A 7-Channel Code-Multiplexed Analog CMOS Front-End using an On-Chip Orthogonal Walsh-Hadamard Sequence Generator

by

Dion Gavin Mascarenhas

in partial fulfilment of the Masters Degree

at

the Delft University of Technology,

Faculty of Electrical Engineering, Mathematics and Computer Science

Under the supervision of Prof. Dr. Ir. Wouter Serdijn

Defended publicly on 22nd August, 2024

Student Number	5154731	
Thesis Committee	Prof. Dr. Ir. W.A. Serdijn,	TU Delft Supervisor
	Prof. Dr. Ir. K. Bult,	TU Delft
	Dr. Ir. Samprajani Rout,	External

TU Delft

Abstract

The recording of biosignals, such as atrial electrograms (AEG), electrocardiograms (ECG), and electroencephalograms (EEG), is progressing towards the adoption of more dense electrode arrays in order to improve spatial and temporal resolution. The proliferation of channels necessitates some form of channel sharing technique so that efficient low power and area recording can be achieved.

In this thesis, a code-division multiplexed analog front-end is designed due to its efficient use of bandwidth in comparison to time and frequency multiplexing. Orthogonal coding schemes such as Walsh-Hadamard sequences are best suited for minimising cross-talk. These codes are typically implemented using an LUT or processor which becomes a significant overhead when a large number of channels are to be multiplexed. However, by investigating the sequences closely, a novel method of generating these codes from a clock signal using digital logic was devised. The proposed algorithm to generate these sequences on-chip provides significant area savings for sequences of length greater than 8, which makes the design scalable for a large number of channels,. The reduction in area ranges from a factor of 10 for a code length of 8 up to a factor as large as 200 for a code length of 128.

As orthogonal sequences require a low bandwidth, a low bandwidth low-noise amplifier and ADC were used for amplification and digitisation of the signal. By using digitally inspired analog blocks, such as an inverter-based amplifier, lower power consumption could be achieved.

The entire design was implemented to share 7 channels. The code generator, low-noise amplifier and ADC consume a total of 78.4µW, which corresponds to 11.2µW per channel. This is a 3.3x improvement to the design in [5] where pseudo-random sequences were used as the coding scheme. However, the design consumes more power than a state-of-the-art design reported in [6] which uses 1.97µW per channel. While the design is currently not optimized with respect to power consumption in comparison to the design in [6], the novel code generation technique reported in this thesis makes the design scalable for larger number of channels. This is because the area constraint of the LUT is no more the limiting factor in terms of area.

Acknowledgements

As I come to the culmination of my master's thesis work over the past year, I write these few lines of appreciation to all those involved in helping and supporting me throughout this tough but fulfilling journey.

Firstly, I thank the Almighty God for giving me the opportunity to pursue my masters and giving me the strength, wisdom and perseverance to embark on this challenging journey.

My journey at the Department of Bioelectronics has been nothing short of remarkable. The praise and gratitude I indebt to Prof. Wouter Serdijn can simply not be captured by mere words. Right from understanding my difficulty in a different research group and accepting to mentor me in a new master's thesis, Prof. Wouter has been incredibly supportive in every aspect of the project. His mastery of telling you how to think instead of telling you what to do, although frustrating at times, has developed me into a much better designer. The flexibility he gave me to explore ideas and designs irrespective of whether they worked or not, made the journey of research immensely satisfying. Prof Wouter, I can go on and on, but here's a big shout-out to you for all that you've done. Thank you for molding me into a better designer in the world of Analog IC Design! I indeed cherish and will miss the discussions we had from amplifiers to ADCs, not to forget the long chats about politics, cookies, and even Dutch swear words!!

I would like to thank Dr. Sampi Rout who looked over this project and guided me with some of the nitty gritty details. A special word of gratitude to Prof. Klaas Bult, who although was not my supervisor, was always obliging to take time out whenever he was on campus to explain and simplify many of the concepts related to Amplifiers and ADCs.

Throughout my master's degree, a never-ending source of encouragement and support has come from my parents and sister. Mama, Dada, and Angie thank you for always being by my side and instilling in me the confidence that I can do it.

Lastly, I also thank my colleagues especially Karolis, Floris, Arnau and Amar from whom I learned a lot through casual discussions. A lot of knowledge gaps were plugged during these casual exchanges. Thank you, and I wish you the best in your future career as IC designers.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Background of Medical Condition

Atrial Fibrillation (AFib) is one of the most prevalent types of arrhythmia that occurs in the heart. The heart is divided into four chambers – the top two called atria, and the bottom two called ventricles. AFib is characterized by an irregular heartbeat that is caused when the upper chambers of the heart, the atria, beat chaotically and out of sync with the lower chambers of the heart, the ventricles.

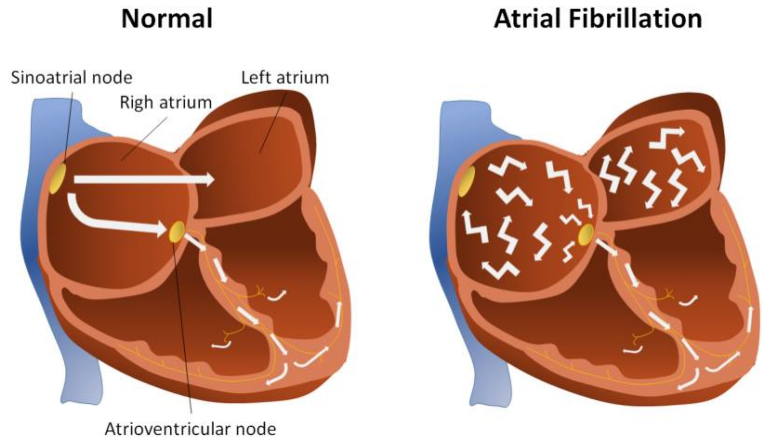


Fig. 1: Functioning of Normal Sinus Rhythm (Left) and Atrial Fibrillation (Right) [1].

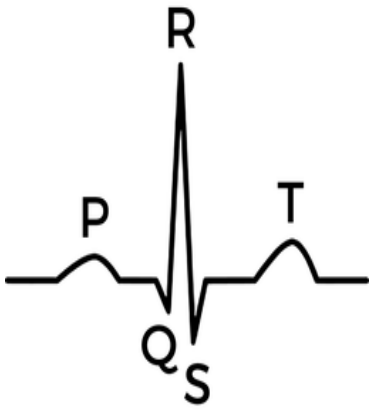


Fig. 2: Typical ECG signal [17].

In a normal heart rhythm, the sinoatrial node (SA node), present in the right atrium shown in Fig. 1, initiates electrical impulses that propagate through the electrical conduction system of the heart. This pulse regulates a normal heartbeat consisting of contraction and relaxation. The SA node is thus termed as the natural pacemaker of the heart. The activation of the SA node is characterized by P waves in an electro-cardiac signal, as shown in Fig. 2. In atrial fibrillation, however, there are sites in and around the atria, called ectopic sites, that initiate random electrical impulses. These random impulses cause the atria to quiver or fibrillate instead of contract. As most of these impulses do not pass the atrioventricular node (Fig. 1) into the

ventricle, there is a lack of synchronisation between the top and bottom chambers that leads to an irregular heartbeat. There are multiple stages in the occurrence of atrial fibrillation. The first stage is called *paroxysmal* which is the earliest stage. During this stage the quivers typically occur for less than a week and vary in frequency and intensity. The paroxysmal stage can either be severe that it leads to an attack, or mild enough that it resolves itself. The second stage is *persistent AFib* which usually lasts for more than a week and does not resolve itself. This stage develops after months or years of paroxysmal AFib. The last stage is *long standing persistent AFib* which is considered to be resistant to treatment, wherein complete elimination is highly unlikely. It is the most advanced stage in which the quivers are continuous for more than a year.

Due to the irregular heartbeat, the heart pumps out less blood which leads to some residue that can form clots. The most common implication is that these blood clots in the atria may then pass into the bloodstream and block small arteries which can be fatal. For instance, if the blood clot travels to arteries in the brain, it can result in a stroke.

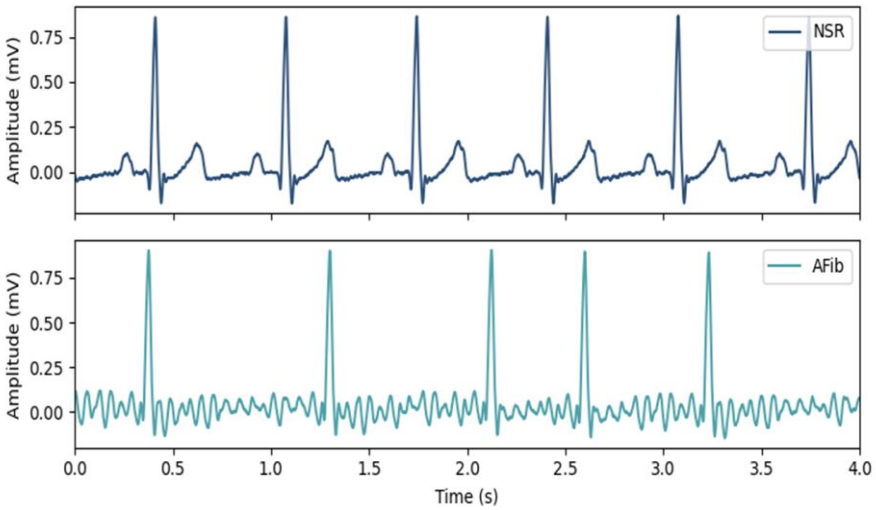


Fig. 3: Normal Sinus Rhythm (Top) and Atrial Fibrillation (Bottom) Waveforms [2].

The waveforms of a normal sinus rhythm (SR) and that of atrial fibrillation (AFib) are shown in Fig. 3. From the waveforms, it can be seen that in AFib, P waves are absent and there is a presence of irregular narrow QRS complexes. The P waves are absent because they are clouded by the activity of ectopic sites firing random impulses. The presence of larger ectopic sites results in flatter baselines between two spikes in the waveform.

1.2 Requirements & Challenges of Bio-Signal Acquisition

As is the case with all medical anomalies and diseases, mitigation can be done only when the root cause can be identified and mapped accurately. In the case of bio-signal recording, methods such as electrocardiography (ECG) and electroencephalography (EEG) are widely adopted. These methods of signal acquisition are popular due to their convenience and non-intrusiveness. However, since the electrodes are much further away from the site of activity to be recorded, they typically have low temporal and spatial resolution. This is because as the site of recording gets further away from the actual site that is being recorded, the signal undergoes filtering, and what is seen on the surface of the body is a vector summation of the signals over a distance. The spreading and refraction of the signals as they conduct through various layers of the body is termed as volume conduction which causes a drop in resolution. Reference electrodes also play a role in distorting the potentials due to contamination or electrolyte depletion that introduces low-frequency noise or baseline wander [3]. In situations where more detailed information is required for better accuracy, implantable chips with a multitude of electrodes close to the recording site are desired due to the high temporal and spatial resolution they offer. In AFib for instance, ECGs can provide sufficient information to detect its presence. However, they do not provide enough resolution to precisely map the sources of the substrate that perpetuate AFib. While an implantable with multiple electrodes does provide superior resolution, these devices come with their own set of challenges that need to be addressed. Some of these challenges are described below.

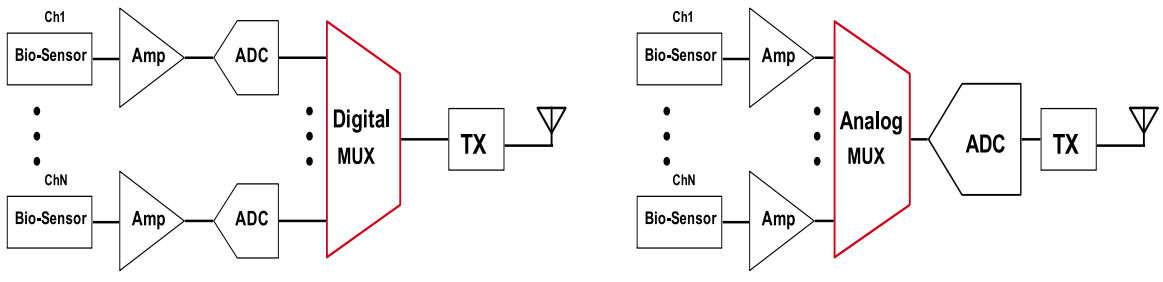


Fig. 4: Block diagram of typical multi-channel acquisition systems. On the left, each channel uses an LNA and an ADC of its own. On the right, the ADC is shared but each channel has its own LNA [4].

As an implantable device aims to be as minimally invasive as possible, the proliferation in the number of channels that are used in recording pose 3 main hurdles, [5].

i. Firstly - the circuit overhead scales proportionally to the number of channels. A typical recording system requires an LNA and ADC for each channel or only shares an ADC, as shown in Fig. 4. The increase in the front-end circuitry required for each channel causes large power and area consumption. This is highly undesirable for a minimally invasive SoC.
ii. Secondly - the long cable that reads out the data can cause significant degradation in the signal quality due to interference. The interferers could be from neighbouring channels, the power supply, or any external electromagnetic interference (EMI).
iii. Lastly - the number of wires going out restricts movement and can cause motion artifacts which once again will degrade the quality of the signal. The diameter of the catheter further restricts the number of wires that can be used for the readout.

