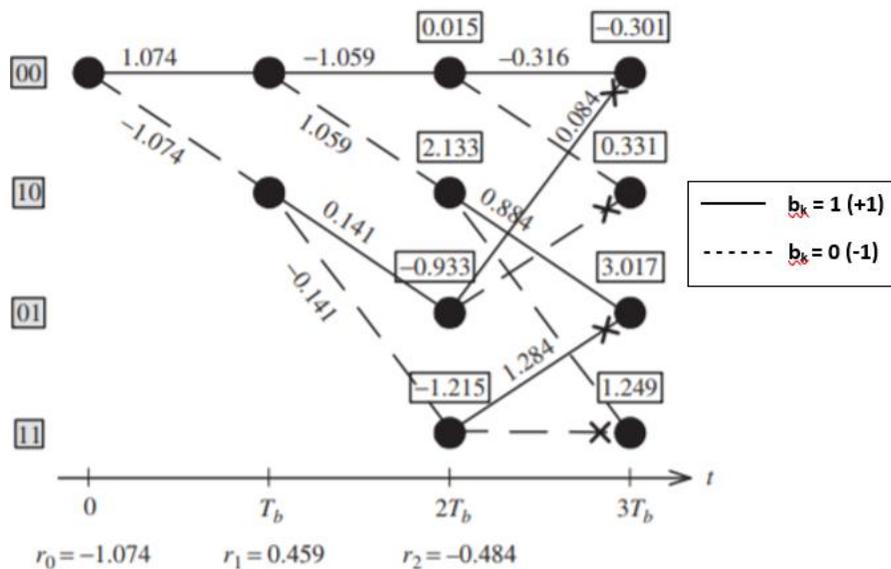


Fig. 5. Partial Path Metrics of First Three-Bit Transmission [7].

each of the distances is squared to account for negative terms, and compared against each other to find the maximum likelihood sequence [8]. In this model, path metrics with sequence 000 had the shortest distance; therefore 000 is considered to be the survivor state metrics.

### **Modeling Viterbi Decoder**

Computer simulations can be used to evaluate this architecture, and to find the efficiency of the Viterbi algorithm in an ADC-based serial link. To model this algorithm, Matlab was used and different aspects of the algorithm were broken into pieces and tested separately to ensure the overall simulation is being performed correctly.

#### *Signal Encoding Stage*

Initially, an arbitrary number of normalized integer bits (0 and 1) are defined as the transmitted signal (input metrics) in the model. Normalization of input bits was done by using the function “randn”, and the bits were converted to (-3, -1, +1, +3) to display the transmitted bits in terms of impulses for a Pam-4 modulation. Next, the transmitted signal was encoded by adding ISI and noise term into each of the bits in the transmitted sequence. ISI was added to the sequence by taking pre and post-pulses into consideration and convolving them using the “conv” function and taking the modulus of it with respect to the modulation scheme (PAM-4). To consider the noise (Eq. 3.1) in the system, noise variance (NVar) must be found. User-defined SNR and signal variance (Var), found by using the “var” function, are taken into account to compute for the noise variance (NVar), which is then computed using Eq. 3.2. Noise is then added to the convolved bits, and passed on to the decoder as a received signal [9]. A high-level block diagram of the signal encoding stage is also demonstrated using Fig. 6. As shown in this

$$\text{Noise} = \sqrt{N\text{Var}} \cdot \text{randn}(1, \text{bit}_{amt}) \quad 3.1$$

