UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Posgrados

Viterbi Equalizer for Bluetooth Classic: Study and Design

Tesis aplicada derivada de prácticas en industria

Santiago Sebastián Pérez Dávila

Adam Quotb, Ph.D.
Director de Trabajo de Titulación

Trabajo de titulación de posgrado presentado como requisito
para la obtención del título de Magíster en Nanoelectrónica

Quito, 10 de diciembre de 2024

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO DE POSGRADOS

HOJA DE APROBACIÓN DE TRABAJO DE TITULACIÓN

Viterbi Equalizer for Bluetooth Classic: Study and Design

Santiago Sebastián Pérez Dávila

Nombre del Director del Programa:	Luis Miguel Prócel
Título académico:	Doctor of Philosophy
Director del programa de:	Maestría en Nanoelectrónica

Nombre del Decano del colegio Académico:	Eduardo Alba
Título académico:	Doctor of Philosophy
Decano del Colegio:	Colegio de Ciencias e Ingenierías

Nombre del Decano del Colegio de Posgrados:	Dario Niebieskikwiat
Título académico:	Doctor of Physics

Quito, diciembre 2024

© DERECHOS DE AUTOR

Nombre del estudiante:	Santiago Sebastián Pérez Dávila
Código de estudiante:	00335271
C.I.:	1003138946
Lugar y fecha:	Quito, 10 de diciembre de 2024.

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following graduation project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

DEDICATORIA

A mi yo de niño, que soñó con llegar hasta aquí. A mi yo del presente, que nunca se rindió. Y a mi yo del futuro, que seguirá soñando en grande.

A mis padres, luz constante y motivo para no rendirme jamás.

AGRADECIMIENTOS

Quiero expresar mi sincera gratitud a Luis Miguel Prócel, profesor y director de la Maestría en Nanoelectrónica, por su acompañamiento a lo largo del programa y por brindarme la oportunidad de involucrarme en proyectos de investigación e innovación tecnológica. Asimismo, agradezco a César Zambrano, decano de Investigación, por enseñarme el valor de la convicción en el liderazgo y la actitud profesional necesaria para destacarse.

Extiendo mi agradecimiento a Nicola di Teodoro, profesor y coordinador de la carrera de Matemáticas, por compartir su pasión por la matemática y el arte, inspirándome a valorar ambas disciplinas y su interrelación. A la Universidad San Francisco de Quito, mi alma mater, le reconozco el invaluable aporte de su formación integral y el espíritu de las artes liberales, que moldearon mi visión académica y profesional.

Este trabajo no habría sido posible sin el apoyo de Emmanuel Gautier y Pascal Blouin, quienes apreciaron mi talento y me abrieron las puertas al equipo de Silicon Labs, permitiendo que mi diseño formara parte del sistema de demodulación en el próximo chip de conectividad para IoT. También agradezco a mi supervisor Frédéric Pirot y a Fabrice Portier por su colaboración constante, sus enseñanzas teóricas y su compañerismo ejemplar en cada etapa de este proyecto.

Finalmente, extiendo mi más profundo reconocimiento a mis padres, cuyo respaldo incondicional y confianza han sido el pilar fundamental que me permitió llegar hasta aquí. Sin ellos, este logro no sería posible.

RESUMEN

Esta tesis presenta el desarrollo de un Ecualizador Viterbi para Bluetooth Classic (BTC) dentro del chip *Everest*, desarrollado por Silicon Labs para aplicaciones de IoT. El enfoque del proyecto abarca el análisis teórico, la implementación de algoritmos, el diseño de hardware y la optimización.

El trabajo sigue una metodología estructurada, comenzando con una revisión teórica sobre Bluetooth Classic, la modulación y demodulación GFSK, y el Algoritmo de Viterbi. La necesidad de un ecualizador surge debido a la presencia de Interferencia Inter-Simbólica (ISI), que degrada la precisión de los demoduladores GFSK estándar en condiciones de alto ruido. El Ecualizador Viterbi emplea el Algoritmo de Viterbi para reconstruir la secuencia de señal transmitida más probable, reduciendo así la Tasa de Error de Bit (BER).

El proyecto consta de dos fases principales:

1) Implementación en MATLAB: Se realizaron simulaciones iniciales del Ecualizador Viterbi en MATLAB para evaluar su rendimiento, sensibilidad y requisitos de profundidad de retroceso. Se determinó que una profundidad de retroceso de 4 optimiza el equilibrio entre la ganancia de rendimiento y la eficiencia computacional.
2) Implementación en Hardware: El ecualizador se diseñó en SystemVerilog y se implementó en FPGA para validar su desempeño en condiciones reales. El Algoritmo de Viterbi con *Hard Output* logró una ganancia de 3.57 dB en sensibilidad en comparación con el demodulador GFSK estándar. Además, se implementó el Algoritmo de Viterbi con *Soft Output* (SOVA), obteniendo una ganancia de 0.61 dB en *frames* codificados, mejorando así la precisión y rendimiento.

La síntesis final en ASIC mostró una reducción del 46.17% en el área en comparación con un ecualizador original, lo que hace que el diseño sea significativamente más eficiente en consumo de energía y escalable. Estas mejoras posicionan al Ecualizador Viterbi como una solución robusta y eficiente en recursos para la demodulación BTC en el chip *Everest* de Silicon Labs, con potencial aplicabilidad en otros sistemas de comunicación inalámbrica basados en 2GFSK.

Para optimizar aún más el diseño, se aplicaron técnicas de cuantización con el objetivo de minimizar el consumo de recursos sin comprometer el rendimiento. Se seleccionó una resolución de 8 bits para las señales de entrada como el equilibrio óptimo, logrando una reducción adicional del 25% en el área de hardware y manteniendo al mismo tiempo una alta eficiencia en la demodulación.

Este proyecto demuestra un enfoque integral, combinando procesamiento de señales, optimización de algoritmos y diseño de hardware, lo que contribuye a mejorar la conectividad inalámbrica para futuras aplicaciones de IoT.

Palabras clave: Ecualizador Viterbi, Bluetooth Classic, GFSK, Intersymbol Interference, FPGA, ASIC, IoT, Algoritmo de Viterbi, Procesamiento de Señales.

.

ABSTRACT

This thesis presents the development of a Viterbi Equalizer for Bluetooth Classic (BTC) within the 'Everest' chip, developed by Silicon Labs for IoT applications. The project's approach involves theoretical analysis, algorithm implementation, hardware design, and optimization.

The work follows a structured approach, beginning with a theoretical review of Bluetooth Classic, GFSK modulation and demodulation, and the Viterbi Algorithm. The necessity of an equalizer arises from the presence of Intersymbol Interference (ISI), which degrades the accuracy of standard GFSK demodulators under high-noise conditions. The Viterbi Equalizer leverages the dynamic programming-based Viterbi Algorithm to reconstruct the most likely transmitted signal sequence, thereby reducing BER.

The project consists of two primary phases:

1) MATLAB Implementation: Initial simulations of the Viterbi Equalizer were performed in MATLAB to evaluate its performance, sensitivity, and traceback depth requirements. A traceback depth of 4 was found to optimize the trade-off between performance gain and computational efficiency.
2) Hardware Implementation: The equalizer was designed in System Verilog and implemented in FPGA to validate real-world performance. The Hard Output Viterbi Algorithm achieved a 3.57 dB gain in sensitivity performance compared to the standard GFSK demodulator. Additionally, the Soft Output Viterbi Algorithm (SOVA) was implemented, providing a gain of 0.61 dB in coded frames, ensuring better decoding accuracy.

The final ASIC synthesis showed a 46.17% reduction in area compared to an original equalizer, making the design significantly more power-efficient and scalable. These improvements position the Viterbi Equalizer as a robust, resource-efficient solution for BTC demodulation in Silicon Labs' Everest chip, with potential applicability to other 2GFSK-based wireless communication systems.

To optimize the design further, quantization techniques were applied to minimize resource consumption without compromising performance. An 8-bit resolution for the input signals was selected as the optimal trade-off, achieving an extra 25% reduction in hardware area while maintaining high demodulation efficiency.

This project showcases an integrated approach combining signal processing, algorithm optimization, and hardware design, contributing to enhanced wireless connectivity for next-generation IoT applications.

Key words: Viterbi Equalizer, Bluetooth Classic, GFSK, Intersymbol Interference, FPGA, ASIC, IoT, Soft Output Viterbi Algorithm, Signal Processing.

TABLE OF CONTENTS

LIST OF TABLES

ÍNDICE DE FIGURAS

INTRODUCTION

Company presentation

Silicon Labs is a leading global provider of silicon, software, and solutions for a smarter, more connected world. Founded in 1996 and headquartered in Austin, Texas, Silicon Labs specializes in the design and development of integrated circuits (ICs), microcontrollers, and wireless connectivity solutions that drive innovation in the Internet of Things (IoT) market (Silicon Labs, n.d.).

The company introduced its first product in 1998, a data access arrangement (DAA) chip used as an interface to public telephone lines. Thanks to its small size and low cost, this product achieved significant commercial success. The following year, Silicon Labs launched the first CMOS RF synthesizer, marking the beginning of their venture into wireless communication technologies (Business Wire, 2023). Over the years, Silicon Labs expanded its portfolio to include chips for a wide variety of wireless applications, such as digital TV demodulators, Wi-Fi, and AM/FM radio.

Since 2012, Silicon Labs has strategically shifted its focus towards IoT products, which now constitute a substantial portion of the company's revenue (Business Wire, 2023). This emphasis on IoT has positioned Silicon Labs as a key player in the rapidly growing connected devices market, providing cutting-edge solutions that enable seamless wireless connectivity across multiple platforms and devices.

In 2006, Silicon Labs acquired the Rennes-France office, which is the oldest European branch of the company (Silicon Labs, n.d.). Prior to the acquisition, the Rennes site operated as a startup specializing in the design of digital TV chips. The Rennes office has since become an integral part of Silicon Labs’ global operations, particularly in the development of advanced wireless communication technologies.

The Rennes site is organized into four main teams: the design team (which I joined), a software team responsible for developing stacks to support various network protocols on the chips, an application team that liaises with customers, and a newly established RF team focused on the analog aspects of integrated circuits. The design team, composed of 10 members, specializes in digital communication, with most team members focusing on hardware implementation and others dedicated to signal processing studies using MATLAB simulations.

My internship was positioned at the intersection of these two aspects—signal processing and hardware implementation. The opportunity to work on both the theoretical and mathematical aspects of signal processing, as well as the practical implementation and hardware design, is closely aligned with my academic and professional interests.

Internship context

The context of my internship lies in the development of a new chip for IoT applications at Silicon Labs, named 'Everest.' One of the wireless communication standards that the company intends to integrate into this chip is Bluetooth Classic (BTC).

In this regard, a Bluetooth Classic module was incorporated, which was a soft IP from a previous chip. However, this module relied on a GFSK demodulator. While a standard GFSK demodulator is sufficient for specific scenarios with low noise levels, it falls short when a more robust demodulation solution is needed to provide greater noise resilience and improved performance in terms of bit error rate (BER). In such cases, more reliable and robust options, such as a Viterbi Equalizer, should be considered.