1.3 Problem Statement

The recording of atrial electrogram (AEG) signals from multiple locations concurrently, through an invasive setup allows for identifying the trigger or substrate perpetuating AFib [7]. In the acquisition process, a low- or high-resolution recording may be required as per the doctor's/medical practitioner’s preference. A low-resolution recording can be performed when a mere activation map of the site is desired to be looked at. This low-resolution recording allows the doctor to “zoom out” and look at the overall features or propagation of the wavefronts in a larger region of interest. On the other hand, a high-resolution recording can be done when the doctor/medical practitioner desires to “zoom in” and look at more distinctive, clear features of the cardiac signals.

Minimizing the overhead of the readout circuitry and number of outgoing wires due to the large number of recording electrodes (channels) necessitates some form of circuit-sharing technique. The three forms of multiplexing to share the readout circuitry among the different channels are: Time-Division Multiplexing (TDM), Frequency-Division Multiplexing (FDM) and Code-Division Multiplexing (CDM). Chapter 2 describes the merits and flaws of each of the three techniques and justifies the use of CDM for this application. CDM is chosen, as it is the most

efficient in terms of bandwidth, which in turn allows for a huge reduction in chip area and power.

Thus, in the context of a multi-channel CDM system that makes use of a shared analog CMOS front-end, this thesis tries to answer/address the following:

i. How do various multiplexing coding schemes contrast in their capability to minimize cross-talk between channels while utilizing an optimal bandwidth?
ii. How can generation of these multiplexing codes be achieved on-chip with minimal form factors that show a significant improvement from existing techniques?
iii. To what degree can digital signal processing (DSP) off-chip help alleviate the recording constraints so that an area and power efficient chip can be realised?
iv. What circuit architectures can be deployed for higher efficiencies to reduce power and area that are also suitable for advanced technology nodes with shrinking voltages?

1.4 Thesis Organization

This thesis is organized in the following manner. In Chapter 2, the three channel-sharing techniques are evaluated qualitatively to make a choice for the system. As CDM is chosen as the multiplexing technique, Chapter 3 delves into spread-spectrum modulation and some of the various kinds of codes: its properties, efficiency in terms of bandwidth usage, cross-talk and ease of generation. Chapter 4 describes the system-level design and derives the specifications of the individual blocks. Chapter 5 describes how a new and efficient method of generating Walsh-Hadamard sequences using simple digital logic was devised and the systematic approach that allows a WH code of any length to be constructed using a simple algorithm. In Chapter 6, the circuit-level implementation of the code generator, low-noise amplifier, and analog to digital converter is provided. The circuit schematics, design methodology and simulation results are extensively explained in this chapter. Finally, Chapter 7 concludes the thesis with the key results and future recommendations. Additionally, some of the computations and design of certain circuit blocks have automation routines that are written in MATLAB and are available in the appendices.

Chapter 2

Channel-Sharing Techniques & Prior Work in Shared Analog Front Ends

To tackle the challenges and limitations imposed by a multi-channel recording system, as stated in Section 1.2, sharing of the front-end resources among a certain number of channels must be performed. By using a single LNA and/or ADC for multiple channels, significant strides can be made in reducing the power and area consumption. The three available multiplexing schemes are Time-Division Multiplexing (TDM), Frequency-Division Multiplexing (FDM), and Code-Division Multiplexing (CDM). Each of these schemes have different implications on the system such as complexity of implementation, efficient use of bandwidth, as well as determining which blocks can be shared. In this chapter, a qualitative analysis is carried out for each of these schemes in the context of analog front-ends. After evaluating the various benefits and drawbacks of each technique, CDM is ultimately chosen.

2.1 Time-Division Multiplexing (TDM)

Time-Division Multiplexing/ Time-Interleaving is a technique in which multiple channels can share circuit resources by allocating a fixed time slot for each channel to be readout. In the case of analog front-end designs that involve an LNA and ADC, either the ADC or both the LNA and ADC can be shared. The most obvious choice would be to share both the blocks (Fig. 5) so that the power of one amplifier and one ADC can be used to process multiple channels. The implications of time multiplexing N channels on the SNR, bandwidth and settling time is briefly explained below.

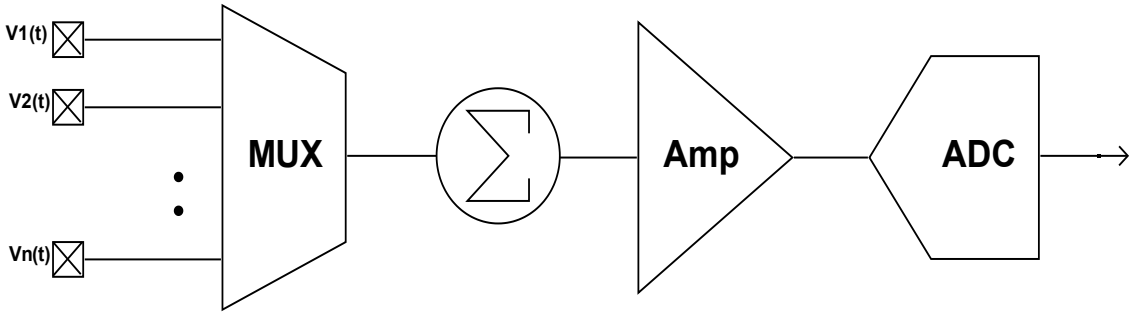


Fig. 5: N input channels are time-multiplexed and summed to share one amplifier and ADC

As explained in [8], for N channels without an anti-alias filter, the SNR is proportional to $\frac{1}{N}$. The primary reason for this limitation is the difference between the signal and noise bandwidths and the different sampling frequency as well. In TDM, the signal is sampled at $\frac{F_S}{N}$, whereas the noise is sampled at *Fs*. Since the amplifier is shared, its bandwidth must be N times larger than that of a single channel. This is required to be able to process all the channels, i.e. it must have a large enough bandwidth to be able to settle to a particular accuracy before the next sample is taken. A consequence of this is that even though each signal uses only $\frac{1}{N}$ of the total available bandwidth, the noise is integrated across the entire bandwidth. Hence, the noise across the entire bandwidth is N times larger than that of a single channel bandwidth. This causes a decrease in the SNR for an increasing number of channels. Apart from noise, the high bandwidth required by the amplifier for each channel to settle in time is not fully utilized by each channel. This is essentially a waste of the total available bandwidth.

As a result of this limitation, typically in many time-division multiplexed designs, each pair of electrode (channel) uses a dedicated amplifier to ease the noise and settling requirements. However, for recording systems that have 100s of channels, this cannot be easily scaled, as area and power constraints once again come into play. This concludes the fact that TDM is not the most efficient channel sharing technique, as it either requires large bandwidth that is not optimally used or requires a dedicated amplifier per input channel.

2.2 Frequency-Division Multiplexing (FDM)

Frequency-Division Multiplexing is another method that can be used to share the analog front end by allocating a different frequency band to each channel. Traditional modulation/chopping, as shown on top of Fig. 6, uses a single frequency to modulate each channel in parallel. This method requires an amplifier and low pass filter for each channel. This is of little to no interest in this application as the number of amplifiers and filters scale proportionally with the number of channels, eating up into the power and area overhead.

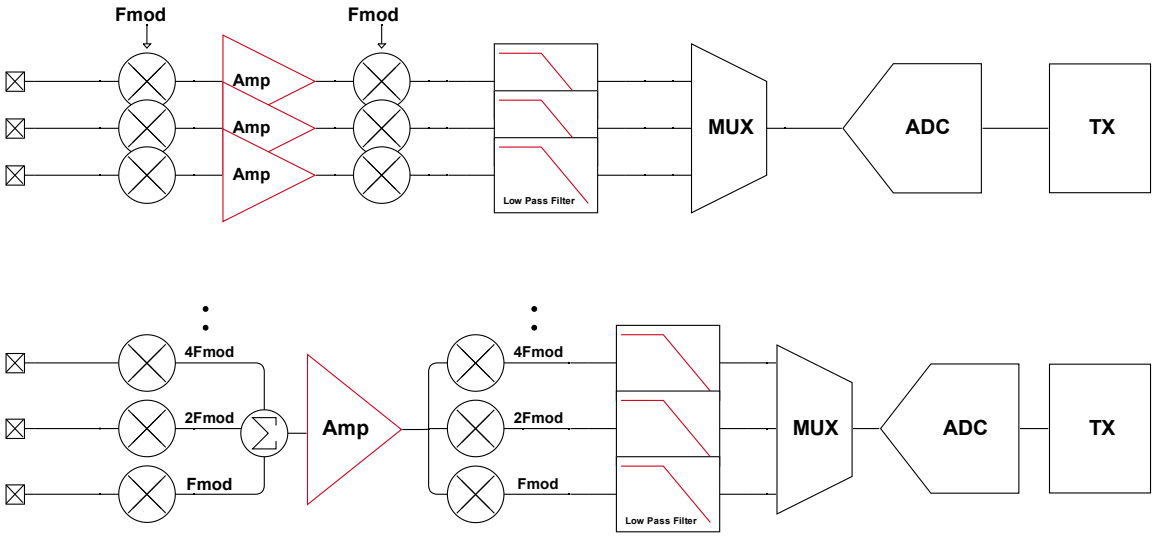


Fig. 6: Traditional Chopping (Top) and Orthogonal Frequency Chopping (Bottom) [6].

An alternative solution is to use orthogonal frequency chopping (OFC). In this technique, each channel is allocated a different frequency. For this to be effective however, the modulation frequencies are chosen in multiples of 2. This is done to reduce the intermodulation products between neighbouring channels and maintain orthogonality [9]. After passing through the shared amplifier, the signals are demodulated back into baseband. In both the above techniques, modulating the input signal makes it easier to get rid of flicker (1/f) noise as well as other noise sources that get added in the signal path. The noise sources such as external interference, flicker and EMI are upmodulated at the demodulation side, which are then filtered out through a low-pass filter. However, it must be noted that the thermal noise still remains but is relatively easy to suppress by increasing g_m, although that means a higher current and power consumption.

While OFC makes use of the shared analog front end, the issues and challenges it brings about do not make it a suitable choice. Firstly, the chopping frequency for N channels grows exponentially as the frequency for each channel is double the frequency of the previous channel. Thus, the highest frequency to be generated is $2^{(N-1)}*f_c$, where f_c is the chopping frequency of the first channel. This implies that the shared amplifier requires a bandwidth equal to that of the fastest chopper, thereby using a massive amount of power [6]. Additionally, each of these frequencies to be generated requires either a dedicated oscillator [10] or can be done using frequency division circuits in the digital domain. On-chip oscillators occupy a lot of area and consume huge amounts of power. While the frequency division circuits are area efficient, the power overhead for higher frequencies is still significant. All these factors do not satisfy the constraints on area and power for this particular implantable SoC. Thus, it is concluded that the increase in implementation complexity for OFC comes without much benefit.

2.3 Code-Division Multiplexing (CDM)

In code-division multiplexing, the same frequency band is used for multiple channels by assigning a different code/sequence to each of the channels. CDM makes use of the concept of spread spectrum, wherein a narrow band signal is purposely spread into a wideband signal. This is achieved by modulating the input with a code sequence that has a much higher bandwidth in comparison to the bandwidth of the original signal. The reason for spreading the signal across a larger bandwidth is its immunity to interference and band-limited noise sources [11]. A more elaborate explanation of spread spectrum and its benefits are in Chapter 3.

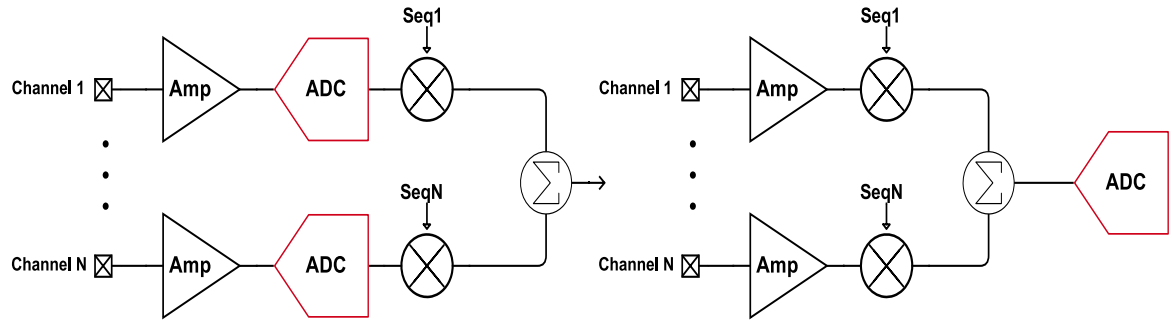


Fig. 7: Conventional Code Division Multiplexed Systems. On the left, the digital output is multiplexed before transmission and there is no sharing of the amplifier or ADC. On the right, the amplifier output is multiplexed to share the ADC.