$$N\text{Var} = \frac{\text{Var}}{10^{\text{SNR}/20}} \quad 3.2$$

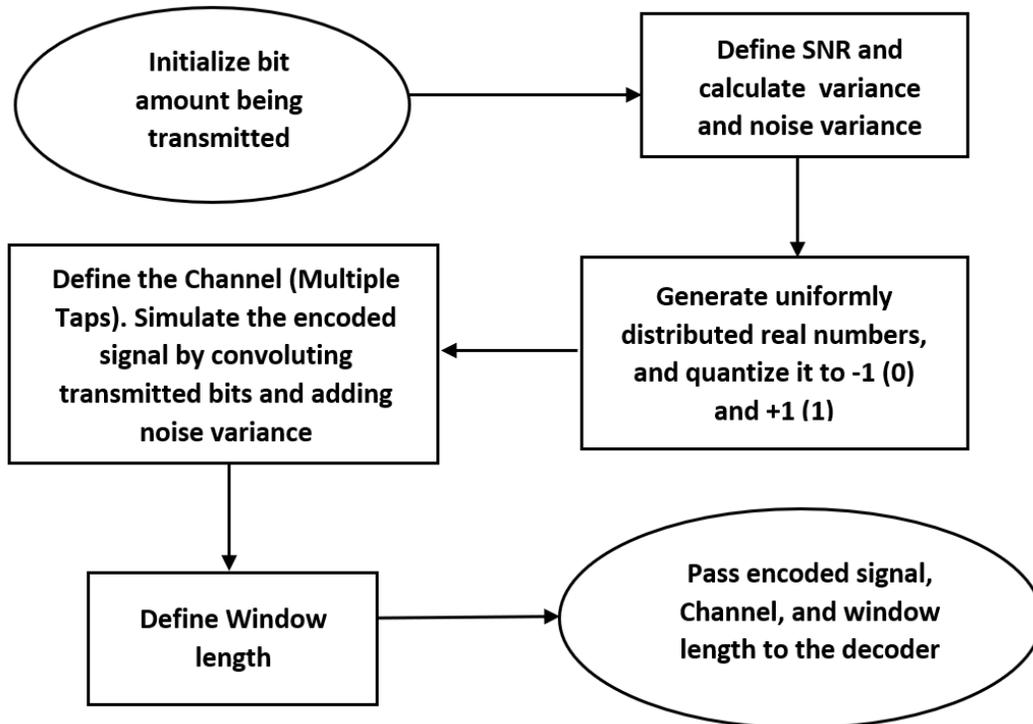


Fig. 6. Block Diagram of Signal Encoding Stage.

figure, initializing the channel, and the window length also takes place during this stage. The Viterbi decoder was modeled to handle multiple taps (also known as sampling cursor), which was previously discussed in the theoretical model using  $h_0$  and  $h_1$ . The window length is a representation of how many intervals the decoder decodes before the path metrics are compared and a conclusion is made about the survivor branch metrics. Although, decoder keeps going until the entire encoded signal has been decoded, the purpose of the window length is to relax the amount of memory space the algorithm has to allocate during the lifetime of this model. One can assume that the window length is steps used by the decoder to decode the entire encoded signal; one disadvantage of this sort of modeling is the size of the transmitted bits must be a multiple of the window length; otherwise, the model will face runtime errors. Finally, the encoded signal,

user-defined channel, and the window length is passed to the Viterbi decoder for signal decoding stage.

### Signal Decoding Stage

Decoding stage starts by receiving information from encoding stage, using which it initializes empty surviving branch and state metrics for all four possible states. The dimension of these metrics is determined by the number of bits being received. These metrics will be used to find the decoding sequence after the overall distance comparison is made. Surviving distance, which keeps track of the length of the overall path distance, is also initialized towards the beginning of this stage. To calculate the path and branch metrics the model must determine the current state given the previous state. This is done by using a function, which uses previous state and decoding bits as inputs, and produces current state as an output. A demonstration of the current state calculation, during the first interval, is shown using Fig. 7. Since, the original

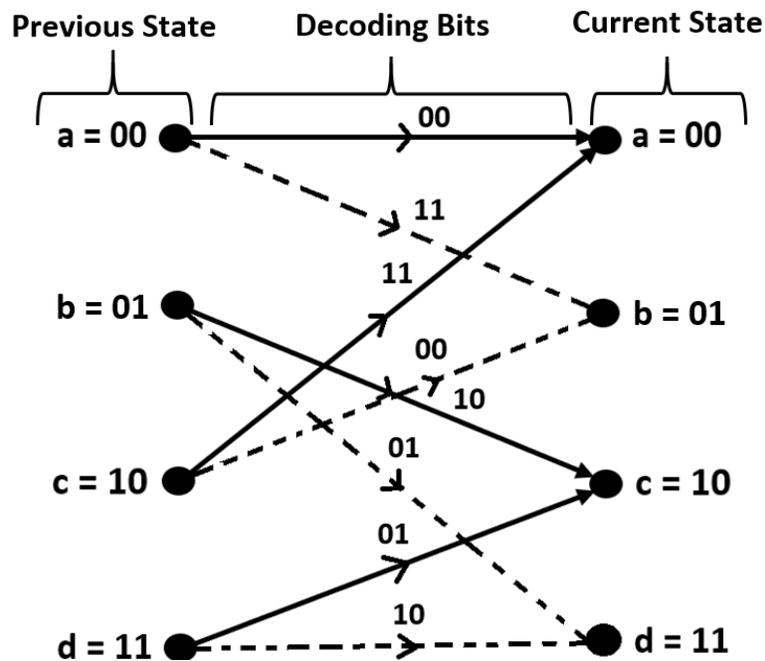


Fig. 7. Current State Calculation.

transmitted bit can follow several different paths, state calculation is done using all possible patterns for all four states.