Indeed, a previous version of a Viterbi Equalizer was used within the BTC module, but it did not function properly. Therefore, it became necessary to investigate the possibility of either fixing the existing equalizer or designing a new one from scratch to ensure the proper

functioning of the BTC module within the 'Everest' chip. To achieve this, it was essential to cover topics related to digital communications and signal demodulation, particularly GFSK, as well as the theory behind the Viterbi Algorithm.

Internship Objective

The primary objective of this internship is to develop a Viterbi Equalizer to be integrated with the Bluetooth Classic (BTC) feature of the 'Everest' chip. This involves designing and implementing an equalizer that can enhance the performance of the BTC module, particularly in challenging environments with higher levels of noise. The equalizer should improve the bit error rate (BER) and ensure reliable data transmission, thereby contributing to the overall robustness and functionality of the 'Everest' chip in IoT applications.

Work Plan

The following outlines the workflow and work plan designed to achieve the objectives of the internship:

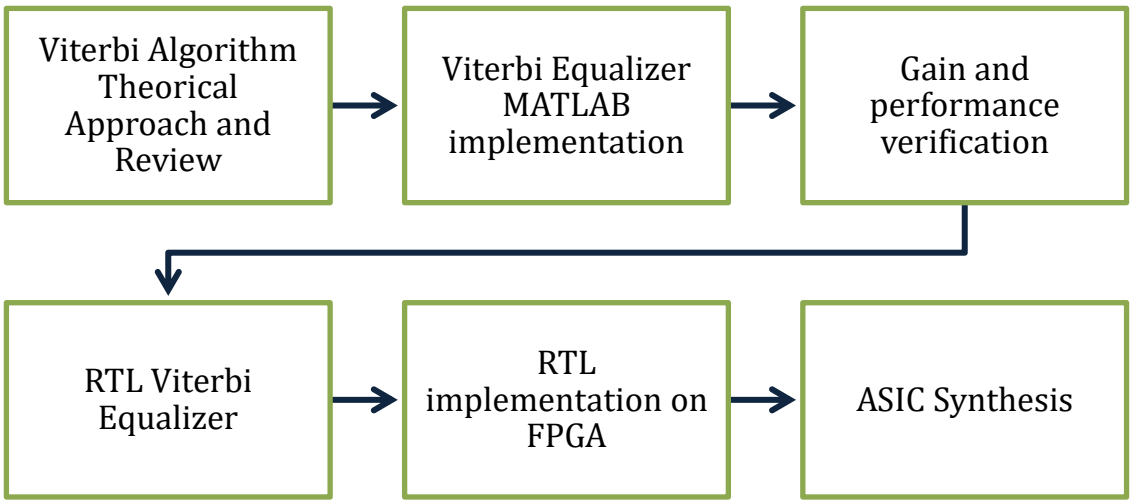

Figure 1: Work Plan Flow

THEORETICAL FRAMEWORK

Bluetooth Classic

Bluetooth Classic (BTC) is a foundational wireless communication standard that has been widely adopted since its inception for short-range data exchange between devices. It is part of the broader Bluetooth technology family, which also includes Bluetooth Low Energy (BLE) (Bluetooth SIG, 2016). While BLE is optimized for low-power, low-data-rate applications, Bluetooth Classic is particularly suited for scenarios where higher data rates and continuous connectivity are required, such as audio streaming, file transfers, and device synchronization (Patel & Mehta, 2020).

One of the primary reasons Bluetooth Classic remains relevant, especially in the context of IoT, is its robust and mature protocol stack, which provides reliable and secure communication across a wide range of devices (Duc, 2004). IoT applications often involve the integration of various devices—such as sensors, smartphones, and computers—into a cohesive network that requires seamless data transmission. Bluetooth Classic's ability to maintain stable connections and handle moderate to high data rates makes it an ideal choice for IoT systems where consistent performance is critical (Patel & Mehta, 2020).

In the context of the 'Everest' chip, the integration of Bluetooth Classic is crucial for ensuring compatibility with a wide range of existing devices and use cases. Given the widespread adoption of Bluetooth Classic, particularly in consumer electronics, its inclusion in the Everest chip allows for interoperability with legacy systems and provides a solid foundation for new IoT innovations.

To ensure the proper functioning of Bluetooth Classic within the Everest chip, several key technical parameters must be carefully considered during the design phase. These

parameters directly influence the performance, reliability, and efficiency of the communication module. The most relevant parameters for the design include:

- **Modulation**: Bluetooth Classic employs *Gaussian Frequency Shift Keying* (GFSK) as its modulation scheme. GFSK is known for its efficiency in minimizing bandwidth usage while maintaining signal integrity, making it well-suited for the typical operating conditions of Bluetooth devices (Duc, 2004).
- **Bandwidth-bit period product (BT)**: The product of the bandwidth and the bit period is set to 0.5. This parameter is critical in determining the spectral efficiency and the overall performance of the GFSK modulation. A BT product of 0.5 is a standard choice that balances the trade-off between signal robustness and bandwidth efficiency (Bluetooth SIG, 2016).
- **Data Rate (R)**: Bluetooth Classic operates at a data rate of 1 Mbps. This rate is sufficient for the majority of IoT applications that require moderate data transfer, such as sensor data collection, command, and control functions, and streaming low-bandwidth audio (Duc, 2004).
- **Modulation Index (h)**: The modulation index, which defines the extent of frequency deviation in GFSK, must be maintained between 0.28 and 0.35. This range ensures optimal performance in terms of signal clarity and error minimization under various operating conditions (Duc, 2004).
- **Bit Error Rate (BER)**: For reliable communication, the Bit Error Rate must be kept at or below 0.1% across an input power range from -70 dBm to -10 dBm. It is important to note that 0.1% BER represents the expected performance at the limit of the specified sensitivity by the standard. In practice, our limiting sensitivity point is much lower than this, which allows for improved

performance under real-world conditions, especially in scenarios with higher noise interference. Maintaining a low BER is essential for minimizing data transmission errors, which is particularly important in IoT devices where data integrity is critical for proper functioning (Duc, 2004).

GFSK

Gaussian Frequency Shift Keying (GFSK) is a digital modulation scheme used in various wireless communication systems, including Bluetooth Classic. GFSK is a variant of Frequency Shift Keying (FSK), where the frequency of the carrier signal is varied according to the input data signal. What distinguishes GFSK from standard FSK is the application of a Gaussian filter to the input data before modulation, which smooths the transitions between frequency shifts. This filtering reduces the bandwidth of the signal and minimizes interference with adjacent channels, making GFSK particularly suitable for environments with limited spectral space, such as in Bluetooth communications (Hagenauer & Hoeher, 1989).

The key advantage of GFSK lies in its ability to achieve a balance between spectral efficiency and robustness against noise and interference. By using a Gaussian filter, GFSK reduces the out-of-band emissions, which helps in conserving bandwidth and allows for multiple devices to operate within the same frequency spectrum with minimal interference (Shlezinger et al., 2024). This makes GFSK an ideal choice for short-range wireless communication standards like Bluetooth Classic, where maintaining signal integrity over a limited bandwidth is crucial.

GFSK operates by shifting the frequency of the carrier signal to represent binary data. Typically, a '1' bit is represented by a positive frequency deviation, while a '0' bit is represented by a negative frequency deviation. The extent of this deviation is defined by the modulation

index (h), which plays a critical role in determining the trade-off between signal bandwidth and the error performance of the system.

To better understand GFSK, we can consider a mathematical approach to its modulation and demodulation processes:

Modulation Process

The basic mathematical expression for a GFSK-modulated signal is represented as:

$$s(t) = Ae^{j\theta(t)}$$

Where:

- s stands for the GFSK-modulated signal that is going to be sent
- A is the amplitude of the GFSK-modulated signal
- θ is the phase of the GFSK-modulated signal

We notice that $s(t)$ depends directly on its phase. Thus, to produce the GFSK-modulated signal, it is crucial to produce the phase which is in function of the instantaneous frequency of the wave which, at the same time, is shifted according to the filtered data signal.

The modulation process is demonstrated through the following five steps:

1) **Mapping Digital Bits to Symbols:** The input bits from the data signal are mapped to symbols, where a '1' bit corresponds to the symbol '1', and a '0' bit corresponds to the symbol '-1'. This binary mapping simplifies the subsequent mathematical operations in the modulation process.

$$\mathbf{1} \rightarrow \ \mathbf{1}$$

$$\mathbf{0} \rightarrow \mathbf{-1}$$

2) **Gaussian Filter:** The symbols generated from the binary data stream are passed through a Gaussian filter, which smooths the data transitions. This smoothing reduces abrupt changes in the instantaneous frequency, thereby controlling the

bandwidth of the modulated signal and minimizing spectral spreading. The resulting signal is referred to as $m(t)$.

3) **Instantaneous Frequency:** The instantaneous frequency of the modulated signal is determined by the scaled version of the filtered data signal $m(t)$. For simplicity, we assume a baseband signal, setting the carrier frequency f_c to zero. Therefore, the instantaneous frequency is given by:

$$f(t) = f_c + f_d\, m(t)$$

$$f(t) = f_d\, m(t)$$

In this context, the chosen deviation frequency is 160 kHz corresponding to a modulation index (h) of 0.32.

4) **Getting the phase:** In physics, the instantaneous frequency and the phase of a signal are related by the following differential equation:

$$f(t) = \frac{1}{2\pi}\frac{d}{dt}\theta(t)$$

By integrating the instantaneous frequency over time, we obtain the phase of the signal as:

$$\Rightarrow \theta(t) = 2\pi\, f_d \int_{-\infty}^{t} m(\tau)\, d\tau$$

Note: For discrete-time implementation, especially in simulations using MATLAB, this integral is approximated by a summation, yielding (Proakis & Salehi, 2008):

$$\theta[n] = 2\pi\, f_d \sum_{k=1}^{n} m[k]$$

5) **Producing the GFSK-modulated signal:** Finally, the calculated phase is used in the complex exponential function to generate the GFSK-modulated signal. The complex signal is then transmitted through the communication channel.