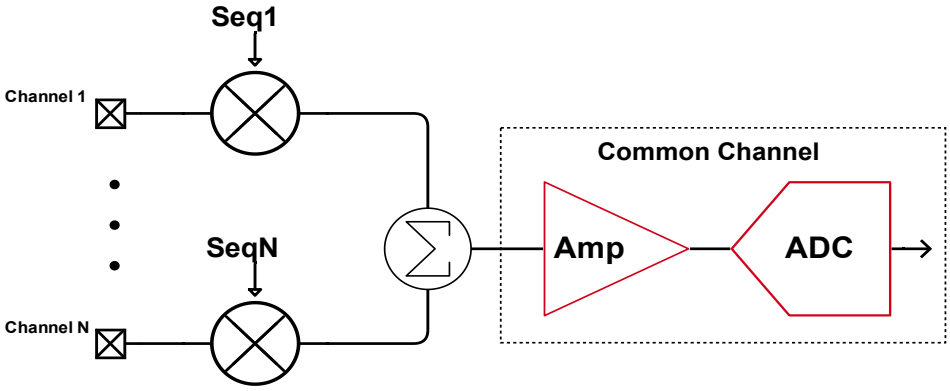


Fig. 8: Direct Coding of Input Channels in the Analog Domain [5].

CDM can be applied either in the digital or analog domain. Conventional designs code the channels either after digitization or amplification as shown in Fig. 7. Modulating the signal after digitization still requires an LNA and ADC per channel, hence is of no interest. Modulation after amplification has two distinct disadvantages. Firstly, the number of amplifiers scales with the number of channels. Secondly, the noise sources such as flicker noise and any external interference or EMI can fall in band and become inseparable. Removing the interferes would require filtering whereas flicker noise cannot be removed since it falls exactly in the signal band. Since the objective is a shared channel, the inputs are coded directly in the analog domain (Fig. 8). This allows us to reap the benefits of the flicker noise being modulated out of band as well as makes use of a shared LNA as well as ADC.

Since CDM uses the same frequency for all the channels, the bandwidth of the amplifier can be optimally used as all the channels are stacked by adding up the signals on top of each other. Therefore, each signal makes use of the entire available bandwidth without wasting any resource. By digitizing the signals prior to demodulation, the overhead that comes with encoding for the transmitter can be avoided. Furthermore, the demodulation can be performed off chip, relieving the power and area constraints [6]. At the receiver end, the signals are multiplied with the same codes they were modulated with. It is important to synchronise the demodulation for the accurate recovery of the signal. If the signals are multiplied without synchronisation, the output signal will just look like noise. This is one of the properties of a spread-spectrum signal as explained in Chapter 3. At the demodulation side, the unwanted noise and interference is upmodulated by the code at the receiver end and can be filtered out easily.

Mathematically, the transmitted and received signal can be expressed as in [5]:

The sum of the incoming signal u(t), modulated with the codes c(t) is:

$$m(t) = \sum_{i=1}^{N} u_i(t) * c_i(t)$$

(2.3.1)

The transmitted signal after addition of noise sources, interference and subsequent amplification and digitization is:

$$s(t) = A\big[\, m(t) + n_{1/f}(t) + \, n_{\mathrm{th}}(t) + \, n_{\mathrm{emi}}(t)\big] + \, n_{\mathrm{q}}(t)$$

(2.3.2)

Finally, on the receiver end the received signal s(t) is multiplied with the code again to give:

$$r_i(t) = s(t) * c_i(t)$$

$$r_i(t) = Au_i(t) + \, c_i(t) * \left[An_{\frac{1}{f}}(t) + \, An_{e\mathrm{mi}}(t) + \, n_{\mathrm{q}}(t) \right]$$

(2.3.3)

From this analysis, we observe that CDM allows for the efficient use of bandwidth as the signal uses the entire available bandwidth of the amplifier. If the input signal is directly modulated, CDM has the nice property of OFC modulation wherein it modulates most of the unwanted noise sources and interference out of band. This removes the need for steep filters. With these benefits of code-division multiplexing, it is deemed as a perfect option for the sharing of analog front-end circuitry in a multi-channel signal-acquisition system. The main consideration to be taken into account for CDM is the choice of codes and the frequency at which they operate. These two parameters must be chosen appropriately as they are key to minimizing the cross talk between channels and keep the unwanted tones out of band. Chapter 3 describes the different type of codes, their properties and their ease of generation in detail.

Chapter 3

Spread-Spectrum Modulation – Pseudo Random and Orthogonal Sequences

As Code-Division Multiplexing is chosen as the channel-sharing technique, in this chapter the concept of spread spectrum is discussed as CDM essentially spreads the spectrum to a larger bandwidth. Additionally, the different types of codes, their properties and ease of generation are discussed. This is done to choose the appropriate codes for modulation to maximise performance by minimising cross-talk between channels.

Spread-spectrum modulation is a technique in which a signal at baseband with a particular bandwidth is purposely spread to a much larger bandwidth by modulating it with a higher frequency signal. The modulation is performed by multiplying the signal with a code or sequence running at a higher frequency. Fig. 9 depicts this operation wherein a narrow band signal (green) is spread to a wideband signal (blue). The higher the frequency of the code signal, the wider the baseband signal is spread. The resulting signal is a noise-like looking signal which provides the basis of many benefits of this technique [11].

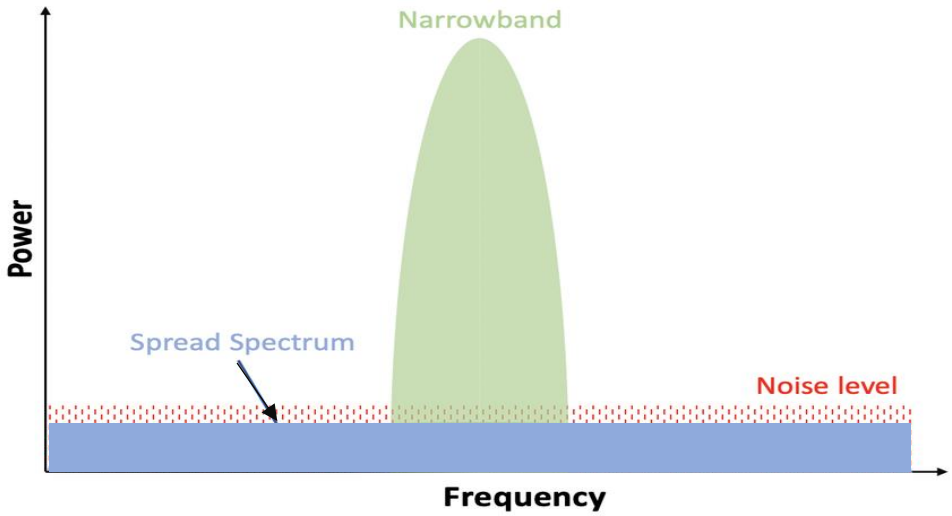


Fig. 9: Narrow band signal spread to a wideband signal by multiplying it with a code sequence of a much higher frequency [18].

Typically in communication systems, spreading the spectrum is done for reasons such as anti-jamming, anti-interference, multi-user random access communications and low interception probability. The reason for this is best visualized from Fig. 10 below. Since the desired signal has been spread, it is at a much lower power spectral density spanning across a larger frequency range. At the receiver or demodulation side, the incoming signal contains the desired signal along with all the jammers, interferers and noise. This signal is now multiplied with an identical copy of the sequence with which the first spreading operation was carried out. This has two consequences. Firstly, the desired signal is now de-spread and its bandwidth collapses to that of its original narrow bandwidth. However, the interferers and jammers (if they are bandlimited) are now spread over a large bandwidth as they haven't been modulated before. The undesired signals now have a much lower power spectral density and are spread over a much larger bandwidth. Hence, the portion of the interferers falling in-band is greatly reduced and a better SNR can be achieved. In a multiple-access application, all the codes operate at the same frequency and hence make use of the entire available spectrum. Despite occupying a larger bandwidth than the original signal, it is compensated by the fact that multiple channels or users occupy the spectrum.

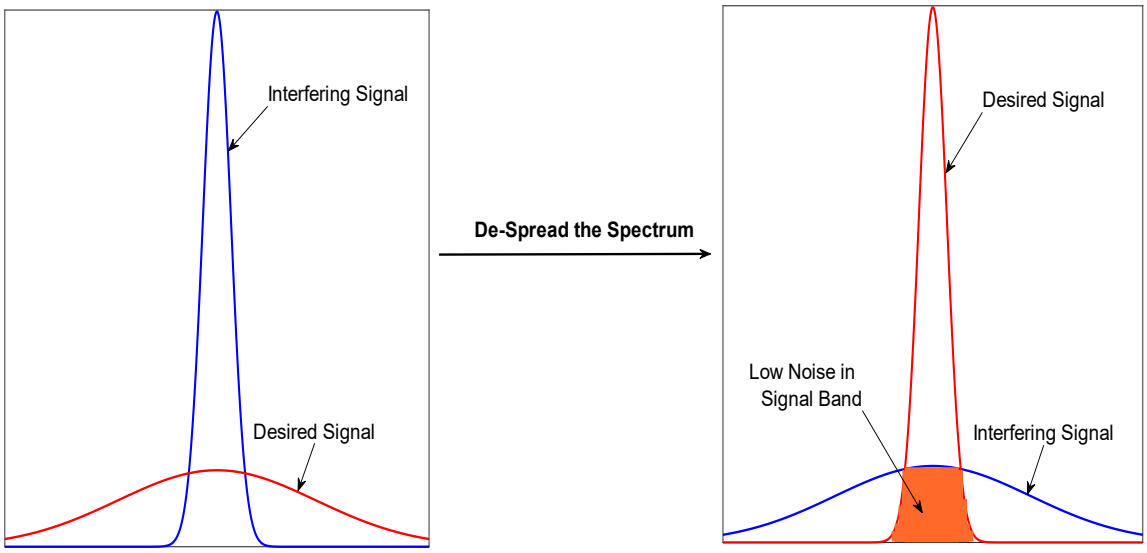


Fig. 10: De-spreading of the signal at the receiver/demodulating end causes the original signal to collapse into a narrow band signal. However, now the interfering signals (if bandlimited) are spread and their effect on the signal quality is reduced as a much smaller power density of the interferer falls in band of the signal.

There are several techniques by which this spreading of the spectrum can be performed. The first is *time-hopping spread spectrum* (THSS). In this technique, the signal is transmitted in brief spurts at random intervals of time. The time spurts are initiated by a pseudorandom sequence. The second is *frequency-hopping spread spectrum* (FHSS), wherein a carrier is used to shift the frequency of the signal in a pseudorandom way. The third is *direct-sequence spread spectrum* (DSSS). DSSS directly multiplies the code with the signal by causing phase transitions in the original signal itself [11]. In all cases the sequences are not truly random as a copy of the sequence is needed at the demodulation side to precisely recover the signal. In the domain of Analog IC Design, DSSS is the same as the chopping technique and can be directly applied to the input with only two pairs of switches which makes its implementation extremely easy. FHSS needs a frequency synthesiser for the sequence to translate into different frequencies. Hence, its complexity is increased. Finally, THSS applies the sequence to the transmission and is applied to the amplifier. This involves switching the amplifier on or off which requires careful design to ensure that the amplifier can operate accurately while being switched. Considering the implementation complexity of the three schemes, DSSS is the easiest to implement while at the same time providing benefits such as reducing the effect of 1/f noise and offset. Hence, in this project, DSSS was implemented.

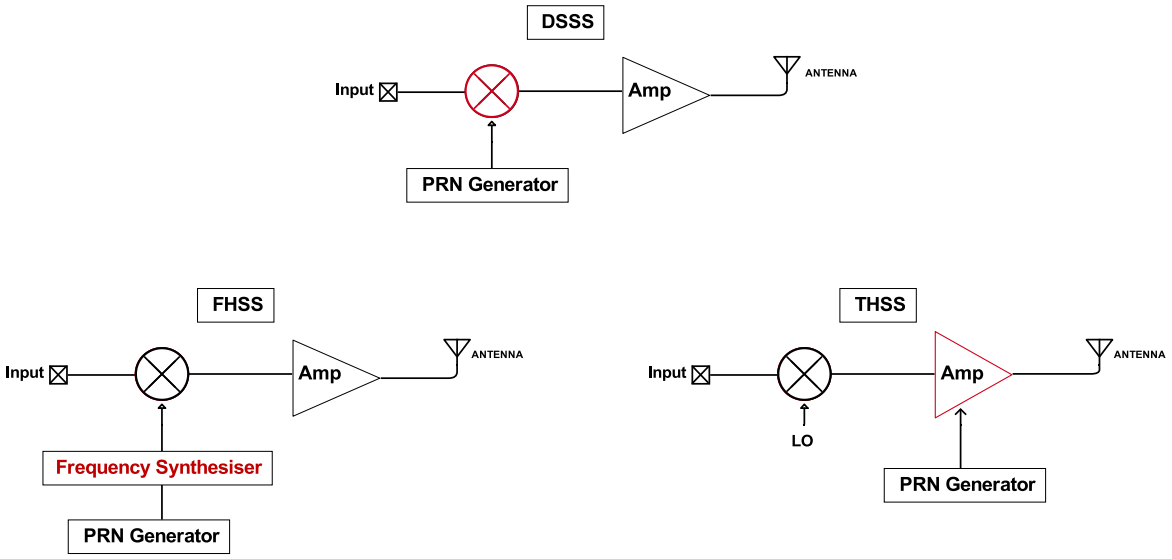


Fig. 11: The 3 different spreading techniques. The top figure depicts Direct Sequence Spread Spectrum (DSSS). The bottom left depicts Frequency Hopping Spread Spectrum (FHSS) and the bottom right is Time Hopping Spread Spectrum (THSS).