To determine which of the two paths (going into each state) will be remaining, Viterbi needs to compare the path distances during the interval. Fig. 8 demonstrates how branch metrics are calculated. The subscript  $j$  is the number of taps being used by the model (which is user-defined). During this step, the value of  $b_{k-1}$  to  $b_{k-j}$  comes from the state function, and value of  $b_k$  depends on the decoding bit being assumed for that branch. After the comparison is done, decoding bits of the surviving branch is pushed into surviving branch metrics, and surviving distance is added to the overall distance from previous iterations, these methods are used to derive the metrics of each state. The model also keeps track of the number of decoded bits; therefore, when this number reaches the window length, surviving state metrics will be calculated to relax the amount of memory being used by the program; this step also helps with decreasing the amount of time needed to run a test.

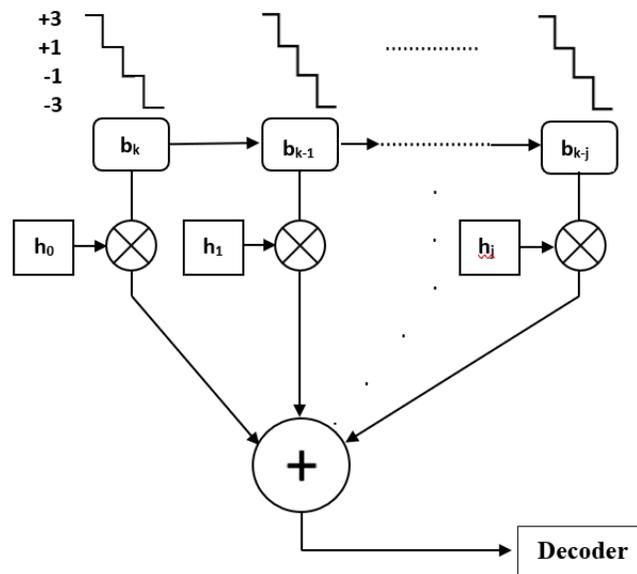


Fig. 8. Encoded branch Metrics.

The following procedure repeats itself until the size of decoded bits equal to the number of bits that are in the received signal. At that point, the decoder does a final comparison, by taking the distance of surviving metrics into consideration. The surviving state metrics corresponding the shortest path will have the bit sequence that is most likely generated by the transmitted signal. Fig. 9 demonstrates a high-level block diagram for steps taken to model the decoding stage. Viterbi traces back the surviving state metrics to acquire the decoded bit sequence. Finally, a comparison was done to see the accuracy of this model by comparing decoded bit sequence with the transmitted bit sequence.