The whole modulation process given by the previous steps are visually represented in detail in the accompanying figure:

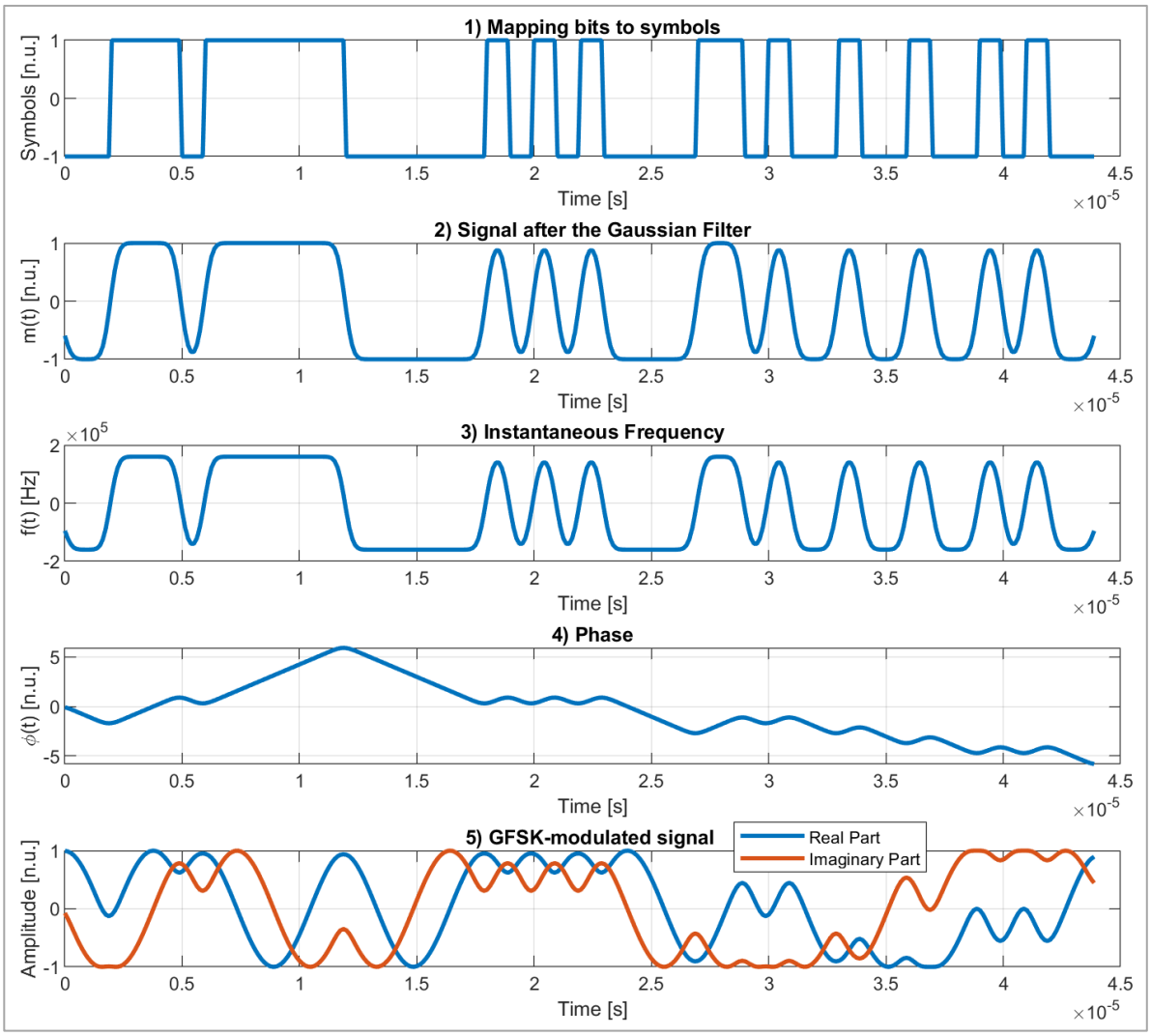

Figure 2: GFSK Modulation Process

Demodulation Process

The demodulation process, which is the focus of our work, mirrors the modulation process but in reverse order. It is no surprise that the steps closely resemble those of modulation, albeit carried out in the opposite direction.

First, the GFSK-modulated signal, now with added noise, is received. This signal is passed through a filter designed to mitigate as much noise as possible, aiming to restore the signal to a cleaner state. Following this, the phase of the signal is extracted, and we differentiate it, taking advantage of the relationship between the derivative of the phase and the instantaneous frequency.

From the resulting phase difference signal, we proceed to apply a standard GFSK demodulator. This demodulator operates using a hard decision method, where if the received value is greater than zero, the bit is determined to be '1', and if the value is less than zero, the bit is determined to be '0'.

Below is a visual representation of the demodulation process, which is essentially the reverse of the modulation process previously discussed.

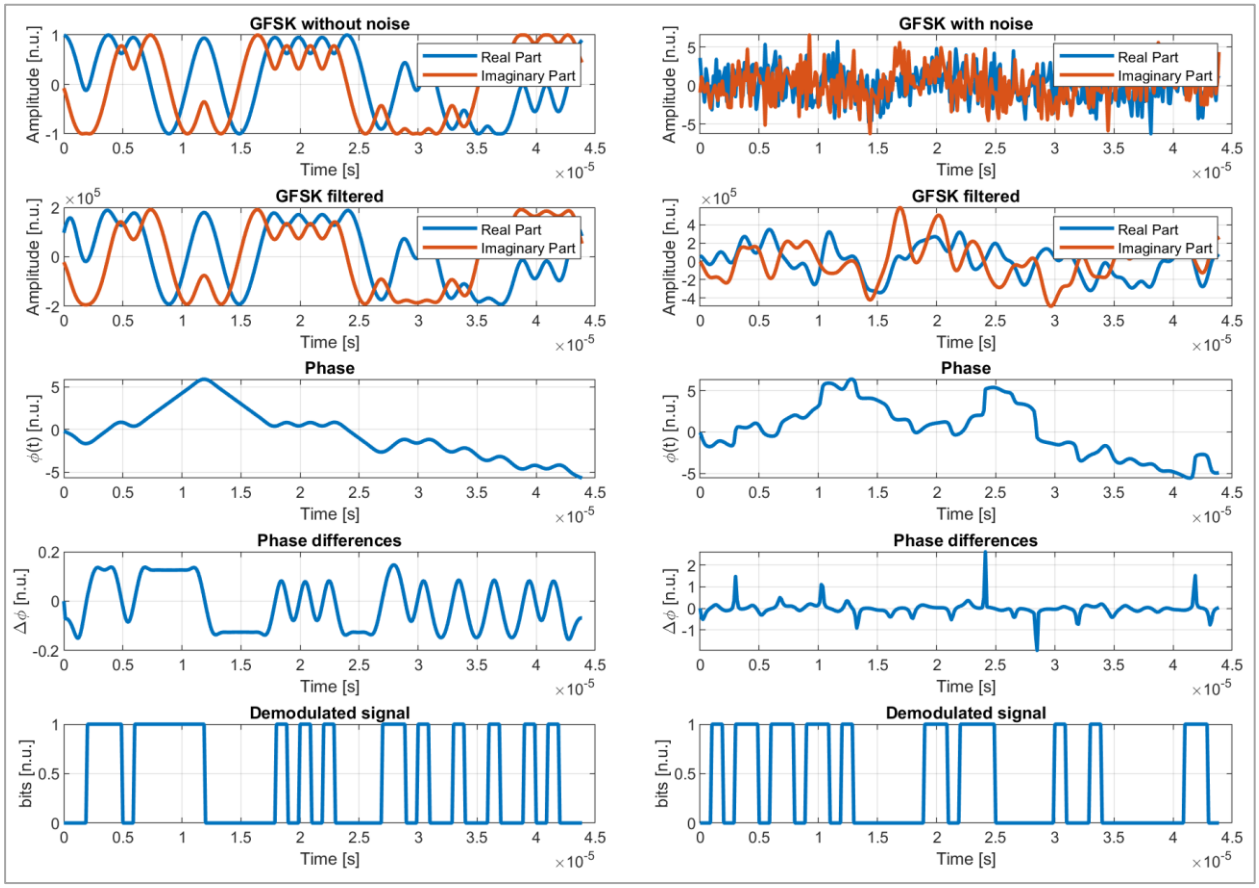

Figure 3: GFSK Demodulation Process with and without noise

In Figure 3, the demodulation process is contrasted for the same received signal under low and high noise conditions, respectively. The left-hand side of the figure demonstrates that when the noise level is low or negligible, the demodulation of the signal based on phase differences is straightforward, and the symbols are well-defined. In this case, using a standard GFSK demodulator with a simple threshold is sufficient.

However, when the received signal has a high noise level, even after processing and filtering, the resulting phase differences do not provide high reliability for using a standard GFSK demodulator. Values close to zero could correspond to either a '0' bit or a '1' bit, making it difficult to determine the correct bit with certainty. This lack of reliability is why a standard GFSK demodulator is inadequate, and a Viterbi equalizer is often preferred, as it accounts for the effects of Intersymbol Interference (ISI).

In Figures 2 and 3, it is noticeable that in the subplots corresponding to instantaneous frequency or phase differences, the values attempt to reach well-defined maximum or minimum levels. These levels depend not only on the received symbol but also on the neighboring symbols, due to the influence of ISI (Shlezinger et al., 2024). This phenomenon arises from the Gaussian filter used during modulation, which smooths transitions between symbols but also introduces intersymbol dependency.

The Viterbi equalizer is designed to address this by considering the impact of ISI, thereby improving the reliability of bit decisions, especially in noisy environments. The equalizer uses a probabilistic approach to estimate the most likely sequence of transmitted symbols, taking into account both the current and previous symbols.

Viterbi Algorithm

The Viterbi Algorithm is a widely used method in digital communications for decoding sequences of symbols that have been transmitted through a noisy channel. It was originally developed by Andrew Viterbi in 1967 as a technique for decoding convolutional codes, which are a type of error-correcting code used to improve the reliability of data transmission (Viterbi, 2010).

In essence, the Viterbi Algorithm is a dynamic programming algorithm that finds the most likely sequence of hidden states (or symbols) that could have generated a sequence of observed events, considering the impact of noise and interference during transmission (Viterbi, 2010). It achieves this by exploring all possible paths through a trellis diagram—a graphical representation of all possible states of the system—while keeping track of the path with the highest probability. This allows the algorithm to make decisions that minimize the probability of error, effectively mitigating the effects of noise and interference.

While our focus in this work is on the application of the Viterbi Algorithm in the context of GFSK demodulation, it is most easily understood through its original application: the decoding of convolutional codes. Convolutional codes are a type of forward error correction code that adds redundancy to the transmitted data by encoding it into a longer sequence using a finite-state machine. This encoding introduces a dependency between consecutive symbols, which can be exploited by the Viterbi Algorithm to correct errors introduced by the channel.

To better understand the essence of the Viterbi Algorithm, it is helpful to explore an example in the context of convolutional encoding and decoding. This example will demonstrate how the algorithm traces back through the trellis to find the most likely sequence of transmitted symbols, effectively correcting errors that may have occurred during transmission.