While in theory, a truly random sequence can modulate the signal, in practice it cannot be random as a copy of the code is required for demodulation. Hence, pseudorandom codes are typically used for modulation. There are various types of codes that are available, each with their own properties depending on the application. In the sections below, the properties and generation of two classes of codes and how they are generated is discussed keeping in line with the properties desired for a shared analog front end. The two classes are pseudorandom bit sequences (PRBS) and orthogonal Walsh-Hadamard (WH) sequences.

In this thesis, DSSS is of main interest as code-division multiplexing is essentially a direct-sequence modulation scheme that spreads the bandwidth. As the signals read out from the electrode array in this project are done by a wire, aspects such as anti-jamming and security are not of primary concern. The main aspects that need to be taken into consideration are the cross-talk between channels and the ease of generation or storage of these sequences on chip. Cross-talk between channels can occur if there is non-zero cross-correlation between the various code sequences. This will affect the recovery of the signal as the spillover of other channels will degrade the SNR. Since there will be capacitive coupling on chip between channels that already degrades the SNR, it is important to use code sequences that do not exacerbate this problem. Choosing orthogonal codes gives a zero cross-correlation in theory, although in practice it will be some non-zero value. The non-zero cross correlation in practice comes from non-idealities in real systems such as synchronisation errors, system noise and jitter that do hamper the perfect alignment of the sequences in the time domain. The small time shifts present from one sequence to another will deteriorate its cross-correlation property. However, as seen from [6], the cross-talk is sufficiently low to not have an adverse effect on performance.

In the following two subsections, PRBS and WH codes are explained in greater detail. Since this thesis makes use of WH codes due to their orthogonality, there is more emphasis placed on this type of code. However, PRBS codes and their generation is briefly discussed too in Section 3.1 and more information can be found in Appendix A. Section 3.2 gives an explanation of how the WH matrices come about in the mathematical domain and why computation or memory seems necessary. Following this, a novel simplified method of generating the codes is proposed with an example in Chapter 5.

3.1 Pseudo-Random Bit Sequences (PRBS)

Introduction, Types and Properties of PRBS Codes

Pseudo-Random or Pseudo-Noise (PN) Sequences are a type of sequence that spread a spectrum and can be used for CDM. They possess good randomness properties, have long periods and are easy to generate [11]. The two main pseudo-random codes that are explored in this thesis are the Maximum Length (ML) and Gold Sequences. While there are other types of sequences such as Kasami, Gold-like and Dual-BCH sequences, they were not explored as a much better code in terms of performance was achieved and is explained in Section 3.2. In this section, the properties and generation of ML and Gold sequences are briefly discussed as some of the work carried out in this thesis went into exploring them as possible implementations for CDM in shared analog front-ends.

In terms of circuit implementation, these PN sequences can be easily realized using linear feedback shift-registers (LFSR). LFSRs are digital circuits that use a shift register whose input is a linear function of two or more states of the intermediate shift registers [14]. Fig. 12 shows how an LFSR is realized. The input is a feedback connection of the XOR function of the taps at positions 3 and 5. The output sequence of these LFSRs is determined by the initial state loaded onto the shift-registers and the feedback connections. The feedback coefficients are determined by a certain type of polynomials. More information about the generation of the polynomials is available as a MATLAB automation script in Appendix A.1. The script was used to generate polynomials up to the 13^{th} degree. Larger degrees are possible but will take a much longer time to generate due to the sheer number of polynomials that exist.

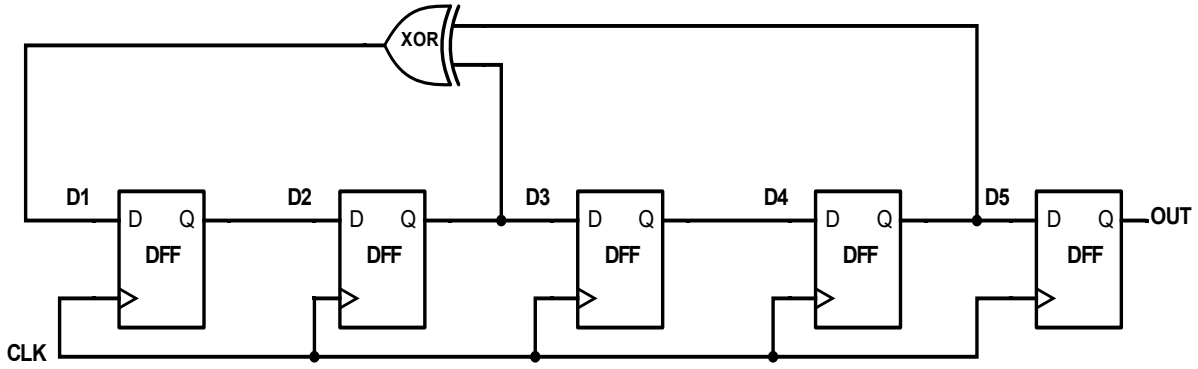


Fig. 12: Linear Feedback Shift Register (LFSR) formed by feeding back the XOR of tap 3 and 5

Maximum-Length Sequences: This is a type of sequence in which the period of the sequence is $2^N - 1$, where N is the degree. For instance a 4 state register has a maximum period of 15 $(2^4 - 1)$, since the all-zeros case is not included. The choice of the feedback taps determine whether a maximal length sequence is produced or not.

The determination of these polynomials stems from field theory, and more specifically belong to a small finite field termed GF(2). Within this field, the polynomials of a given degree must be irreducible and primitive polynomials. A maximum-length sequence is only generated by primitive polynomials. The MATLAB scripts in Appendix A.1 generate these primitive polynomials. The feedback taps are the coefficients in the polynomials. For instance, Fig. 12 is a circuit that generates an ML Sequence, that incorporates the polynomial $x^5 + x^3 + 1$. The feedback taps are 3 and 5 and 1 is the input. The actual output sequence also depends on the initial states that the registers are set to.

The auto-correlation of ML sequences are two-valued:

$$\begin{matrix} N, & at\ 0\ shifts \\ -1, & at\ all\ other\ shifts \end{matrix}$$

(3.1.1)

The cross-correlation is much more complicated. In this case we focus on special class called the *preferred pairs of polynomials.* These are pairs of ML polynomials that either have a 3- or 4-valued spectra. The 4-valued spectra is if n is a multiple of 4 and the 3-valued spectra for other cases. The different cross correlation values in the spectra are given by the formulae:

$$\left.\begin{matrix} -1,\ -t(n)\ \&\ t(n) - 2 \\ -1, -1 + 2^{\frac{n}{2}}, -t(n)\ and\ t(n) - 2 \end{matrix}\right\} \quad t(n) = 1 + 2^{\frac{n+2}{2}}$$

n is the degree of the sequence

(3.1.2)

If, for a given application, a single ML sequence is required, using any of the primitive polynomials is acceptable. However, if there are multiple required, it is better to use these pairs of preferred polynomials for better cross-correlation properties. A more detailed mathematical description can be found in [12].

Gold Sequences: These are a class of sequences with much better cross-correlation properties in comparison to ML sequences. As seen from the cross-correlation formula for a preferred pair of polynomials, as n increases, the peak cross-correlation also increases. Gold codes solve this problem by performing the XOR operation between two preferred pairs of polynomials. The auto-correlation remains good while the cross-correlation is reduced.

The auto-correlation of a Gold sequence $= 2^N - 1$

The cross-correlation between 2 sequences is $-1, -1 + 2^{\frac{n+1}{2}}$ & $1 + 2^{\frac{n+1}{2}}$

As these codes still have non-zero cross-correlation, it affects the cross-talk between channels and hence the following section addresses a set of orthogonal sequences which are favourable for channel sharing.

3.2 Walsh-Hadamard Sequences

Introduction and Properties of Walsh-Hadamard Codes

The Walsh-Hadamard transform belongs to a class of Fourier transforms from which orthogonal matrices are generated. The starting point is a base matrix of size 1, defined as:

$$H_1 = [1]$$

The consecutive matrices of size $2N$ are computed from a base matrix of size N by the recursive Kronecker product that can be expressed as [13]:

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$$

For a 2x2 matrix this is defined below. From the following sections, the -1 coefficients will be replaced by a 0 for ease of visualization and resemblance to binary signals in the electrical domain.

$$H_{2_{1,-1}} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \leftrightarrow H_{2_{1,0}} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Similarly a 4x4 and 8x8 matrix can be formulated as:

$$H_{4_{1,0}} = \begin{bmatrix} H_{2_{1,0}} & H_{2_{1,0}} \\ H_{2_{1,0}} & -H_{2_{1,0}} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$$H_{8_{1,0}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

As these codes are orthogonal, the cross correlation between each of them is 0. This is one of the key properties that makes this type of code useful for CDM. Fig. 13 shows the proof of this in the plot below.

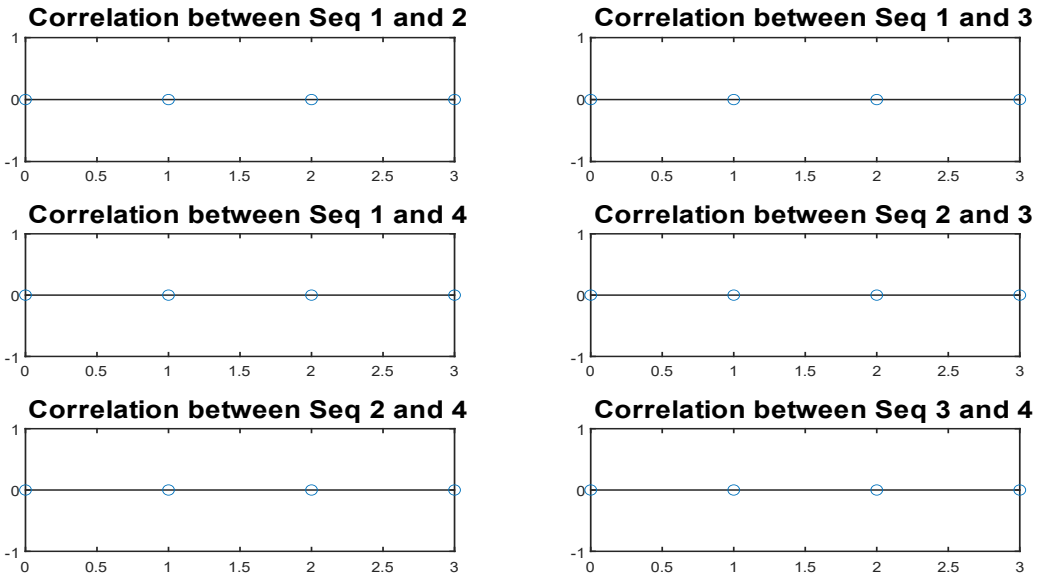


Fig. 13: Zero cross-correlation between all sequences for a 4x4 WH Matrix

By evaluating the correlation between all the possible sequence combinations for a 4x4 matrix, it can be seen that the cross-correlation between any two sequences in a set is equal to 0. On the vertical axis of the plots, the cross-correlation value is depicted and along the horizontal axis is the shifts by each bit.

From the above analysis and comparison of PRBS codes and WH codes, it can be concluded that WH codes, being orthogonal, are best suited for code multiplexing since the main limitation is reducing cross-talk in this application. In the following chapter, by making use of the WH codes, the system level design will be built up to quantify the optimal parameters required to realise such a system.

Chapter 4

System-Level Design

In the previous chapters, the use of Code-Division Multiplexing (CDM) to share a number of channels was deemed to be the best choice from a qualitative point of view. As the most important property required for the application is to minimise crosstalk, orthogonal codes are best suited due to their zero cross-correlation. In this regard, Walsh-Hadamard Codes satisfy the property of zero cross-correlation and are used as the coding scheme.

In this chapter, the system level design will be constructed and design specifications will be quantified. Firstly, the signal characteristics are quantified based on available recordings, after which the best possible system level configuration is described to extract the most optimal performance with the simplest hardware possible.

4.1 Signal Properties

The AEG signals recorded have a signal amplitude ranging from 1-10mV_{pk-pk} for each recording channel. The recording bandwidth ranges from 0.5 – 400 Hz. In this thesis, as there was no access to the actual recording data for the AEG signals, the following specifications are devised using data from MIT-BIH arrhythmia and atrial fibrillation databases available at physionet.org [15][16]. These are recordings from laboratories at Boston's Beth Israel Deaconess Medical Center that is available for basic research into cardiac dynamics. Both these datasets contain ambulatory ECG recordings instead of atrial electrograms. For more accurate results, these specifications may need to be revised by testing the system on actual recorded data. However, AEG signals are quite similar to that of ECG signals in terms of their signal characteristics such as amplitude range and peaks. Hence, the validation below is justified and can be quite accurate for the intended application.

The signal amplitude for both these sets of data have a 10mV range. The arrhythmia database is recorded at 360 samples per second and digitized with a 11-bit resolution per channel. On the other hand, the fibrillation database was recorded at 250 samples per second and digitized with a 12-bit resolution per channel. Fig. 14 plots the signals in the time and frequency domain.

From the frequency-domain plot, it can be devised that these signals have mostly low-frequency components, as these frequencies are seen to have much larger magnitude in the PSD plot. Thus, most of the important features to identify these arrhythmias are captured in the low frequency region up to a few 10s of hertz. The low-frequency, low-bandwidth signals tie directly into a choice for the modulation frequency at which the codes operate. This in turn simplifies the design of the ADC as there is no need to operate at very high speeds to obtain sufficient resolution.