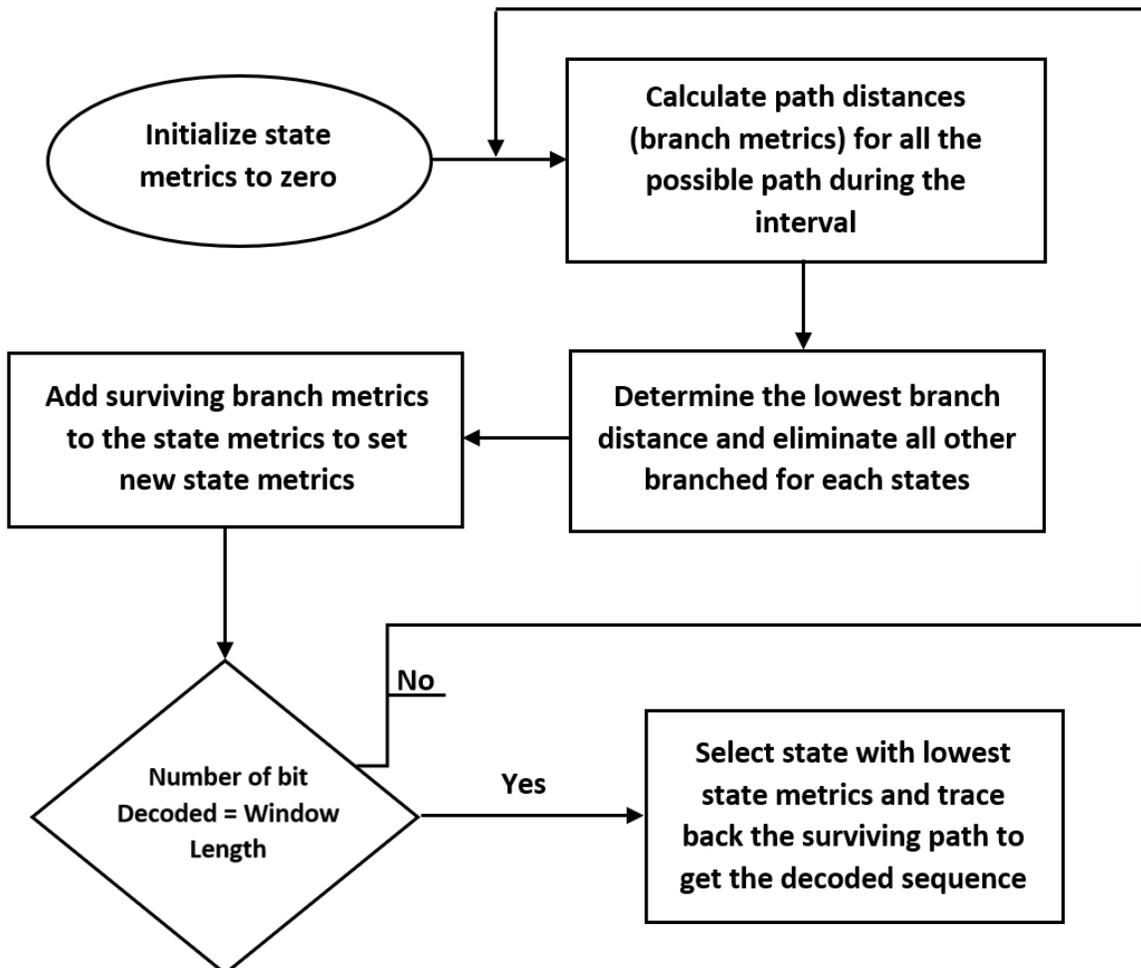


Fig. 9. Block Diagram of Signal Decoding Stage.

## CHAPTER III

### RESULTS

To assess the capabilities of this algorithm, a BER (bit error rate) vs SNR (signal to noise ratio) comparison was done with and without decoding. BER is the number of bits with errors divided by the total number of transmitted bits [9]. These comparisons helped show the impact of MLSE in terms of bit error rate with respect to the transmitted signal. Tests were broken into pieces to see the capabilities in several different environments. Fig. 10 shows the BER of the received signal without any equalization. The blue curve represents the theoretical relationship between BER and SNR in presence of noise, this curve was achieved using the “AWGN” function from Matlab’s communication toolbox. As expected, as SNR increases BER of the blue curve decreases. This happens because as SNR increase the ratio of the strength of signal carrying information with respect to interference (noise) increases [10]. Therefore, high enough

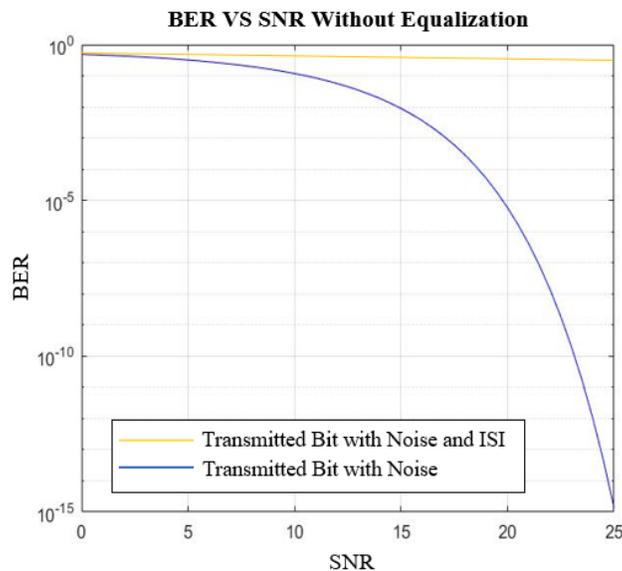


Fig. 10. BER VS SNR Without Equalization.

SNR can be used, in some cases, to provide for BER caused by noise only. The yellow curve on the figure represents the transmitted bits corrupted by ISI and noise. BER of this curve is consistently around 25% – 30% regardless of the SNR [9]. Since the transmitted signal gets corrupted by both noise and ISI before it reaches the receiver, the yellow curve will be used to compare the effectiveness of the MLSE decoder.