Let us consider the following ½ rate encoder, given by the generator polynomials $\boldsymbol{g_1 = (101)}$ and $\boldsymbol{g_2 = (111)}$:

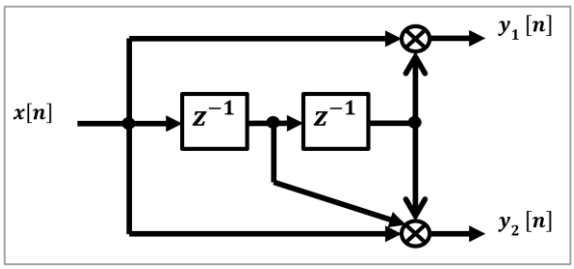

Figure 4: Encoder example

The encoder of Figure 4 also can be represented by the following output sequence expressions:

$$\boldsymbol{y_1[n] = x[n] + x[n-2]}$$

$$\boldsymbol{y_2[n] = x[n] + x[n-1] + x[n-2]}$$

In fact, the encoder under study cannot be fully represented by the previous expressions or the diagram in Figure 4 alone. However, it can be effectively represented as a state machine to observe the state change behavior. A more comprehensive representation can be achieved using a Trellis diagram, which allows for the analysis of state changes over time by incorporating the time dimension. This enables the visualization of how the states evolve over time. Both representations are illustrated in Figure 5 for the same encoder:

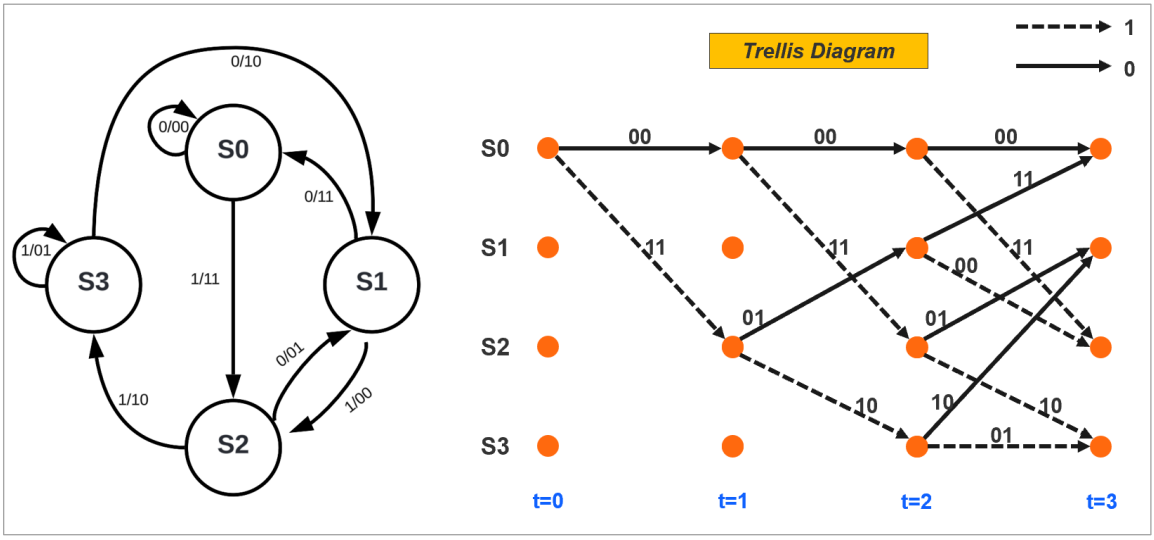

Figure 5: Encoder representation – State Machine and Trellis Diagram

To better understand the Viterbi Algorithm, let us walk through a practical example. Suppose the message we want to transmit is "**1100**". If we input this message into our convolutional encoder, as illustrated in Figure 4, the encoded message would be "**11101011**," which is now ready for transmission. However, to demonstrate the decoding and error correction capabilities of the Viterbi Algorithm, let us assume that the received message was "**10101111**". This received sequence contains two errors when compared to the originally transmitted encoded message.

The Viterbi Algorithm can be broken down into three well-defined steps:

1) Building a Trellis Diagram:

The first step involves constructing a trellis diagram, which represents all possible states of the encoder at each time step (Viterbi, 2010). Each path through the trellis corresponds to a possible sequence of transmitted bits.

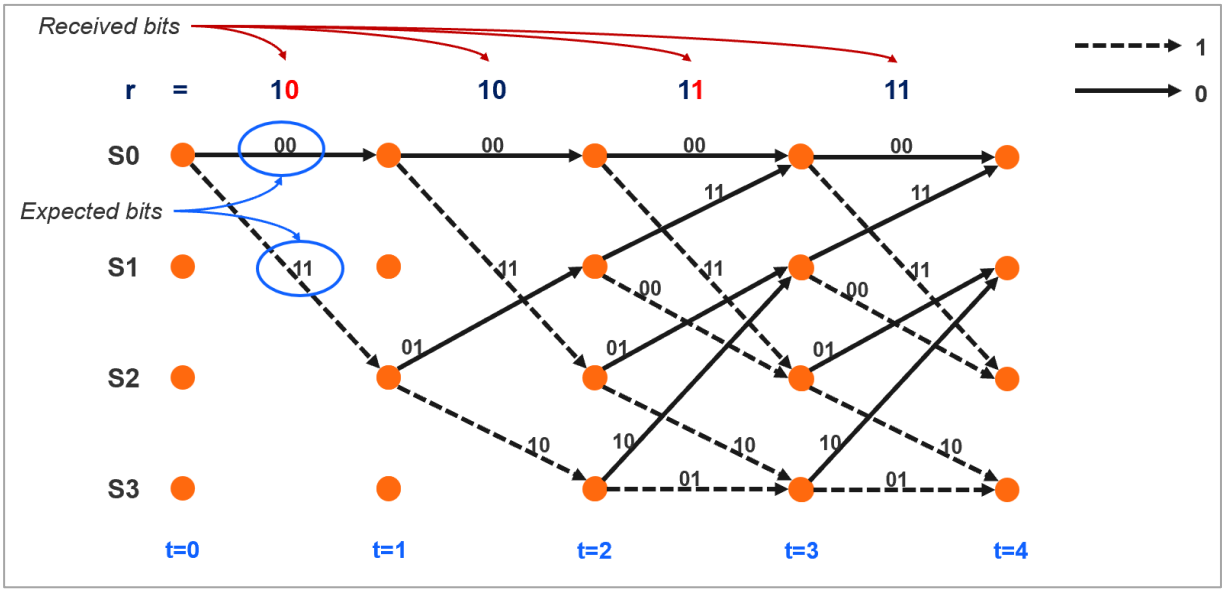

Figure 6: Viterbi Algorithm – Building Trellis Diagram

2) Computing and Updating Metrics to Store Survivor Paths:

As we move forward through the trellis, we compute branch metrics for each possible path, typically by calculating the difference between the received signal and the expected signal for each transition. This can be done using a distance measure like the Hamming distance or absolute value difference (Viterbi, 2010). The survivor paths are those with the lowest cumulative metric, indicating the most likely sequences that could have resulted in the received message. For each node, the path with the highest cumulative metric is discarded, leaving only the path with the lowest metric, referred to as the decision metrics.

The process continues by accumulating metrics at each node, where each node's accumulated metric is the sum of the previous non-discarded branch metric and the corresponding node metric from earlier in the trellis. These cumulative metrics are propagated forward, ensuring that only the most likely paths are retained. Once the forward traversal of the trellis is complete, a traceback process is performed to identify the most likely sequence of transmitted bits.

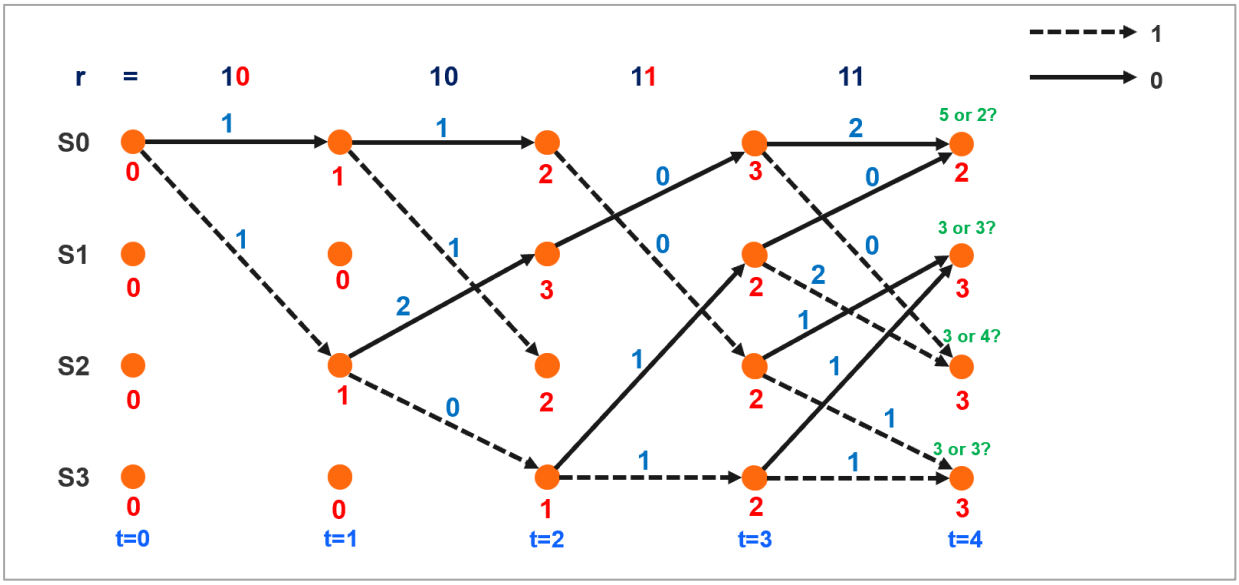

Figure 7: Viterbi Algorithm – Computing metrics and storing survivor paths

In Figure 7, the accumulated metrics, branch metrics, and decision metrics are illustrated for each state (S0, S1, S2, and S3) over time t. The red numbers represent the accumulated metrics, the blue numbers represent the branch metrics, and the black arrows indicate the selected survivor paths for each state at each time step.

3) Tracing Back:

After reaching the end of the trellis, we trace back through the survivor paths to determine the most likely sequence of transmitted bits. This process involves moving backward through the trellis to identify the path with the smallest cumulative metric, which represents the corrected version of the received message.

The deeper the traceback is, the higher the confidence in the decoding process. This is because, as we trace further back through the trellis, the survivor paths from different states will eventually converge into a single path. This phenomenon occurs because, no matter which

path the traceback begins from (typically the one with the lowest metric), all possible paths will converge after a certain number of steps.

As a result, decoded bits located toward the left (i.e., those from the end of the traceback process) have a higher reliability of being correctly decoded as '0' or '1', as these bits represent the positions where all paths have already converged. The reliability of the decoded bits decreases as we move to the right (closer to the start of the traceback), where paths may not have converged yet.

Due to this, traceback depth becomes a critical parameter: a deeper traceback ensures more bits are decoded after paths have converged, increasing the overall accuracy. To ensure convergence and maximize decoding reliability, another common technique is to add tailing bits (usually zeros) at the end of the transmitted sequence. These tailing bits force the trellis into a known final state (often state S0), ensuring that the traceback starts from the correct state at the end of the trellis. This guarantees that the traceback process is initiated correctly and improves the accuracy of the decoded message.