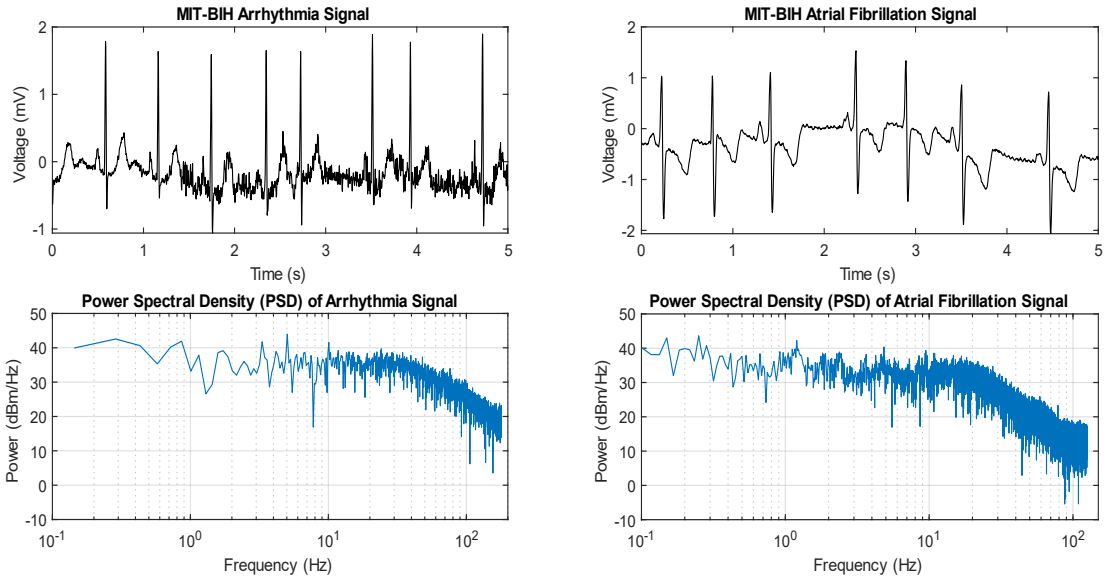


Fig. 14: Arrhythmia (left) and atrial-fibrillation (right) ECG signals in the time (top) and frequency (bottom) domain. Most of the features are captured in the lower frequencies as seen from the higher PSD values in the low-frequency region.

4.2 Required Resolution

As this application targets low-resolution recording, the following experiment is done to determine the minimum amount of SNR required to maintain signal integrity and keep a low percentage root difference (PRD) between the original and reconstructed/corrupted signal. The PRD is computed to ensure that essential features of the signal - especially the slopes and peaks are not lost and are visible to the clinician to make a diagnosis. Although a PRD of less than 5% is usually an acceptable value, here, a slightly tighter constraint of 3% is designed for. This is to keep some margins for other types of noise, interference and distortion that are not modelled. However, as shown in Section 2.3, the spreading and de-spreading gets rid of most of the noise, and hence, this model provides a suitable estimate of the required resolution.

The MIT-BIH signals mentioned above are recorded with a fairly high degree of resolution of 11 and 12 bits. These signals are extracted in MATLAB and white noise is added to these signals incrementally. At each iteration of the addition of the noise level, the SNR of the noisy signal is computed along with the PRD between the original signal and this noisy signal. From Fig. 15 it is evident that for both the types of signals, there is a knee in the curve beyond which there is minimal improvement. For a PRD of less than 3% an SNR in the range of 28-32dB is sufficient. This corresponds to a resolution of 5-6 bits. Considering the non-idealities and non-linearities in an ADC, a 6 bit ADC seems sufficient for the recording of the channels.

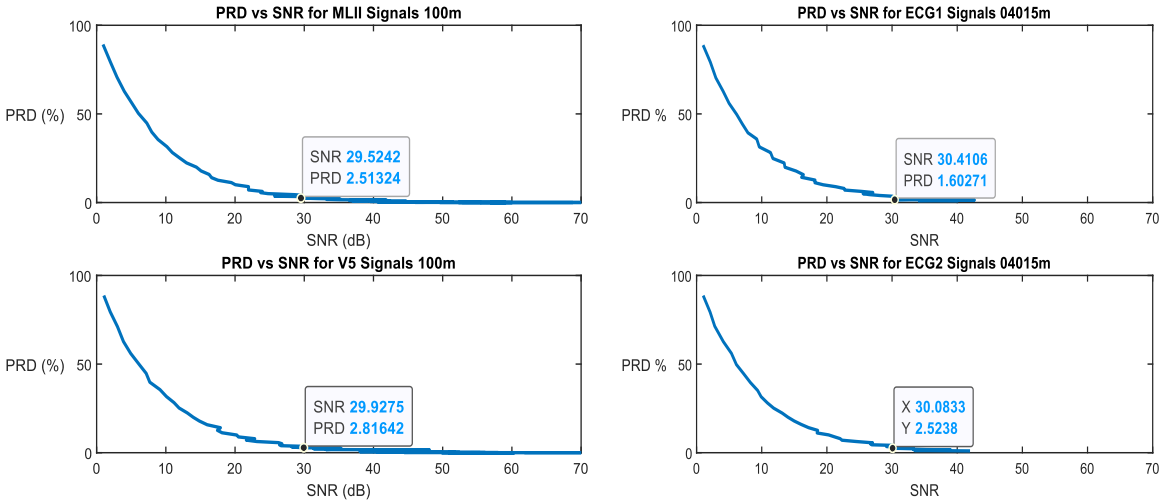


Fig. 15: SNR vs PRD for Arrhythmia (left) and Atrial Fibrillation (right) Signals.

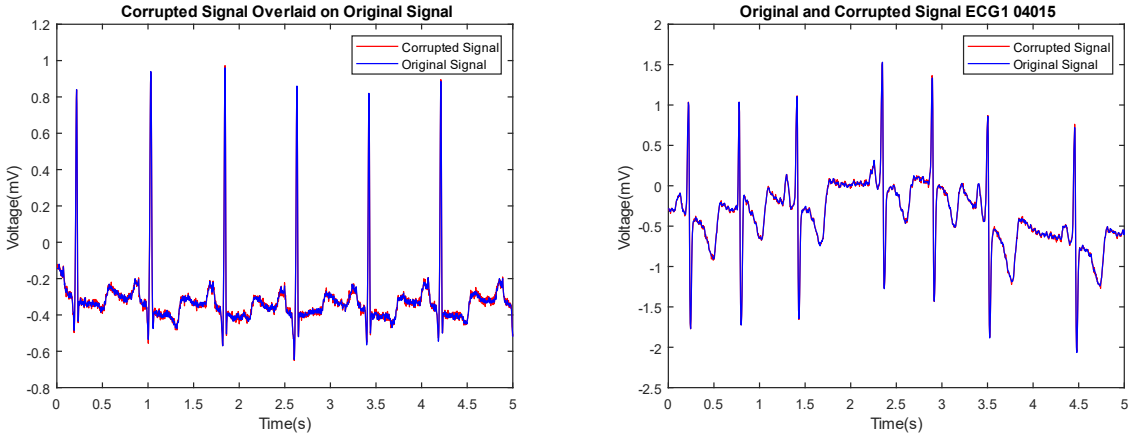


Fig. 16: Overlay of Corrupted and Original Signal for the Arrhythmia (left) and atrial fibrillation (right) corresponding to an SNR of 30dB.

Fig. 16 overlays the original signal with the noisy signal corresponding to the SNR of that in Fig. 15 , i.e. ≈ 29 - 30dB for the arrhythmia and fibrillation signals respectively.

4.3 Number of Channels (N_{CH}) vs Code Length(L)

The number of channels that can be shared using Walsh-Hadamard (WH) codes is straightforward as it is proportional to the length of the code. A WH code of length L gives L unique codes that are orthogonal to each other. Since the first code is an all '1' sequence, it is not useful, as no modulation occurs. Hence, the total number of channels that can be used for a sequence of length L is equal to $L - 1$. Since WH codes come in powers of 2, the total useful number of sequences is $2^n - 1$, where n is an integer greater than or equal to 2.

Alternatively, if the number of channels (N_{CH}) is already known or determined by another system parameter, the minimum length of the WH code required is the closest power of 2 that is greater than N_{CH}. Table 1 summarizes the minimum code length required depending on the desired number of channels to be shared.

The fact that WH codes are in powers of 2 can sometimes be disadvantageous when the number of channels is fixed. For instance, if 64 channels are required to be shared, a WH code of length of 128 must be chosen even though the remaining 64 codes are not required. A WH code that is much longer needs to either be generated or stored, taking up more computation power or area on a chip. Additionally, the frequency of modulation also increases with increasing length, so as to obtain a better performance. The frequency dependance on the code length is explained in Section 4.4.

Table 1: Number of Channels vs Minimum Code Length

Number of channels to share (N_{ch})	Minimum Required WH Code Length (L)
$2 \leq N_{CH} \leq 3$	$2^2 = 4$
$4 \leq N_{CH} \leq 7$	$2^3 = 8$
$8 \leq N_{CH} \leq 15$	$2^4 = 16$
$16 \leq N_{CH} \leq 31$	$2^5 = 32$
$32 \leq N_{CH} \leq 63$	$2^6 = 64$
$64 \leq N_{CH} \leq 127$	$2^7 = 128$

4.4 Modulation Frequency

The choice of frequency at which the codes run is determined by the bandwidth of the signals, and the number of channels being shared or the code length. Fig. 17 plots the SNR versus the modulation frequency for 3, 7 and 15 channels that are multiplexed with code lengths of 4, 8 and 16 respectively. It can be seen that for fewer number of channels, the modulation frequency required to achieve a certain SNR is much lower than those that multiplex more channels. This is because as the number channels increases, the length of the code required increases. This in turn increases the modulation frequency to achieve a similar SNR.

As the modulation frequency for a given length increases, the SNR improves up to a certain point after which the curve flattens out and the improvement in SNR is minimal. This stems from the fact that spreading the spectrum indefinitely is of no use as the fundamental limit to the achievable SNR is thermal noise. Spreading the spectrum does not eliminate thermal noise as it is white noise. During the modulation phase, the thermal noise present is up-modulated along with the signal and it is then down-modulated during the demodulation phase. Hence, an improvement in SNR can only be achieved by lowering the thermal noise in-band.

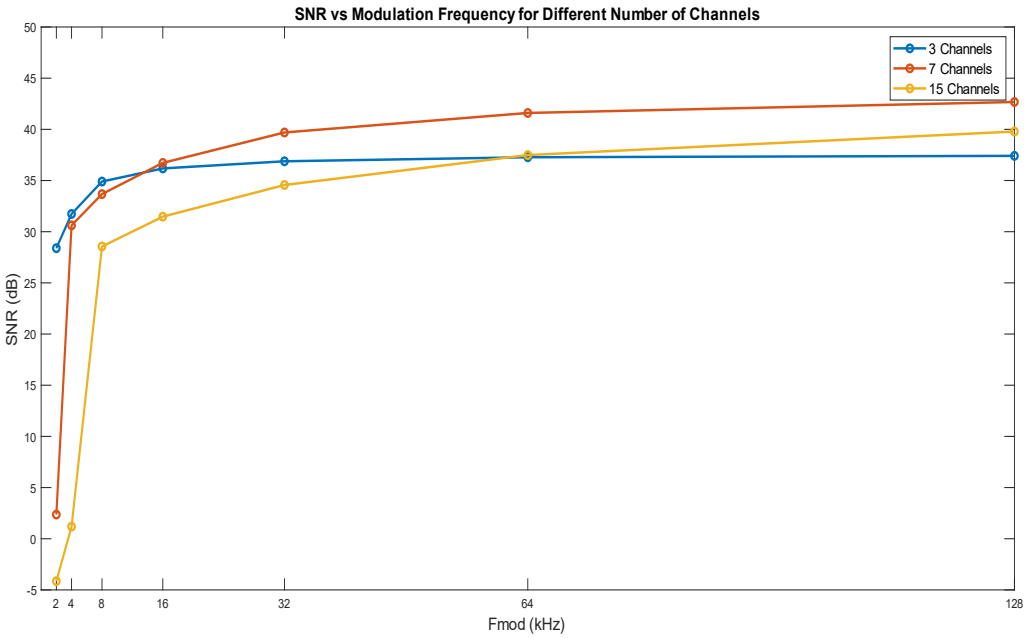


Fig. 17: SNR vs Modulation Frequency for Different Number of Channels

From the above plot, to achieve a resolution of about 5 bits or more, the modulation frequency for 3 channels should be greater than 2kHz, for 7 channels should greater than 4kHz, and for 15 channels should be greater than 16kHz.

4.5 System Block Diagram

Having quantitatively derived the required resolution, type of code and length required along with the corresponding modulation frequency, the system level design can now be constructed. The design is split into 2 blocks – the *On-Chip* recording circuitry and the *Off-Chip* recovery and processing unit that uses digital signal processing.

The On-Chip recording circuit consists of 3 main circuit blocks. The first block consists of a switching modulator for each channel. Each channel is modulated with a unique sequence that is generated by the code generator circuit on chip. The WH Code Generator, shown in Fig. 18, generates the codes that are fed into a non-overlapping clock generator. These non-overlapping signals drive the switches of the modulator. The second block is a summing amplifier that sums up all the channels. Finally, the summed output is digitized using a 6-bit Nyquist rate ramp ADC before being transmitted off-chip for processing and recovery. A ramp ADC is used due to its simplicity of implementation given the low resolution and frequency of operation.