To assess how different channels with ISI and noise affect the BER after decoding (using MLSE), a comparison was done by changing the length of the taps. This test will provide information on what happens when the number of previous symbols (bits) being considered by ISI increases. Fig. 11 demonstrates the effectiveness of decoding using different channels. Although decoding was done after considering ISI and noise, MLSE was able to minimize the BER very close to the theoretical level (BER in presence of noise only). For consistency, the sample size of all signals were kept at constant (100,000 bits). Table II represents the channels used to model this comparison. MLSE performs better under channel A than channel B and C, this degrading performance is due to the length of taps being used by the channels [7]. Since, tap length directly corresponds to the number of bits considered by the ISI, as tap length increase channel performance worsens.

Table II. Channel Weight Coefficients.

Channel	Tap Length	$h_0$	$h_1$	$h_2$	$h_3$	$h_4$
A	2	1	0.72	0.36	N/A	N/A
B	3	1	0.72	0.36	0.21	N/A
C	4	1	0.72	0.36	0.21	0.030

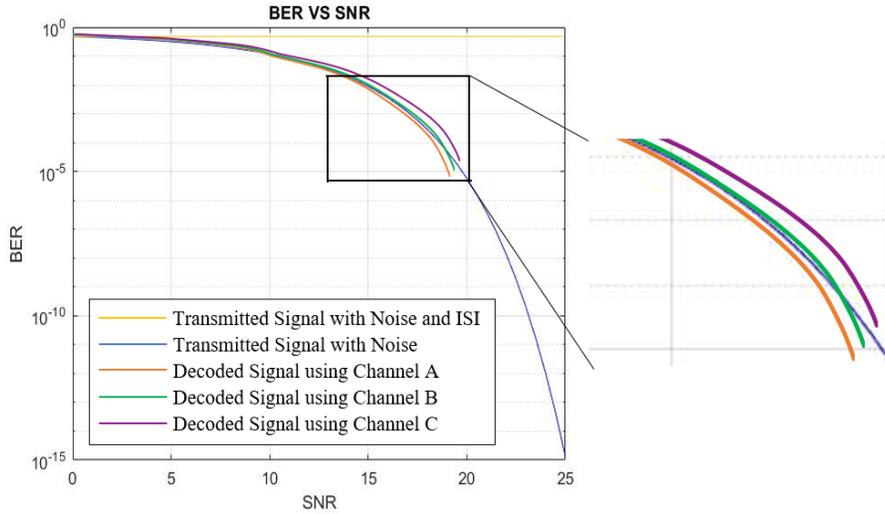


Fig. 11. BER vs SNR Comparison With Respect to Channels.

Finally, BER derived from decoded signal, using MLSE, was compared against decoded signal using traditional methods. The comparison is shown in Fig. 12. FFE and DFE were used with a number of taps to model this comparison. The coefficient values for the FFE and the DFE were computed using the MMSE criterion [7]. Same received signal (corrupted transmitted

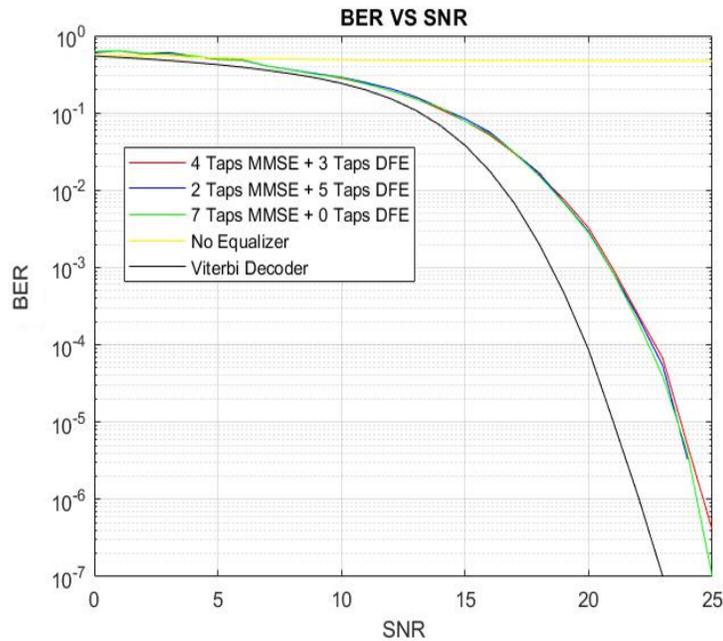


Fig. 12. Performance of MLSE vs Traditional Equalization.

signal with ISI and signal) was used for all of the decoders with a sample size of 10,000,000 bits. MLSE decoder was modeled using a 4 tap channel (1 main cursor and 3 post-cursors) with a window size of 25. As shown in the figure, although the performance of Channel C is worse than other channels demonstrated in Fig. 12, the signal decoded by MLSE (represent by the black line) still has a lower BER than signal decoded by using 4 taps MMSE with 3 taps DFE (red line), 2 taps MMSE with 5 taps DFE (blue), 7 taps MMSE with 0 taps DFE (green).