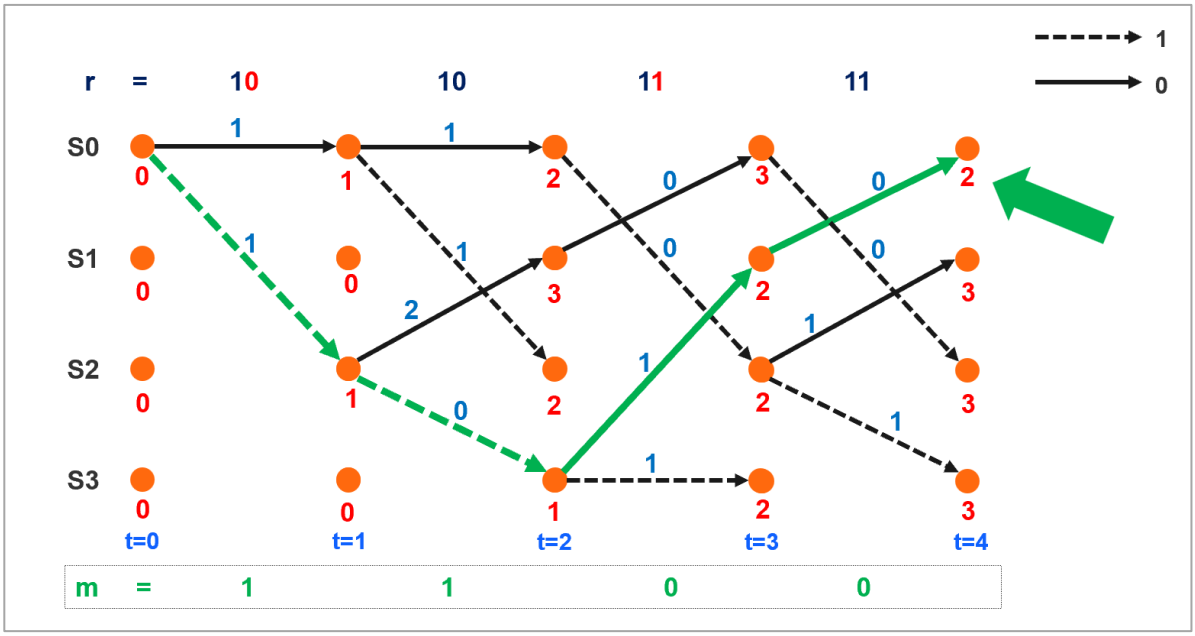

Figure 8: Viterbi Algorithm – Tracing back to find the original message

In this example we realize how the Viterbi Algorithm is capable of recovering the original message from the received bits although it contains errors. In simple terms, the Viterbi Algorithm consists on a process of moving forward through the trellis while building and updating the possible paths, and then moving backward to make a decision about which path most likely represents the original transmitted message. It can be also thought of as traveling forward into the future to gather information, and then coming back to the past to make a decision.

Viterbi Equalizer

The term "Equalizer" in the context of a Viterbi Equalizer should not be misunderstood as it does not refer to the traditional concept of equalization, such as frequency equalization, which is commonly known in audio processing. Instead, a Viterbi Equalizer is essentially a demodulator, and it could just as accurately be called a "Viterbi Demodulator". The primary

role of a Viterbi Equalizer is to make optimal decisions about the transmitted symbols based on the received signal.

Once the Viterbi Algorithm is understood in the context of convolutional decoding, it becomes straightforward to adapt it for the purpose of demodulation, which is the context that interests us here. In traditional convolutional decoding, the algorithm works by comparing the received sequence with expected sequences of bits or symbols. In the case of demodulation, particularly for GFSK signals, the algorithm instead works with expected phase differences as it is shown in Figure 9.

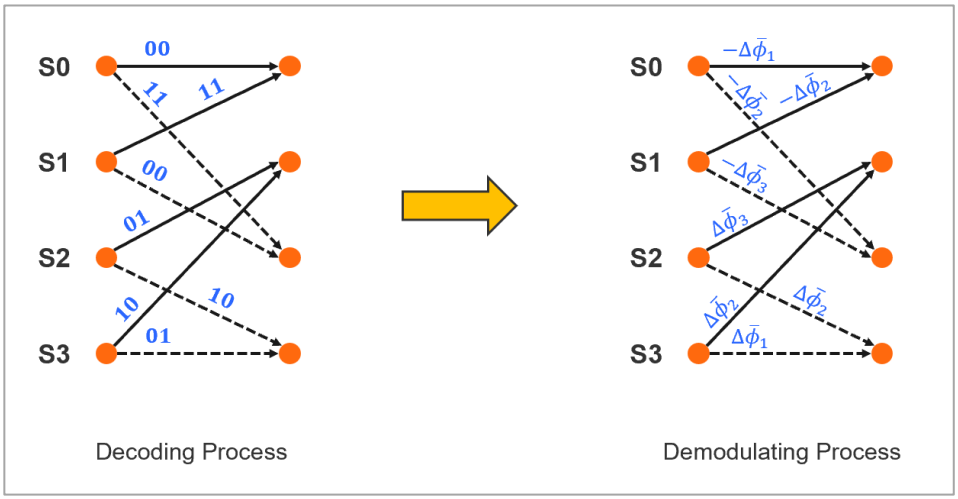

Figure 9: Extended use of the Viterbi Algorithm

Due to intersymbol interference (ISI), the sequences of received phase differences do not occur independently but are influenced by neighboring symbols. This dependency can be modeled using a state machine or trellis diagram, similar to the trellis diagram used in convolutional decoding. Each state in the machine represents a possible combination of past and future symbols that could influence the current phase difference.

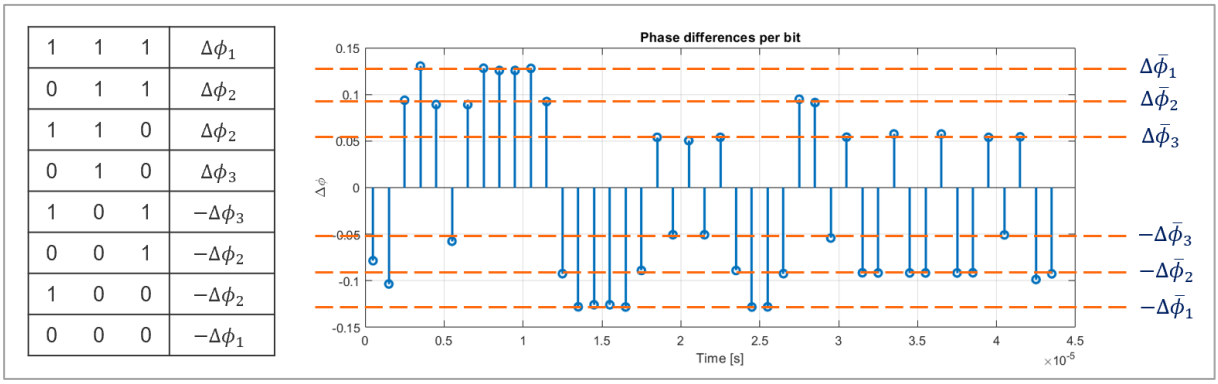

Figure 10: Phase differences levels and neighboring symbol dependency

The Viterbi Equalizer, therefore, uses the same principles as the Viterbi Algorithm in decoding: it constructs a trellis based on possible phase differences, computes metrics for each path, and traces back to find the most likely sequence of transmitted symbols, considering the influence of ISI. In this way, the Viterbi Equalizer effectively acts as a GFSK demodulator that optimally handles the challenges posed by ISI, providing a robust solution for reliable communication in noisy environments.

While a Viterbi Equalizer typically focuses on producing a "hard output," where the demodulated bits are decisively identified as either '0' or '1,' it can also be modified to produce "soft output."

SOVA

The Soft Output Viterbi Algorithm (SOVA) is an extension of the traditional Viterbi Algorithm that provides more nuanced information about the decoded bits or symbols (Berrou et al., 1993). While the standard Viterbi Algorithm yields a "hard decision" output, where each decoded bit is definitively classified as either a '0' or a '1', SOVA produces a "soft output," which includes additional information about the likelihood or confidence of each decision.

The importance of SOVA lies in its ability to enhance the performance of communication systems, particularly when combined with iterative decoding techniques like block coding. By providing a measure of uncertainty for each bit, SOVA allows for more effective error correction. This measure of uncertainty is often expressed in the form of Log-Likelihood Ratios (LLR), which quantify how much more likely it is that a given bit is a '0' versus a '1' (Berrou et al., 1993).

The Log-Likelihood Ratio (LLR) is a crucial component in the SOVA process. It provides a probabilistic measure of each bit's value, calculated as:

$$LLR = \ln\left(\frac{P(bit = 1 \mid received\ signal)}{P(bit = 0 \mid received\ signal)}\right)$$

Where:

- $P(bit = 1 \mid received\ signal)$ is the probability that the bit is '1' given the received signal
- $P(bit = 0 \mid received\ signal)$ is the probability that the bit is '0' given the received signal

Which implies:

- If the LLR is a large positive number, it means the bit is very likely to be '1'
- If the LLR is a large negative number, it means the bit is very likely to be '0'
- If the LLR is close to zero, the bit could be either '1' or '0', and the algorithm is not very confident

SOVA modifies the conventional Viterbi Algorithm by incorporating the calculation of these LLRs during the decoding process. As the algorithm progresses through the trellis, it computes both the path metrics (as in the traditional Viterbi Algorithm) and the LLRs for each

bit. This additional information is then used to generate the soft output, providing the decoder with valuable insights that can be leveraged in subsequent stages of processing, such as in iterative decoding schemes.

The importance of SOVA and LLRs cannot be overstated in modern communication systems, where maximizing data reliability and minimizing error rates are paramount. By generating soft outputs, SOVA enables more robust error correction and ultimately improves the overall performance of the system.

Below, I propose a mathematical demonstration of how to calculate the soft output using LLRs. This will involve deriving the LLR expression based on the path metrics calculated during the Viterbi decoding process, illustrating how SOVA integrates these calculations into the decoding algorithm to produce soft decisions.

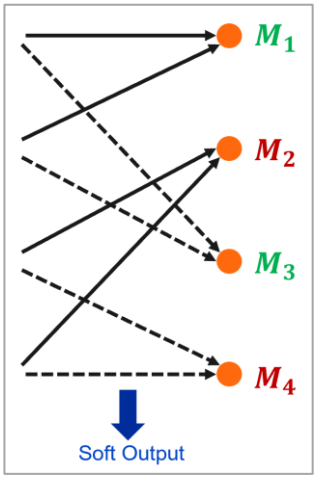

Figure 11: Metrics in Trellis

From Figure 11, we observe that metrics M_1 and M_3 originate from either state S0 or S1, which indicates a high likelihood that the soft output will correspond to a '0'. On the other hand, metrics M_2 and M_4 originate from either state S2 or S3 , indicating a high likelihood that the soft output will correspond to a '1'.

Given this, the probability of selecting a path that leads to a soft output representing a '1' can be expressed as the ratio of the metric associated with the '1' paths to the sum of all relevant metrics. Mathematically, this probability can be calculated as:

$$P(bit = 1 \mid received\ signal) = \frac{e^{-M_2} + e^{-M_4}}{e^{-M_1} + e^{-M_2} + e^{-M_3} + e^{-M_4}}$$

In the same way, the probability of selecting a path that leads to a soft output representing a '0' can be calculated as:

$$P(bit = 0 \mid received\ signal) = \frac{e^{-M_1} + e^{-M_3}}{e^{-M_1} + e^{-M_2} + e^{-M_3} + e^{-M_4}}$$

Note: The exponential function e^{-M_i} represents the likelihood that a given path i is the correct path due to the fact that $M_i \in [0, \infty[$. Thus, the smaller the metric, the more likely it is that the path is the correct one.

Therefore,

$$LLR = \ln\left(\frac{\frac{e^{-M_2} + e^{-M_4}}{e^{-M_1} + e^{-M_2} + e^{-M_3} + e^{-M_4}}}{\frac{e^{-M_1} + e^{-M_3}}{e^{-M_1} + e^{-M_2} + e^{-M_3} + e^{-M_4}}}\right)$$

$$LLR = \ln\left(\frac{e^{-M_2} + e^{-M_4}}{e^{-M_1} + e^{-M_3}}\right)$$

Considering the following approximation:

$$e^{-A} + e^{-B} \approx e^{-\min(A,B)}$$

Finally, the soft output LLR value is calculated as:

$$LLR = \ln\left(e^{\min(M_1,M_3)-\min(M_2,M_4)}\right)$$

$$\Rightarrow LLR = \min(M_1, M_3) - \min(M_2, M_4)$$

WORK PERFORMED

The work I carried out throughout this project was focused on applying the theoretical concepts discussed earlier to design a Viterbi Equalizer from scratch. In this process, I adopted a two-stage approach.