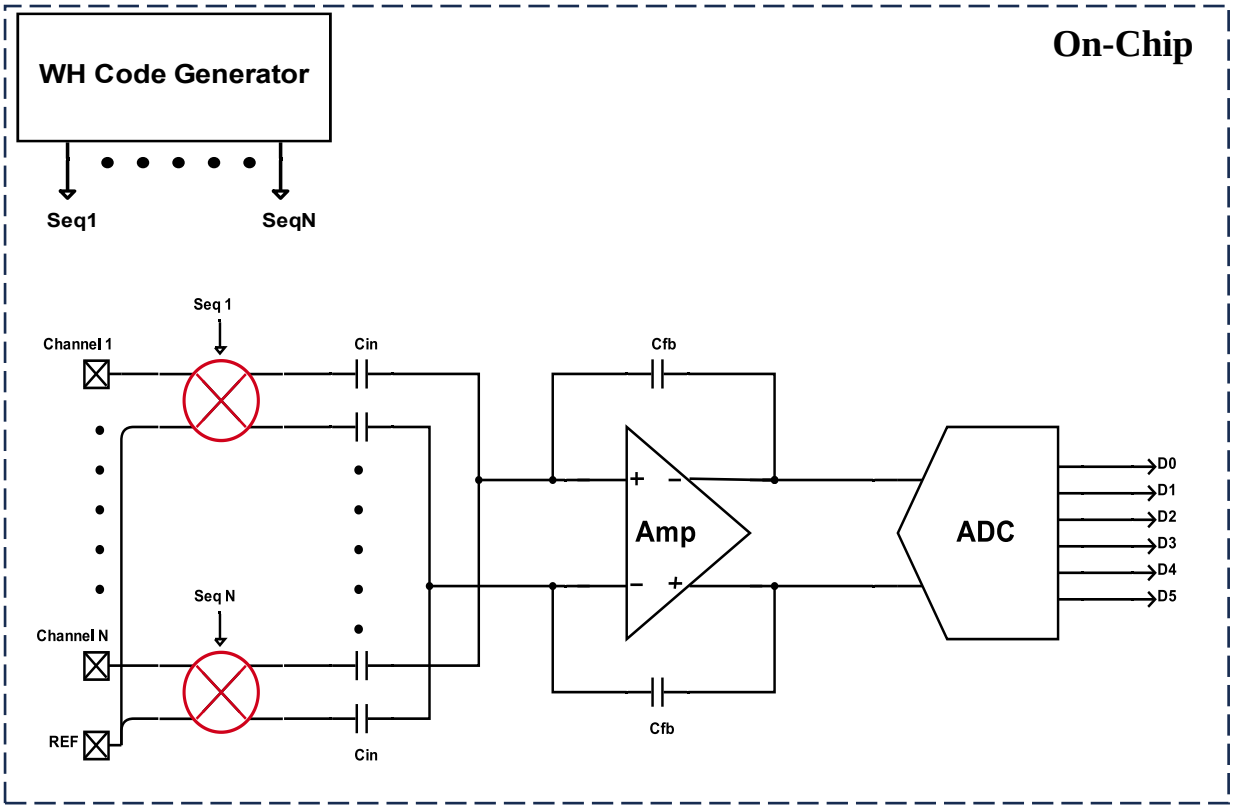


Fig. 18: On-Chip System Level Block Diagram

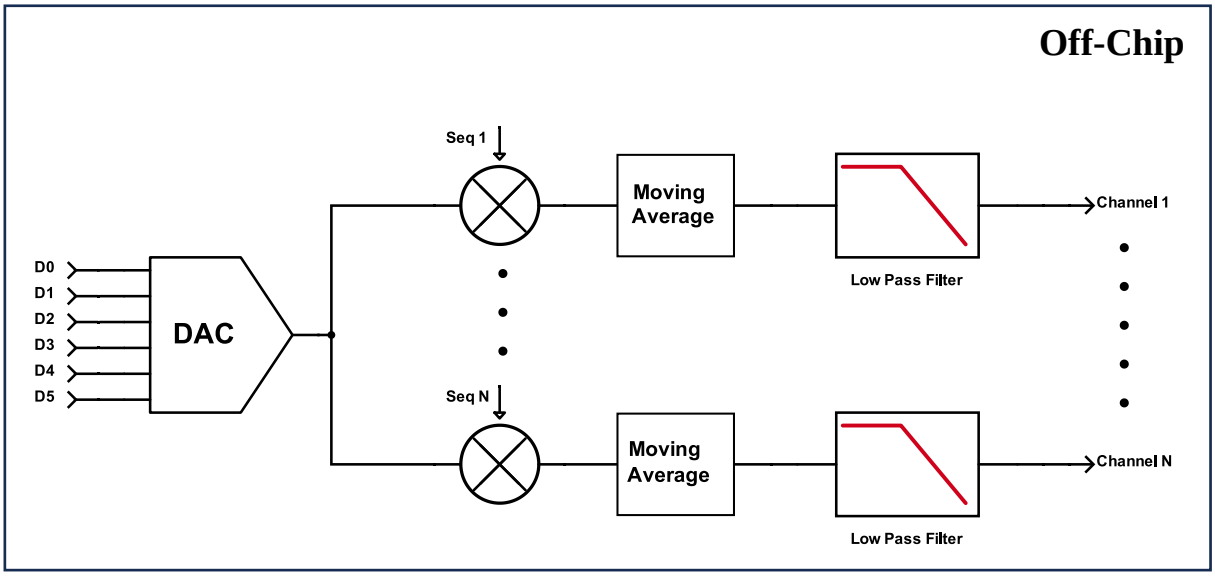


Fig. 19: Off-Chip Signal Recovery using Digital Signal Processing

The entire off-chip digital processing and signal recovery is done using MATLAB. The transmitted digital bits are converted back to an analog signal using an ideal DAC. The reconstructed analog signal is demodulated by multiplying this single output analog signal with all the N sequences running at the same modulation frequency that was used at the transmission end. The channels that are not synchronised cannot be recovered and look like noise as explained in Chapter 3. Thus, each channel only correctly recovers its own channel's data given that the synchronisation is fairly accurate. At this stage, the signal from each channel can be filtered to remove all the unwanted signals and noise. A digital LPF in MATLAB with a 6^{th} order roll-off is used. A moving average is also used for additional noise reduction and smoothening of the signal. The moving average filter allows for another 2-3dB improvement in SNR, thereby allowing for an additional half a bit of resolution without the requirement for additional bits or oversampling on-chip.

4.6 System Level Verification for 7 Shared Channels

Using the above derived specifications, the system is modelled using MATLAB and Simulink for 7 shared channels to verify the idea and specification derived above. Simulink is used as it allows for quick characterization due to the availability of in-built signal processing as well as circuit blocks that can be interfaced with ease.

The setup of the system in Simulink is shown in Fig. 20. The input channels contain single tone signals of different frequencies. The in-built Walsh-Hadamard Code Generator Block generates the WH code with parameters such as frequency and code length set by the user. In this case, the code operates at 4kHz with a code length of 8, as obtained from Sections 4.3 and 4.4. These modulated signals are summed and amplified using a summation and gain block. A low pass-filter with a tunable bandwidth is inserted to model the finite bandwidth of the amplifier. The amplified analog output is then digitized by using a simple quantizer and the digital output data is exported to MATLAB for signal recovery.

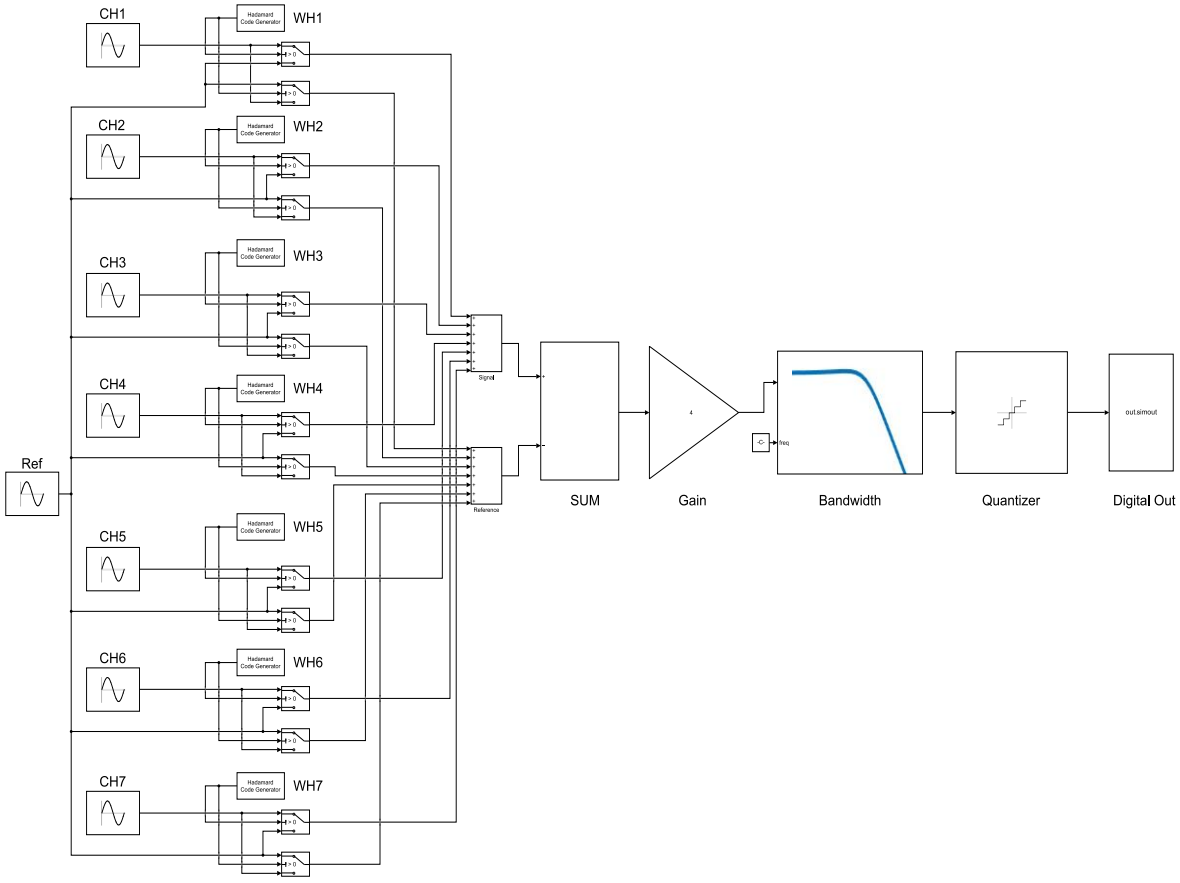


Fig. 20: Simulink Model of System for 7 Shared Channels

The seven inputs are single tone test signals - each with a unique frequency and an amplitude of $10mV_{pk\text{-}pk}$. The test signals are chosen as prime numbers to avoid any of its harmonics falling into the signal bin of another signal tone. This is to avoid any exaggeration or degradation of the SNR, thereby giving a wrong enhancement or degradation of the system level performance. The 7 signal frequencies are 13Hz, 17Hz, 23Hz, 29Hz, 37Hz, 41Hz and 47Hz, labelled CH1 to CH7 respectively as shown in the time domain plot in Fig. 21. Fig. 22 plots the frequency spectrum of the input signals.

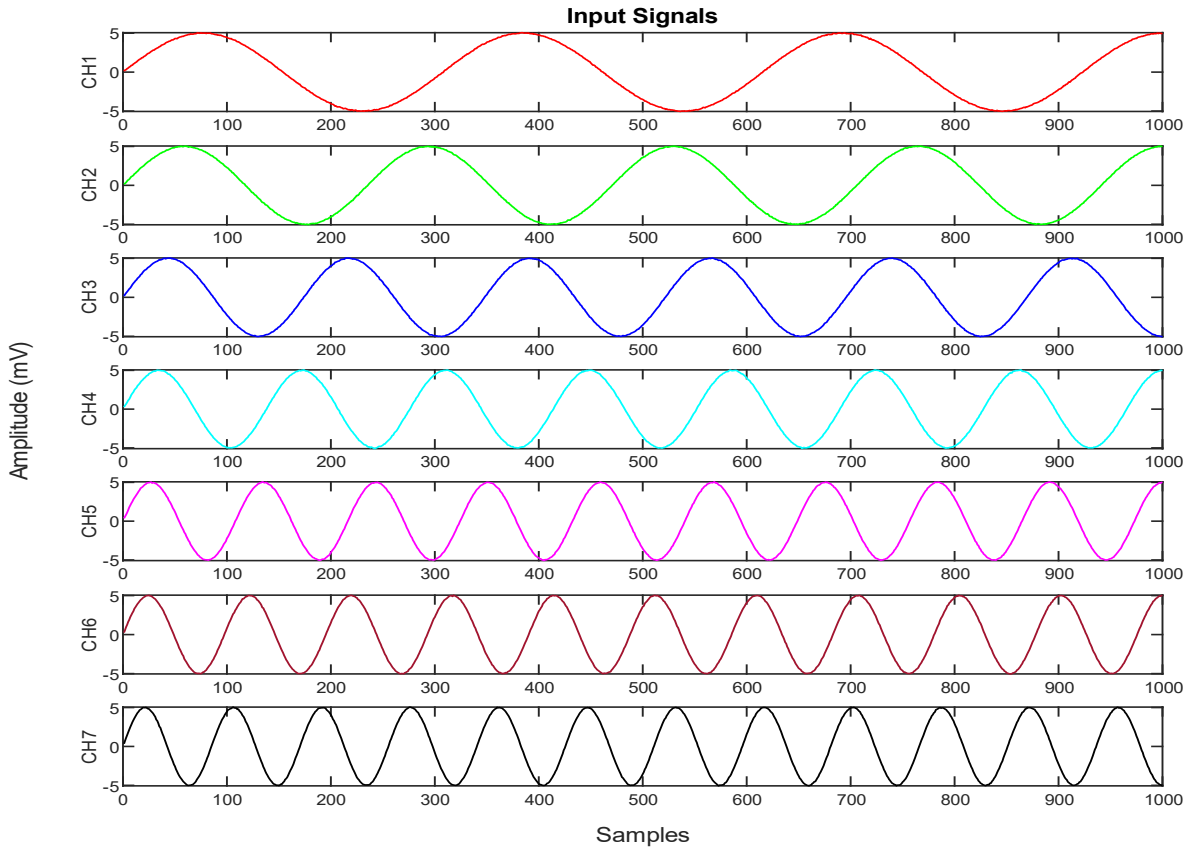


Fig. 21: Time Domain Waveforms of the 7 input signals, each with a unique frequency.