## CHAPTER IV

### CONCLUSION

MLSE Decoder was modeled using Matlab to provide for channel impairments in a high data rate communication system. The model convolves the transmitted signal with ISI and adds noise (encoding phase) before passing it to the decoder. In order to see how these impairments affect the originally transmitted signal, a BER comparison was done. After considering both ISI and noise the BER is consistently at 25% - 30%; therefore, to display the correct transmitted signal to the user equalization must be done. Without equalization, the received signal and the transmitted signal will most likely be different due to the high BER.

The effectiveness of MLSE decoder was demonstrated by obtaining the BER curve after decoding the convolved signal using Viterbi, and comparing it with BER gained after equalizing the same convolved signal with traditional methods. The traditional method yielded little above  $10^{-6}$  BER with a sample size of 10,000,000 symbols; however, the transmitted signal must have an SNR of over 25 dB to achieve this result. MLSE decoder (using Viterbi algorithm) can obtain a better result, BER  $10^{-7}$ , using an SNR of 23 dB. Due to the slope of these curves, for a lower BER, SNR difference between traditional and MLSE decoder only gets bigger. The decrease in SNR requirement simplifies the analog frontend as more complex computations are done in the DSP, which takes advantage of complementary metal–oxide–semiconductor (CMOS) technology [10].

## REFERENCES

- [1] S. Cai, E. Zhian Tabasy, A. Shafik, S. Kiran, S. Hoyos, and S. Palermo, "A 25GS/s 6b TI Binary Search ADC with Soft-Decision Selection in 65nm CMOS," *IEEE Symposium on VLSI Circuits*, 2015.
- [2] A. Shafik, E. Zhian Tabasy, S. Cai, K. Lee, S. Hoyos, and S. Palermo, "A 10Gb/s Hybrid ADC-Based Receiver with Embedded 3-Tap Analog FFE and Dynamically-Enabled Digital Equalization in 65nm CMOS," *International Solid-State Circuits Conference*, Nov. 2015.
- [3] K. Meerkotter, "On the Relation between Duration and Bandwidth of Discrete-Time Signals", *Archiv für Elektronik und Übertragungstechnik*, vol. 43, pp. 149-152, 1989.
- [4] V. Abramzon: "Analog to Digital Converter Architectures." *Principles of Data Conversion System Design*, doi:10.1109/9780470545638.ch6, 2009.
- [5] E. Zhian Tabasy, A. Shafik, K. Lee, S. Hoyos, and S. Palermo, "A 6b 10GS/s TI-SAR ADC with Low-Overhead Embedded FFE/DFE Equalization for Wireline Receiver Applications," *IEEE Journal of Solid-State Circuits*, Nov. 2014.
- [6] B. Min, N. H.-W. Yang, and S. Palermo, "10Gb/s Adaptive Receive-Side Near-End and Far-End Crosstalk Cancellation Circuitry," *IEEE International Midwest Symposium on Circuits and Systems*, Aug. 2014. [Best Student Paper Award]
- [7] H. Nguyen and E. Shwedyk, "A first course in digital communications" *Cambridge: Cambridge University Press*, pp.360-371, 2009.
- [8] J. Kim, E.-H. Chen, J. Ren, B. S. Leibowitz, P. Satarzadeh, J. L. Zerbe, and C.-K. Ken Yang, "Equalizer Design and Performance Trade-Offs in ADC-Based Serial Links" *IEEE Transactions on Circuits and Systems*, Sept. 2011.
- [9] A. J. Viterbi "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13 pp. 260-269 Apr. 1967.

- [10] K. Deguchi, N. Suwa, M. Ito, T. Kumamoto, and T. Miki, "A 6-bit 3.5-GS/s 0.9-V 98-mW flash ADC in 90-nm CMOS," *IEEE J. Solid State Circuits*, vol. 43, no. 10, pp. 2303–2310, Oct. 2008.