In the first stage, I used MATLAB to simulate the Viterbi Equalizer. Since processing resources are not a primary concern in this environment, I was able to run extensive simulations, adjusting various parameters and fine-tuning the details of the algorithm. This allowed me to thoroughly analyze the performance of the equalizer and identify the most optimal model based on different criteria such as error rates and computational efficiency.

Once the optimal parameters were determined through MATLAB simulations, I proceeded to the second stage, where the focus shifted to designing the Viterbi Equalizer at the hardware level. For this, I developed the model using RTL (Register Transfer Level) design, specifically utilizing System Verilog as the hardware description language (HDL). This phase of the project involved translating the algorithm into a hardware-implementable form, ensuring that the design was resource-efficient, particularly in terms of area, which translates to fewer gates used and, consequently, lower power consumption.

MATLAB Implementation

The first step of the project was to familiarize myself with the complete workflow of digital wireless communications, encompassing both the transmitter side (encoding and modulation of the message) and the receiver side (demodulation and decoding of the received signal). With this in mind, I began by studying a simpler modulation scheme, such as BPSK, and utilized MATLAB's predefined functions to encode and decode convolutional codes. After

developing a solid understanding of these concepts, I progressed to GFSK modulation and implemented the Viterbi algorithm from scratch.

The most complex part of the Viterbi algorithm is clearly the accumulation of path metrics in a feedback process, where decisions are made based on the best path metrics, as explained in Section (2.3.). One essential aspect to consider is that, for ***n*** received bits, the ideal Viterbi algorithm would need to move ***n*** steps forward to accumulate the metrics and then ***n*** steps backward to trace the best path. While this would not be an issue for a small ***n***, in reality, ***n*** is typically very large, and the resources required to process the entire trellis forward and backward would be extremely high.

However, by paying attention to the nature of the algorithm, we realize that it does not matter if we start the traceback from a correct or incorrect path; eventually, all paths will converge to the correct one. This means that, instead of tracing back the entire trellis at once, we can trace it in smaller segments, progressing through the trellis incrementally until the process is complete

Extensive simulations were conducted by me using 10 million random bits transmitted through the channel, and the bit error rate (BER) was calculated after the Viterbi algorithm was applied, both for the ideal case where the entire trellis is processed and for the optimized case where smaller segments are traced back and processed in sequence. A depth of 7 was chosen as a neutral value for the depth of the smaller segments to trace back. The results in both cases were nearly identical, demonstrating that the optimized approach is more resource-efficient while achieving the same performance as the ideal case.

Additionally, I ran simulations where, instead of transmitting 10 million bits consecutively, I transmitted 1000 blocks of 10 000 bits each, simulating the frames that will be used in the final hardware design. Again, the results were unchanged, confirming the robustness of the optimized approach.

Traceback Depth Analysis

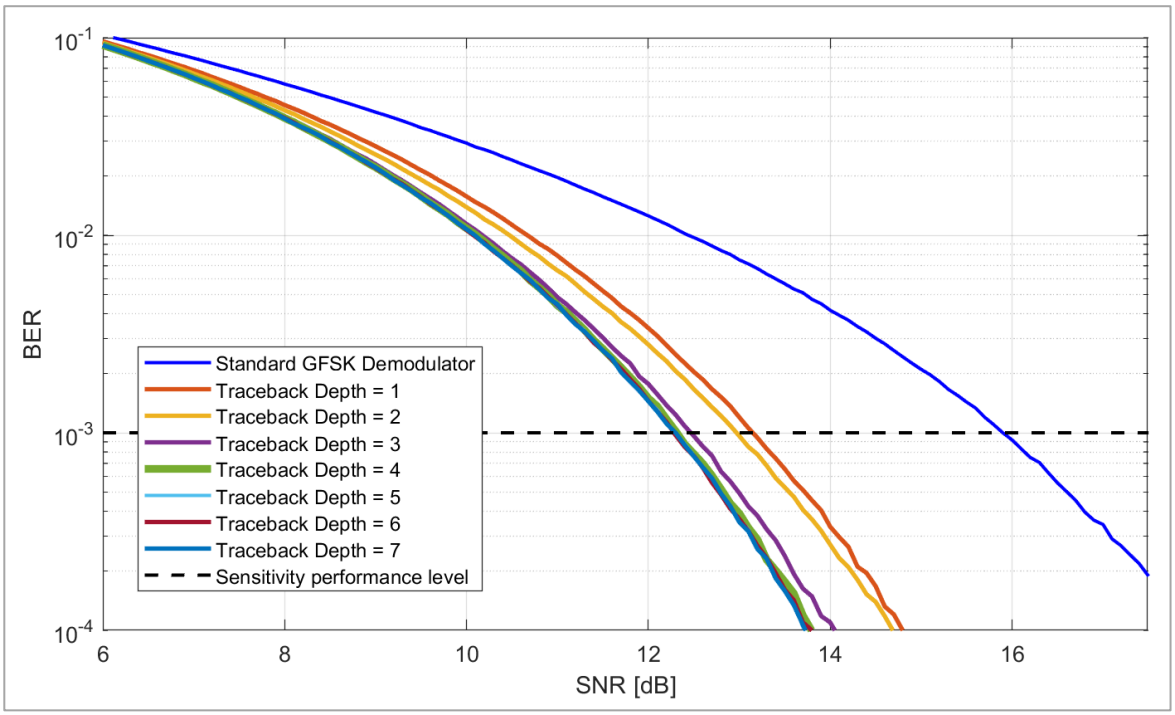

Figure 12: BER curves in function of the traceback depth

The concept of Traceback Depth refers to how many steps backward the Viterbi algorithm must go in the trellis to find the most likely sequence of transmitted bits. As mentioned in Section (3.1.), instead of traversing the entire trellis from start to finish, we can optimize the process by performing the traceback in smaller segments, reducing the computational burden without sacrificing performance.

While using a fixed window for traceback over the entire trellis is a solid strategy, there is still room for further optimization. Specifically, we can fine-tune the depth of the traceback window to balance between performance and resource efficiency. To explore this, I conducted simulations with varying traceback depths, the results of which are shown in Figure 12.

In Figure 12, we observe the Bit Error Rate (BER) results for different traceback depths. Additionally, the figure includes the BER results using a standard GFSK demodulator for comparison. This allowed me to calculate the performance gain for each of the Viterbi Equalizer models with different traceback depths, relative to the performance of the original GFSK demodulator, at the sensitivity performance level of 0.001, which is our limit target as explained in Section (2.1.).

The table below summarizes the performance gain for each model:

Table 1: Viterbi Equalizer gain relative to Standard GFSK Demodulator

Traceback Depth	**GAIN [dB]**
1	2.75
2	2.93
3	3.43
4	3.59
5	3.60
6	3.61
7	3.62

Based on these results, I proposed using a traceback depth of 4, as this value maximizes the performance gain. Choosing a higher traceback depth would lead to diminishing returns in terms of performance improvement, making the additional resources required unjustifiable.

Conversely, selecting a traceback depth lower than 4 would significantly reduce the gain achieved by the design.

HDL implementation and validation

Based on the analyses conducted so far using the MATLAB approach, the focus now shifts to implementing the Viterbi Equalizer at the hardware level. For this purpose, as previously mentioned, I used *System Verilog* as the hardware description language (HDL). The design of the proposed Viterbi Equalizer block is shown below in Figure 13.

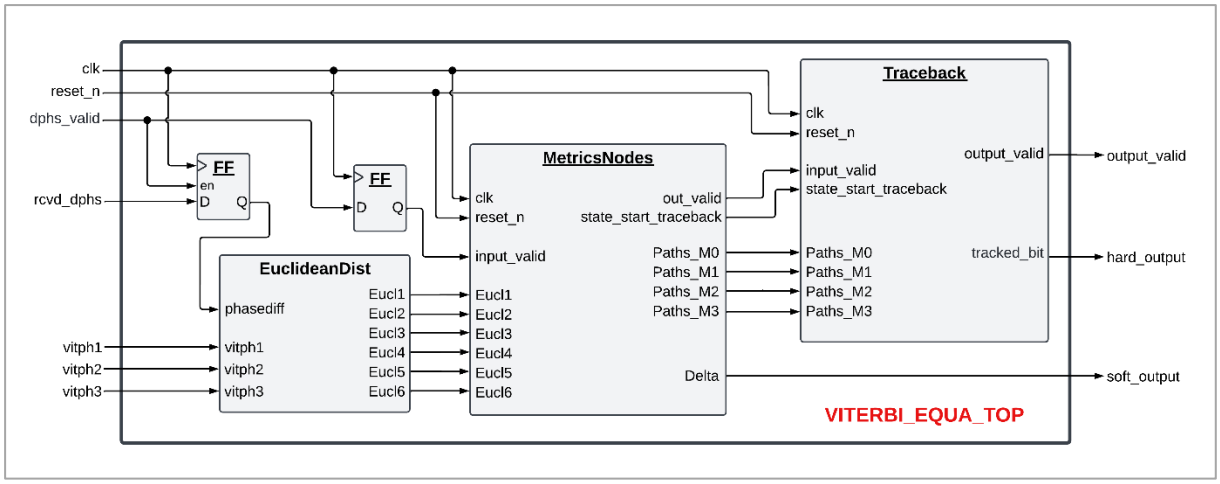

Figure 13: Top module diagram – Viterbi Equalizer

The design I propose consists of three main submodules:

1) **Submodule 'EuclideanDist'**:

   This block computes the branch metrics using the Euclidean distance as the criterion. Specifically, it calculates the absolute difference between the received phase difference, found in the signal 'phasediff,' and the six expected phase difference levels. It is important to note that the other input signals, 'vitph1,' 'vitph2,'

and 'vitph3,' are the static values of the three levels of the expected positive phase differences. Due to symmetry, these values also determine the three levels of the expected negative phase differences. The combinational logic performed by this block can be described as follows, with operations involving absolute value differences:

$$\text{Eucl1} = |-\phi_1 - \text{phasediff}|$$

$$\text{Eucl2} = |-\phi_2 - \text{phasediff}|$$

$$\text{Eucl3} = |-\phi_3 - \text{phasediff}|$$

$$\text{Eucl4} = |\ \phi_3 - \text{phasediff}|$$

$$\text{Eucl5} = |\ \phi_2 - \text{phasediff}|$$

$$\text{Eucl6} = |\ \phi_1 - \text{phasediff}|$$

These operations are used to compare the received and expected phase differences, computing the Euclidean distance for each branch.

2) Submodule 'MetricsNodes':

This is the most complex block, responsible for handling the metrics and making decisions. It consists of two internal subprocesses, visualized in Figure 14. The first subprocess involves accumulating the branch metrics across the trellis. The metrics of the respective branches converging at each node are added to the accumulated metrics of the corresponding previous nodes. Each node then retains the path with the lowest metric, discarding the others.