The transmitted output data bit stream is converted back to an analog signal using an ideal DAC, multiplied once again with the WH codes at the same frequency and then low pass filtered. The demodulated signals are shown in Fig. 23. It can be seen that there is some high frequency content that sits on top of the lower tone signals despite a 6^{th} order LPF being used.

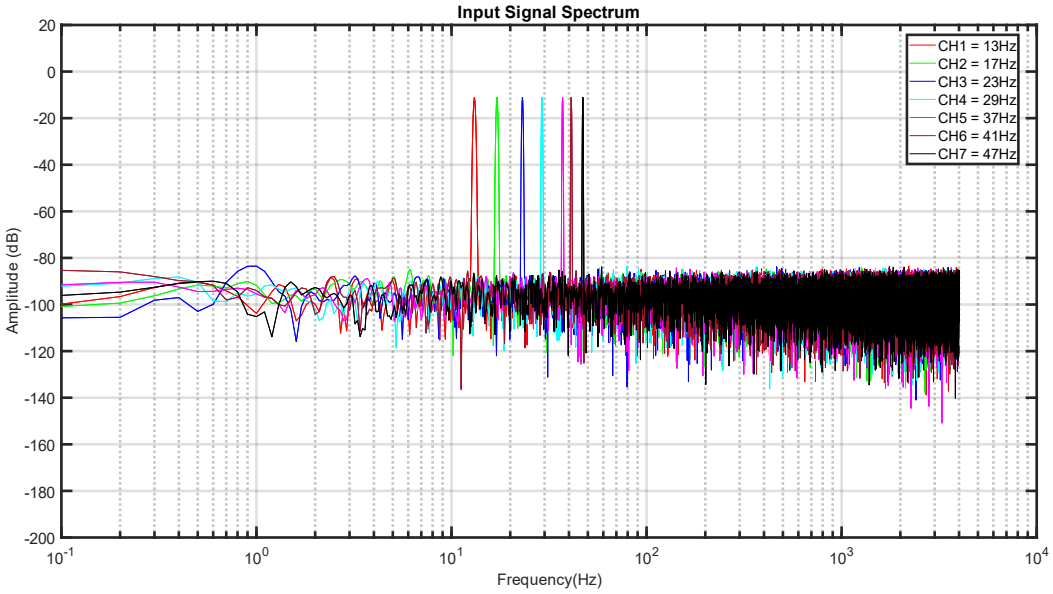


Fig. 22: Frequency Spectrum of Input Signals

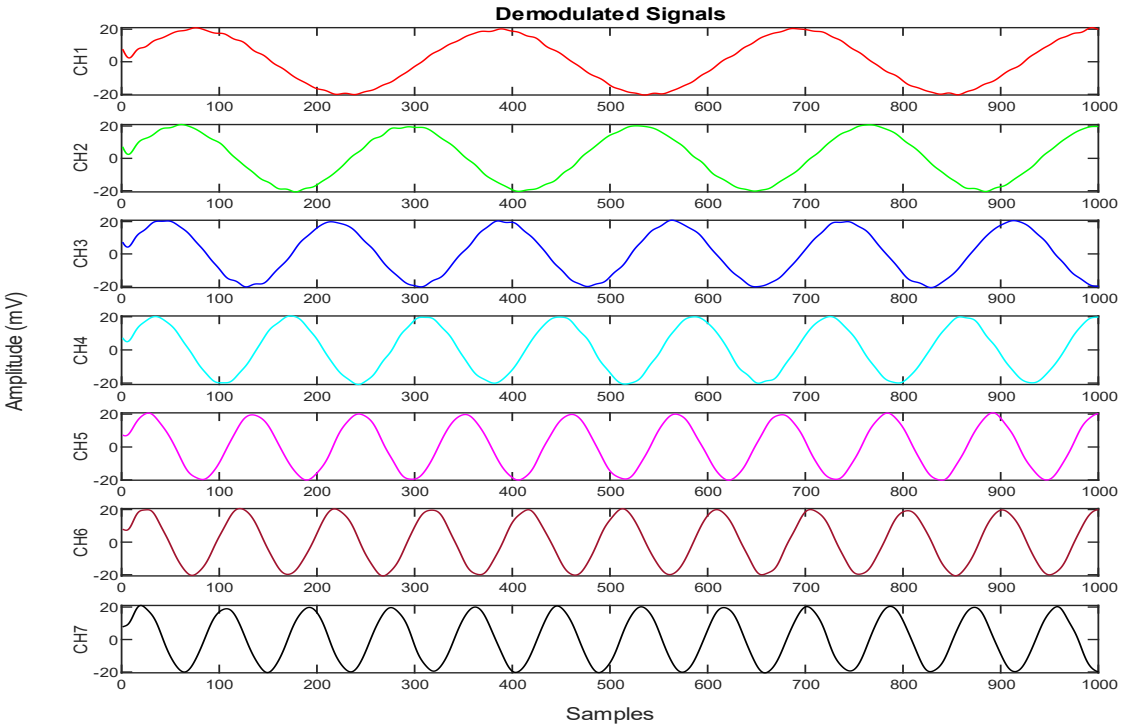


Fig. 23: Demodulated signals without moving average.

The higher order frequency content can be smoothened out by applying a moving average. On performing the moving average, an improvement of SNR of about 2-3dB is noticed. Comparing the time domain waveforms of Fig. 23 and Fig. 24, it can be seen that the higher frequency

ripples have been reduced. It must be noted that the window of averaging must also be chosen carefully so that critical information is not lost by aggressive filtering. Table 2 provides a comparison of each channel and its SNR after demodulation with and without a moving average for the 7 channel system.

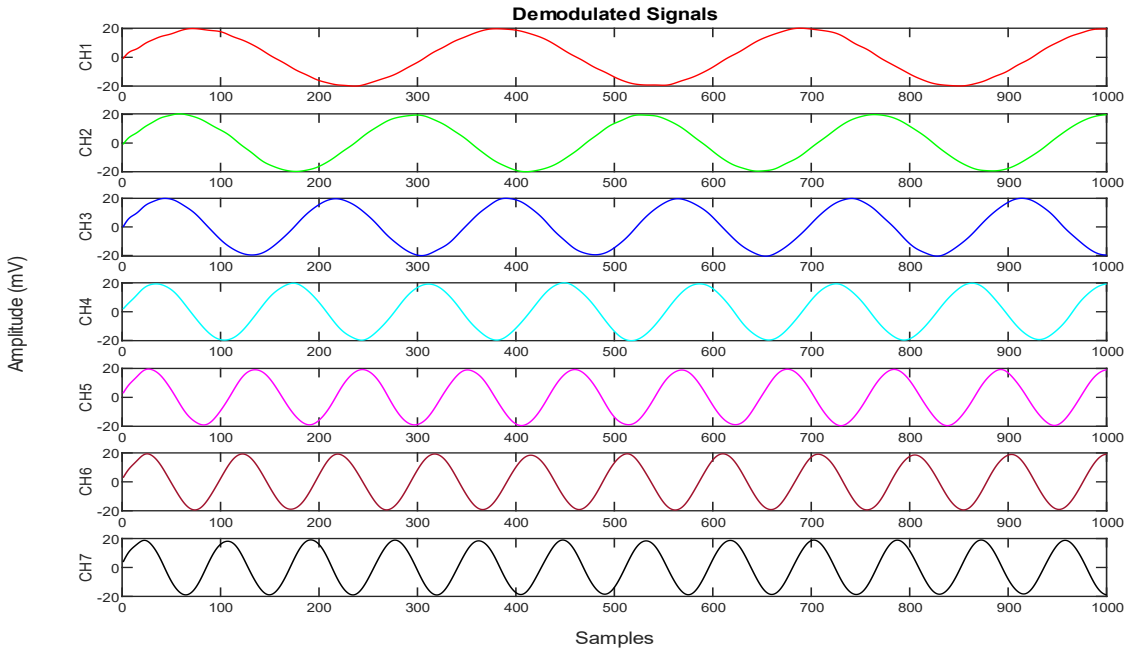


Fig. 24: Demodulated Signal with Moving Average of 16 Samples each.

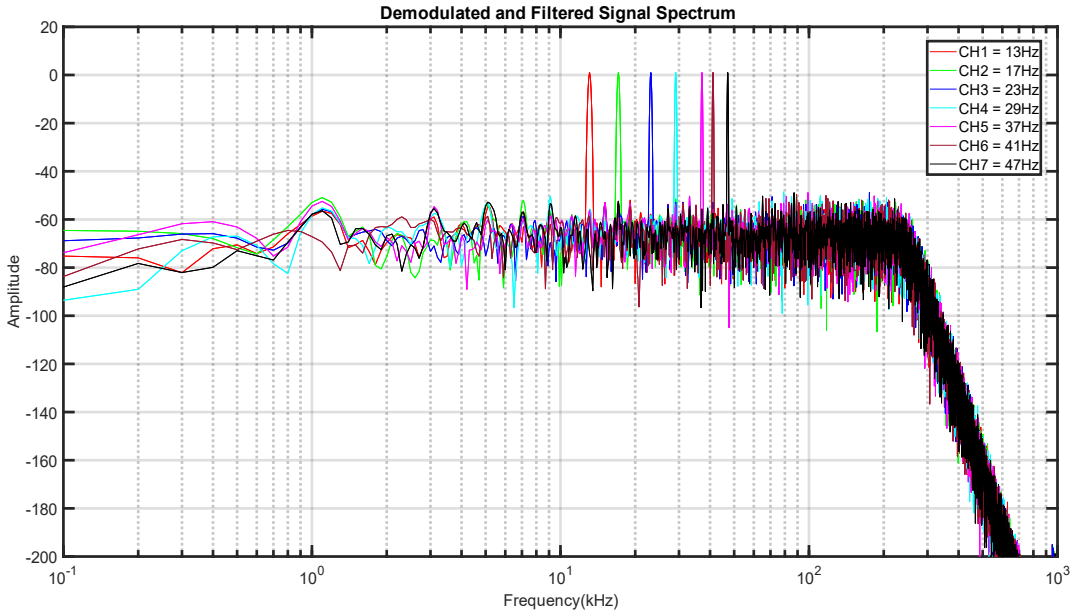


Fig. 25: Frequency Spectrum of Demodulated Signals

Table 2: Comparison of SNR with and without a moving average for 7 channels at F_{MOD} = 4kHz.

Channel	Frequency (Hz)	SNR without averaging (dB)	SNR with averaging (dB)
1	13	31.045	34.351
2	17	31.509	34.368
3	23	30.870	34.260
4	29	31.392	34.202
5	37	31.492	34.506
6	41	31.448	34.493
7	47	31.376	34.177

In Fig. 24, Fig. 25 and Table 2 we observe that the system level specification derived in Sections 4.2, 4.3 and 4.4 have been verified by the successful recovery of each of the signal tones with sufficient resolution.

4.7 System Level Specifications

From the system level parameters that were derived in the above sections, the required circuit level specifications can be deduced. Table 3 summarizes the required specifications.

Table 3: Design Specifications

Parameter	Description	Value
Number of Channels (N_{CH})	-	7
Input Voltage Range	(1-10mV_{pk-pk}) * 7	7-70mV_{pk-pk}
Output Voltage Swing	-	150mV
Modulation Frequency (F_{MOD})	Optimum for 7 channels	4kHz
Resolution (N)	6 bits	6 bits
Sampling Frequency (F_S)	> F_{MOD}	16kHz
Channel Gain	4 Gain Settings (3,6,12,24)	9.5 – 27.6dB

Parameter	Description	Value
THD	< 5%	< 26dB
Gain of Amplifier	> THD + Channel Gain	> 53dB

In the next chapter, a novel method for generating the WH codes on-chip using simple digital logic is proposed and elaborated upon. Using the code generator and the system level parameters verified in this chapter, a 7 channel multiplexed system is realised and the circuits implemented are explained in Chapter 6.