The four metrics corresponding to the four nodes of the trellis at time $\boldsymbol{t}$ are sent to another block that calculates the Log-Likelihood Ratio (LLR), as discussed in Section (2.3.2.), and identifies which node has the lowest metric. This information

is used in the next subprocess. Additionally, the value of this lowest metric is subtracted from all the node metrics, preventing metric overflow or saturation in subsequent cycles.

The second subprocess involves storing the paths with the lowest metrics in a register array based on the determined traceback depth. This stored path information is then fed back through the system as the traceback window advances through the trellis, ensuring that only the best paths are considered.

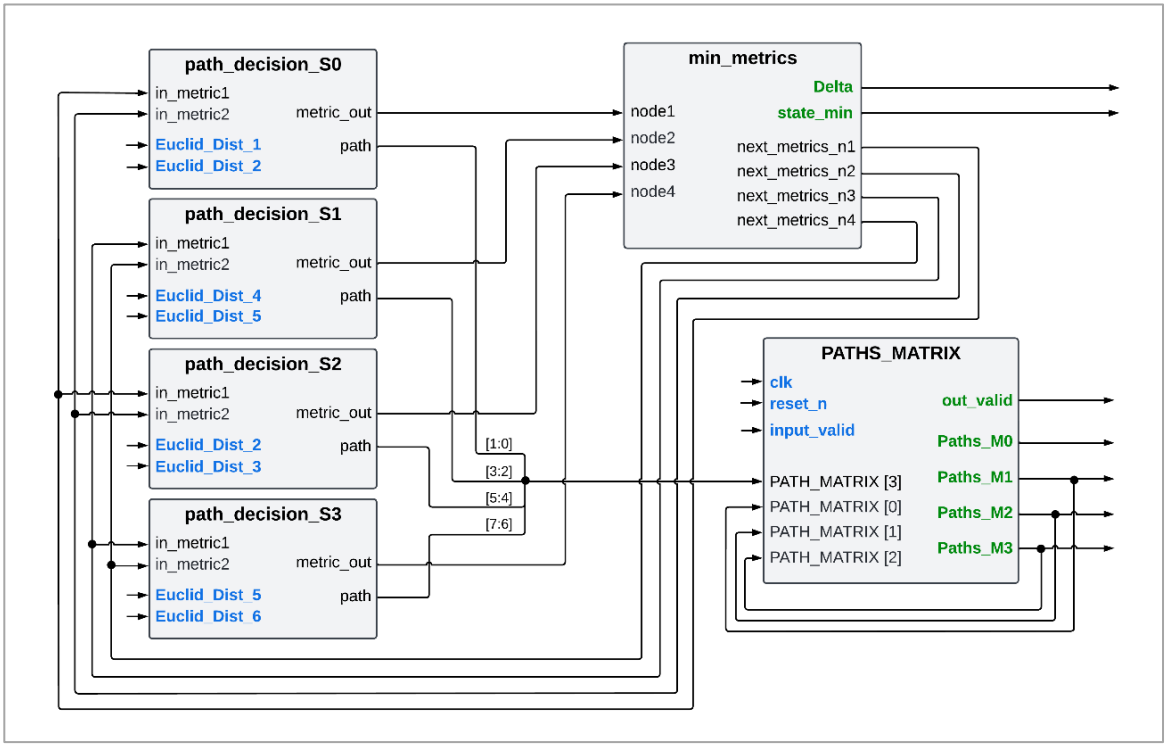

Figure 14: Submodule 'MetricsNodes' diagram

3) Submodule 'Traceback':

This block is compact, simple, and purely combinational. It uses four multiplexers to perform each traceback step with a depth of four. Additionally, a final multiplexer is used to decode whether the bit is a '0' or '1'. The internal diagram of this block is shown below in Figure 15.

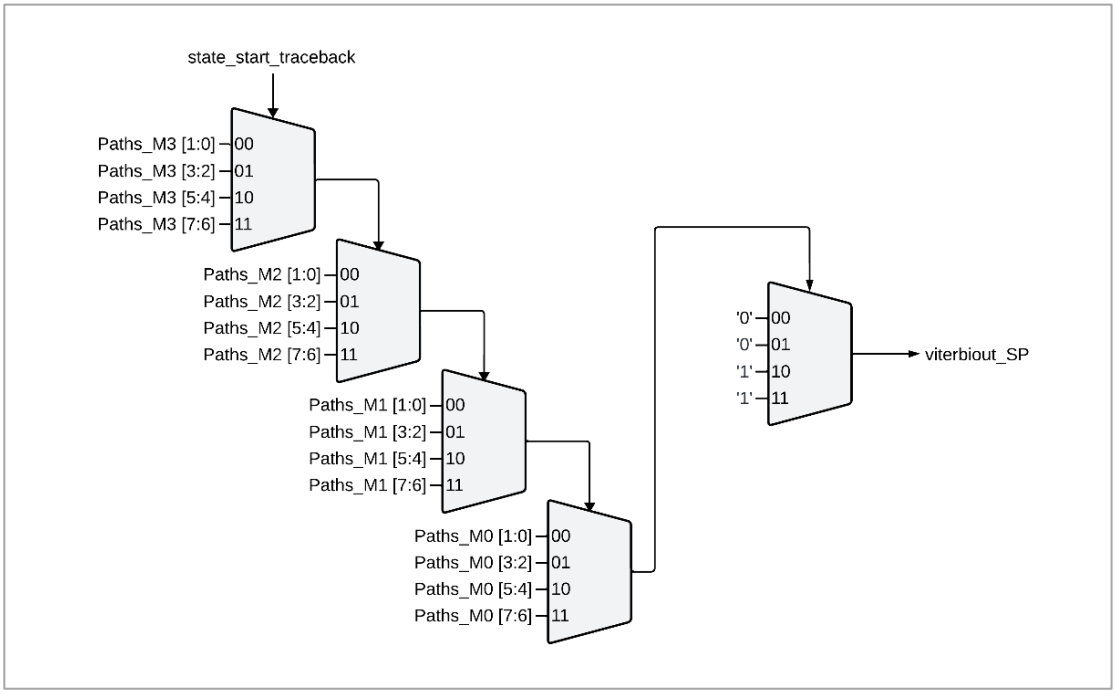

Figure 15: Submodule "Traceback" diagram

This modular design allows the Viterbi Equalizer to efficiently process the received signals while simplifying the algorithhm flow and avoiding metric overflow. After HDL model was done, I synthesized it and implemented the system on an FPGA to validate the hardware performance, ensuring that the design operates as expected under realistic conditions.

For **uncoded frames**, I used the standard mode of the Viterbi Equalizer with hard output, obtaining a 3.57 dB gain in sensitivity performance at the BER target, compared to the BER results of the original GFSK demodulator. On the other hand, for **coded frames**, I used the SOVA mode of the Viterbi Equalizer and achieved a BER of 0.001 at 11.61 dB of SNR. Both results from the hardware implementation align well with the MATLAB model, demonstrating consistency between the two approaches. For hard outputs, the expected gain from the MATLAB simulations was 3.59 dB, as discussed in Section (3.2.), which is very close to the observed gain of 3.57 dB in the FPGA implementation. Similarly, for soft outputs, the ideal SOVA function in the MATLAB model, which utilized full computational resources, indicated that achieving a BER of 0.001 required 11.59 dB of SNR. This result was also closely matched in the hardware implementation, with 11.61 dB needed to reach the same BER.

This consistency between the results of the implemented hardware model on FPGA and the results of the MATLAB model validates the accuracy and coherence of both Viterbi Equalizer designs.

Design Optimization

So far, the results obtained have been very promising, and there is consistency between the MATLAB model and the HDL model, meaning that the design is almost ready for deployment. However, I decided to take the design a step further by optimizing it, focusing on signal resolution as well as the parametrization and dependency of all internal signals based on the input signals.

One key strategy I used for optimization is signal quantization. Quantization refers to the process of reducing the resolution or bit length of signals, typically to minimize resource usage or to improve processing efficiency, while still maintaining acceptable performance (Duc, 2004). I chose this approach because by reducing the bit lengths of specific signals in the Viterbi Equalizer, I could significantly optimize resource usage without compromising the overall accuracy of the system.

To achieve this, I first needed to analyze the relationships between the signals in the design. By observing the diagram in Figure 13, we can categorize the signals that can either be quantized or have their bit length reduced into three distinct groups:

1) Signals at the input

2) Signals related to the Euclidean Distance

3) Signals within the ‘MetricsNodes’ block

The required bit lengths for the signals in Group (3) are directly dependent on the normalized bit length of the signals in Group (2). However, since the signals in Group (2) must also be quantized and depend only on simple combinational logic derived from the signals in Group (1), the best approach for optimization is to focus on quantizing the signals in Group (1), the input signals. After doing so, the bit lengths of the signals in Group 2 should be minimized to the required amount without causing performance degradation.

In this context, and considering the RTL implementation where the Viterbi Equalizer block will be used, the input signals have a bit length of **11 bits**. The critical question then became: how many Least Significant Bits (LSBs) can be removed from the input signals during

quantization to strike the right balance between reducing the number of bits and maintaining a bit error rate (BER) that meets performance expectations?

To address this, the best strategy was to ensure that the bit length of all internal signals is a function of the input signal length. Therefore, I proceeded to conduct an initial analysis in MATLAB, where I simulated the system using 5 million bits while varying the input signal lengths. Based on these results, I selected the optimal bit lengths for modifying the HDL model.

Afterward, I synthesized and implemented the optimized design on an FPGA to validate the hardware. This allowed me to confirm that the design maintained its performance even with the reduced bit lengths, leading to a more efficient implementation in terms of resource usage.

RESULTS

The results presented below are based on the necessary parameters to optimize my design for the specific context in which it will be used—demodulating coded frames within the Bluetooth Classic (BTC) block. In this case, the SOVA mode is required, as it exhibits high sensitivity to changes in the resolution of the input signals. Therefore, it is crucial to study the impact of changing the input signal resolution on the soft outputs, which carry the bit reliability information and are passed to other blocks and processes within the digital communication flow of BTC.

The hard output mode is not directly affected by changes in resolution, as it relies solely on comparing metrics and making binary decisions. In contrast, the soft output mode depends entirely on the value of the metrics and their sequential relationships, making it much more sensitive to the precision of the input signals.

In Figure 16, the results of MATLAB simulations for SOVA are shown, considering the quantization of the input signal resolution. The original input signals had a resolution of 11 bits.

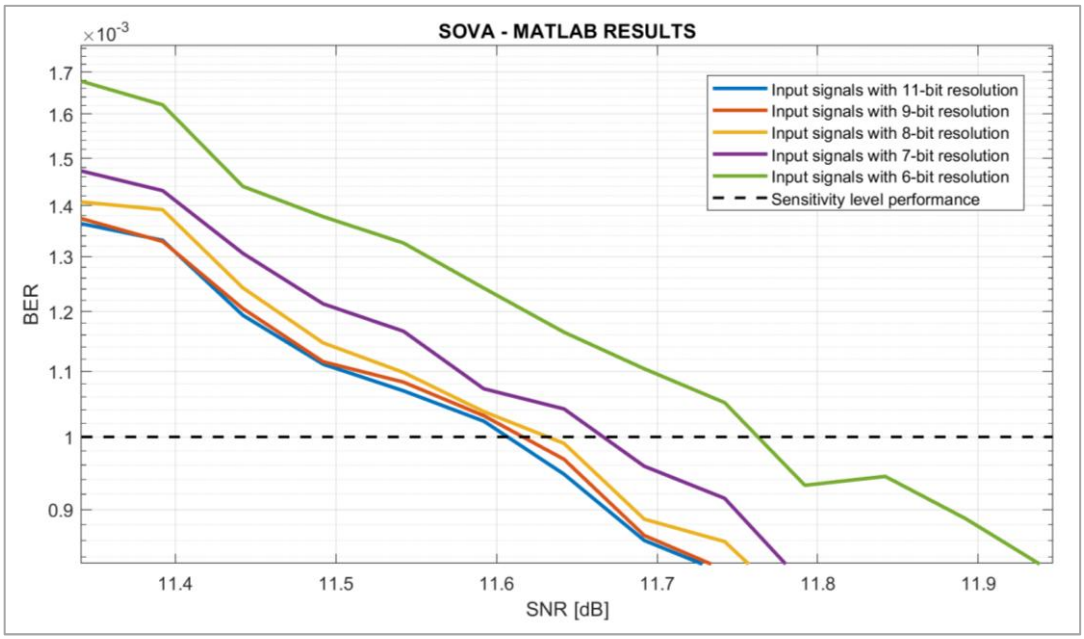

Figure 16: MATLAB SOVA results changing input signals resolution

The importance of Figure 16 lies in the evidence it provides about which resolutions are most favorable for hardware implementation. In this case, it is clear that input signals with 8-bit and 7-bit resolutions are the most promising, as they allow a reduction in hardware resource usage without significantly compromising performance.

Once the most promising resolutions were identified, I proceeded to modify my hardware design based on the MATLAB simulation results. These designs were synthesized and implemented on an FPGA, and the results are shown in Figure 17.

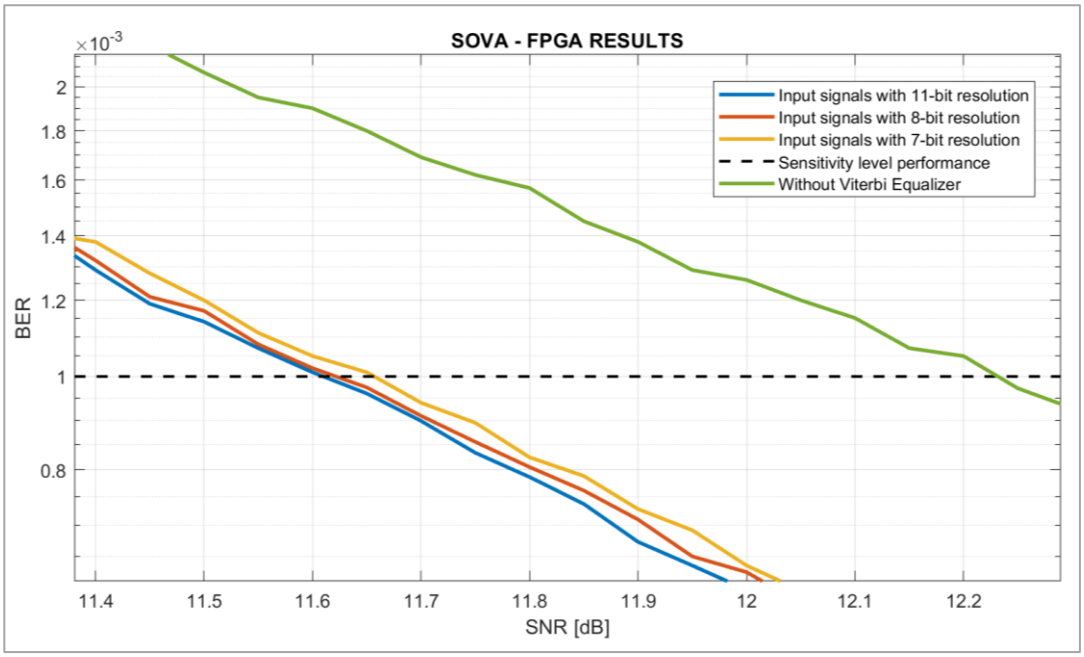

Figure 17: FPGA implementation results for different optimized SOVAs

Figure 17 is critical for making the final design decision. It demonstrates the gain obtained for both my unoptimized resolution design and the 8-bit and 7-bit resolution designs, compared to the results from the communication process without using the Viterbi Equalizer within the BTC block (This would imply connecting the soft inputs of the noisy phase differences directly to the next soft input decoding phase). It is also important to recall what

was discussed at the end of Section (3.3.): using an ideal SOVA process in MATLAB with full computational resources, it was concluded that to achieve a BER of 0.001, an SNR of 11.59 dB is required. This represents the theoretical performance limit for any optimized design.

The gain results are summarized in Table 2 below.

Table 2: Viterbi Equalizer gain relative to not using any demodulator

Design	**GAIN [dB]**
Ideal MATLAB Viterbi Equalizer without optimization	0.64
Viterbi Equalizer 11-bit input signals resolution	0.62
Viterbi Equalizer 8-bit input signals resolution	0.61
Viterbi Equalizer 7-bit input signals resolution	0.57

The analysis of Table 2 reveals two critical points. First, the version of the Viterbi Equalizer that does not have optimized input signal resolution performs similarly to the maximum performance that can be achieved, with a gain difference of only 0.02 dB, which is negligible under real-world conditions. Second, the results for the 8-bit and 7-bit resolution versions are also promising. If we prioritize resource usage over gain, the 7-bit version is the best option. However, if we prioritize gain over resource usage, the 8-bit version is clearly the most favorable.

As the designer of this block, I proposed prioritizing gain, evaluating the resource trade-off, and to provide further insight into the decision, I also performed ASIC synthesis (which will lead the final decision) to complement the FPGA synthesis results.

Table 3: FPGA Synthesis Results

FPGA Hardware Resources	LUT	FF
Original Viterbi Equalizer 11-bit input signals resolution	1120	342
New Viterbi Equalizer 11-bit input signals resolution	539	168
New Viterbi Equalizer 8-bit input signals resolution	417	137
New Viterbi Equalizer 7-bit input signals resolution	410	122

Tables 3 and 4 summarize the synthesis results in terms of resources and area for both FPGA and ASIC implementations. Additionally, the results of the original Viterbi Equalizer that was targeted for replacement as part of my internship project are included for a more comprehensive comparison and reflection within the context.

Table 4: ASIC Synthesis Results

ASIC Synthesis	Area [µm²]	Gates
Original Viterbi Equalizer 11-bit input signals resolution	1938	7100
New Viterbi Equalizer 11-bit input signals resolution	1043	3821
New Viterbi Equalizer 8-bit input signals resolution	780	2857
New Viterbi Equalizer 7-bit input signals resolution	688	2520

The results are highly favorable. First of all, the original design uses an area of 1938 μm², while my initial design uses only 1043 μm². This represents a 46.17% reduction in area, which is a substantial improvement. Reducing area is critical in ASIC designs as it directly impacts the cost of fabrication, power consumption, and the scalability of the design. However, we can go even further. The difference between the 8-bit and 7-bit designs in terms of area is still acceptable and affordable for the performance, and therefore, my proposal to use the 8-bit version for the input signal resolution is the most promising, as it further reduces the area by an additional 25% while maintaining a 0.61 dB gain compared to not using any demodulator.

CONCLUSIONS

The development of a fully functional Viterbi Equalizer for Bluetooth Classic applications represents a significant achievement. Starting from a theoretical foundation, I successfully designed and implemented a robust solution for improving coded frame demodulation within the Bluetooth Classic (BTC) block. The equalizer was developed from scratch, leveraging a deep understanding of both signal processing and hardware implementation.

The design process was grounded in a thorough review of the Viterbi Algorithm and its application to GFSK demodulation. Extensive MATLAB simulations provided critical insights into the performance of various configurations, while code debugging and iterative adjustments ensured that the final design was optimal within the given performance and resource constraints.

Through traceback depth analysis, I determined that a traceback depth of 4 provides the best balance between performance and resource efficiency. Furthermore, I optimized the design by quantizing the input signals, reducing the resolution from 11 bits to 8 bits. This quantization led to a significant reduction in the overall area and resource usage, making the design more practical for hardware implementation while maintaining performance within acceptable limits.

The final design was implemented in System Verilog and tested on FPGA, demonstrating a 3.59 dB gain in sensitivity performance at the BER target compared to the results of the original standard GFSK demodulator for hard outputs in non-coded frames. For soft outputs in coded frames, a gain between 0.61 dB and 0.62 dB was achieved compared to not using any demodulator. Additionally, my proposed design uses 46.17% less area than the original Viterbi Equalizer, which represents a significant improvement. Reducing area is crucial in ASIC designs, as it directly impacts fabrication costs, power consumption, and the overall

scalability of the system. Moreover, this optimization can be taken further with my proposal to use the 8-bit version for the input signal resolution, which reduces the area by an additional 25% while maintaining a 0.61 dB gain, making it the most promising balance between performance and resource efficiency.

Furthermore, one of the key strengths of my design lies in its parametrization and generalization, making it versatile and adaptable for other wireless communication standards that employ 2GFSK modulation. This flexibility allows for future applications beyond Bluetooth Classic, potentially benefiting a wide range of communication systems.

Throughout this internship, I enhanced my skills in signal processing, code abstraction, hardware design, and project management, working extensively with tools like Git, System Verilog, MATLAB, Questa Simulator, and Perforce version control. I also improved my time management, problem-solving, and creativity, gaining valuable experience in a professional and technical environment, as well as fluency in French.

In conclusion, the Viterbi Equalizer developed during this internship not only meets the performance requirements for the Bluetooth Classic BTC block but also introduces substantial resource savings, paving the way for more efficient designs in future wireless communication systems.

REFERENCES

Bluetooth SIG. (2016). *Bluetooth core specification version 5.0*.

Business Wire. (2023). *Silicon Labs celebrates 25 years of wireless innovation*. Retrieved July 5, 2024, from https://www.businesswire.com

Duc, K. L. (2004). *Channel coding techniques for wireless communications*. Academic Press.

Hagenauer, J., & Hoeher, P. (1989). A Viterbi algorithm with soft-decision outputs and its applications. In *Proceedings of IEEE Globecom '89* (pp. 47.11-47.17). Dallas, TX.

Patel, J., & Mehta, R. (2020). *Bluetooth 5.0 modem design*. Wiley.

Proakis, J. G., & Salehi, M. (2008). *Digital communications* (5th ed.). McGraw-Hill.

S. Berrou, C., Adde, P., Angui, E., & Faudeil, S. (1993). A low complexity soft-output Viterbi decoder architecture. In *Proceedings of the IEEE International Conference on Communications* (pp. 737-741). Geneva, Switzerland.

Shlezinger, A., Farsad, N., Eldar, Y. C., & Goldsmith, A. J. (2024). Meta-ViterbiNet: Online meta-learned Viterbi equalization for non-stationary channels. *IEEE Transactions on Wireless Communications, XX(X),* 1-12.

Silicon Labs. (n.d.). *Company overview*. Retrieved July 1, 2024, from https://www.silabs.com/about-us/company-overview

Viterbi, A. (2010). *Viterbi decoding of convolutional codes*. MIT. Retrieved July 5, 2024, from https://web.mit.edu/