Chapter 5

Walsh-Hadamard Code Generator using Digital Logic

5.1 Proposed Walsh-Hadamard Code Generator Circuit

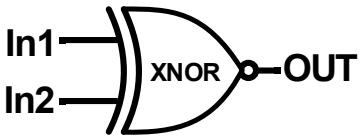
One of the concerns/limitations with orthogonal codes such as Walsh-Hadamard codes is that they are typically either stored on chip using some memory [6], or implemented by computing the matrices on-chip, which then requires a processor. Storage of large sets of Walsh-Hadamard codes poses a constraint on area as the look-up table scales exponentially with the length of the sequence. Implementing a processor requires large power consumption while at the same time occupying more area with a higher implementation complexity to embed the processor on chip. This stems from the way in which these codes are constructed mathematically by taking the Kronecker products as was explained in Section 3.2. However, looking at the sequences in terms of frequency and time-shifts, these codes can be constructed using extremely simple digital logic accurately, as they can all be derived from a single clock signal. This novel method of mapping the WH sequence onto silicon using digital circuits is elaborated with an example below.

Prior to explaining the WH code generation methodology, two fundamental logic functions from digital circuits are explained to better understand how the pattern was devised and came about. The two logic functionalities are the *XNOR* logic gate and a *Frequency Divider* using a flip-flop. The former is a combinational logic circuit whereas the latter is a sequential logic circuit. The combination of these two principles will allow an accurate generation of the WH codes that is derived from the clock signal.

The first useful logic gate is the XNOR. Table 4 shows the truth table for a 2-input XNOR gate along with its schematic symbol. The key point to notice from this table is that when one of the inputs is a logic ‘0’ (In1), the output is an inverted or flipped version of the other input (In2) . On the other hand, when one of the inputs is a logic ‘1’ (In1), the output follows or copies the other input (In2).

Table 4: XNOR Logic Function and Symbol

In1	In2	OUT	Operation
0	0	1	Invert In2
0	1	0	
1	0	0	Copy In2
1	1	1	



The second useful digital logic circuit is the Frequency-Divider circuit using a simple D Flip-Flop (DFF). The idea stems from the fact that a D Flip-Flop is a clocked device whose output follows the input only on transition of one of the clock edges. Hence, if the input of the DFF is a complementary version of the original clock, the output frequency will be half of the input clock frequency. This can be achieved by feeding back the complementary output of the DFF to the input, which is best visualized from Fig. 26. Using these two concepts, a WH code of any length can generated using solely digital logic.

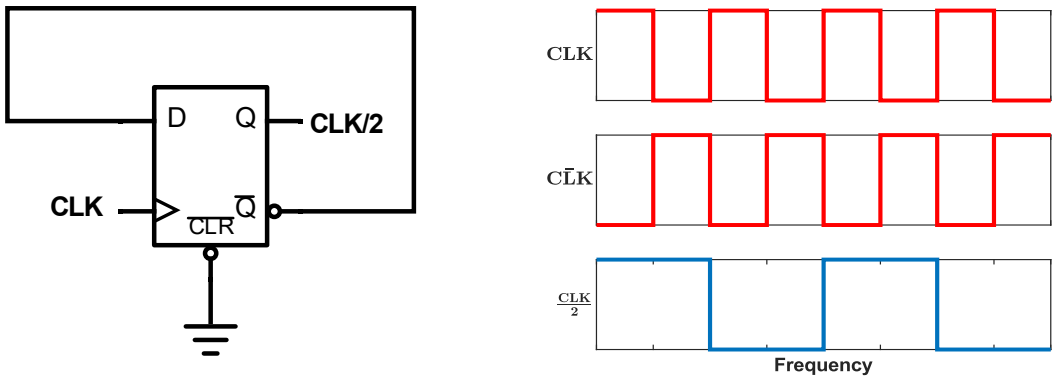


Fig. 26: Frequency Division by 2 using a D Flip-Flop

In Section 3.2, the mathematical construction of WH matrices was derived. It was seen that by computing the recursive Kronecker product of 2 matrices, a matrix of size N is achieved.

The algorithm proposed here works on the premise of constructing a sequence of length 2N given an existing sequence of length N. This has just 1 implication – an initial condition or sequence is required to start the computation. This is fairly easy to achieve, as can be seen from the 4x4 matrix below. The first row is just a logic '1' which can be tied to the power supply. This sequence is also not useful as there is essentially no modulation performed by this sequence. The second row is an alternating sequence between '1' and '0'. This sequence resembles a clock signal, and since a clock is anyway required, one of the system clocks or a fraction of it can be directly used for this sequence.

$$H_{4_{1,0}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Starting from a base matrix where the first row is just '1' and the second row is a clock sequence, the remaining rows can be constructed using the following algorithm. The base matrix is of length N and the matrix to be constructed is of length $2N$.

Construction of $2N$ Length Sequence from N Length Sequence:

1) Row 1 to N $\rightarrow$ Sequence Repeats
2) Frequency of Row $N + 1 \rightarrow \frac{\text{Frequency of Row } \left(\frac{N}{2}\right)+1}{2}$
3) Row $N + x \rightarrow$ Row $(N + 1)$ **XNOR** Row x $\quad$ *where* $\quad 2 \leq x \leq N$

An example to depict the working of this algorithm is given below. A WH code of length 8 is constructed from a WH code of length 4. Here, $N = 4$, $\frac{N}{2} + 1 = 3, N + 1 = 5$, $2N = 8$ and $2 \leq x \leq 4$.

Table 5 shows how the first 5 sequences are derived and Table 6 computes the remaining 3 sequences. The corresponding colour codes depict the operation performed. Following the algorithm, it can be seen that from Rule 1, the first 4 rows simply repeat and do not require any further computation. Following Rule 2, the 5th row is half the frequency of the 3rd row. The

halving of the frequency can be implemented using the frequency-divider circuit explained above.

Table 5: WH Code Pattern of First N+1 Sequences

Seq No. (4x4)	WH Code 4x4	Pattern	WH Code 8x8	Seq No. (8x8)
$Seq\ 1$	1 1 1 1	Repeats →	1 1 1 1 1 1 1 1	$Seq\ 1$
$Seq\ 2$	1 0 1 0	Repeats →	1 0 1 0 1 0 1 0	$Seq\ 2$
$Seq\ 3$	1 1 0 0	Repeats →	1 1 0 0 1 1 0 0	$Seq\ 3$
$Seq\ 4$	1 0 0 1	Repeats →	1 0 0 1 1 0 0 1	$Seq\ 4$
$Seq\ 3$ $\left(\frac{N}{2}+1\right)$	1 1 0 0	Half the Frequency →	1 1 1 1 0 0 0 0	$Seq\ 5$ $(N+1)$

Table 6: WH Code Patterns of Remaining Sequences from N+2 to 2N

Seq No. (8x8)	WH Code 8x8		Pattern	WH Code 8x8		Seq No. (8x8)
	Repeat	Flip		Repeated	Flipped	
$Seq\ 2$ $Seq\ 5$	1 0 1 0 1 1 1 1	1 0 1 0 0 0 0 0	$Seq\ 2$ **XNOR** $Seq\ 5$	1 0 1 0	0 1 0 1	$Seq\ 6$ $(N+2)$
$Seq\ 3$ $Seq\ 5$	1 1 0 0 1 1 1 1	1 1 0 0 0 0 0 0	$Seq\ 3$ **XNOR** $Seq\ 5$	1 1 0 0	0 0 1 1	$Seq\ 7$ $(N+3)$
$Seq\ 4$ $Seq\ 5$	1 0 0 1 1 1 1 1	1 0 0 1 0 0 0 0	$Seq\ 4$ **XNOR** $Seq\ 5$	1 0 0 1	0 1 1 0	$Seq\ 8$ $(N+4)$

Finally, the remaining $N-1$ rows follow Rule 3. By observing the two halves of the code, it can be seen that the *first half* of the sequence is repeated from Sequence $N+x$ and the *second half* is flipped. This property can be implemented using the XNOR function as described above, since Sequence 5 has a sequence of '1's followed by a sequence of '0's. The 6th row is XNOR of Sequence 2 and Sequence 5, 7th row the XNOR of Sequence 3 and Sequence 5 and 8th row

is the XNOR of Sequence 4 and Sequence 5. Fig. 27 shows the waveforms of the sequences built using the proposed algorithm for a 8x8 WH Sequence using ideal digital logic gates in LTSpice. The first signal on top of the figure is the clock signal followed by the 7 different sequences. Each symbol of the sequence corresponds to one period of the clock signal.

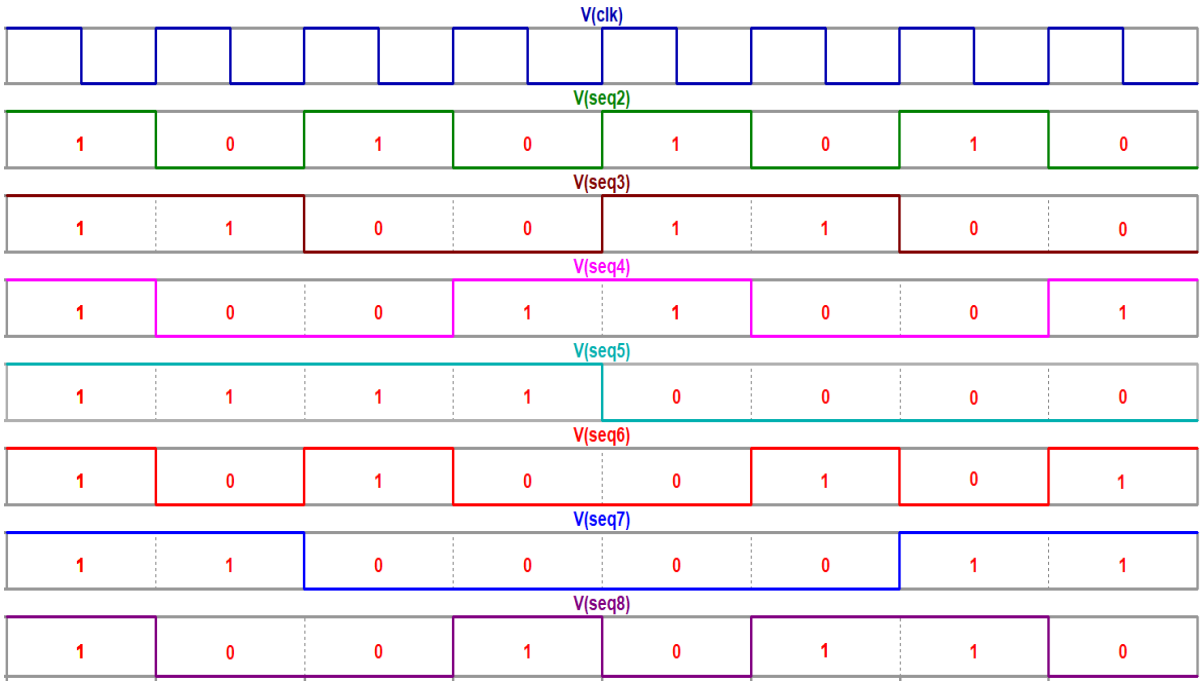


Fig. 27: WH8 Code Waveforms using the proposed circuit that solely uses digital logic.

This methodology or algorithm extends to all the powers of 2. In this thesis, the algorithm was verified for sequence lengths ranging from 2 to 128 in powers of 2. Since building and verifying the circuit for lengths greater than 16 is tedious, an automation script in MATLAB is written to write a netlist into LTSpice and extract the simulation results into MATLAB for comparison. The automation script that does this operation is provided in Appendix A.1.3.

5.2 Area Savings of Proposed WH Code Generator

The main benefit of using this methodology to generate a WH sequence on-chip is that it uses very few gates, which results in enormous savings in area. The area savings on-chip with the newly proposed circuit is quantified in this sub-section. Apart from the proposed algorithmic circuit being extremely simple and functional, the circuitry used to realise the code generator is orders of magnitude less in terms of number of gates used and the area occupied when

compared to a look-up table (LUT), which is typically how WH codes are used on chip. A LUT uses a D flip-flop cell to store each bit of the code [6]. Thus, an N bit code requires N flip-flops for each code sequence of the entire set. Given that $N-1$ codes of the WH sequence are used, this corresponds to a total array size of $N*(N-1)$ for the LUT.

On the other hand, for a sequence of length N, using the proposed algorithmic circuit:

i. Number of Flip-Flops Required $= \log_2(N)$
ii. Number of XNORs required = $[N - \log_2(N)] - 1$

The area used by the LUT and proposed circuit area are calculated using the cell size of a flip-flop and XNOR gate. In the technology process used in this project, a single D Flip-Flop occupies roughly 70µm^2, while a single XNOR gate occupies about 43µm^2. These measurements from the layout are used to make a rough estimate of the area occupied by the circuit itself, not accounting for aspects such as placement and routing on an actual chip. The calculation solely uses the area of the respective cells layout as provided by the foundry.

Area $_{\text{LUT}}$ = Area $_{\text{D flip-flop}} * [N*(N-1)]$

Area $_{\text{Proposed Circuit}}$ = [Area $_{\text{D flip-flop}} * \log_2(N)$] + [Area $_{\text{XNOR}} * \{N - \log_2(N) - 1\}$]

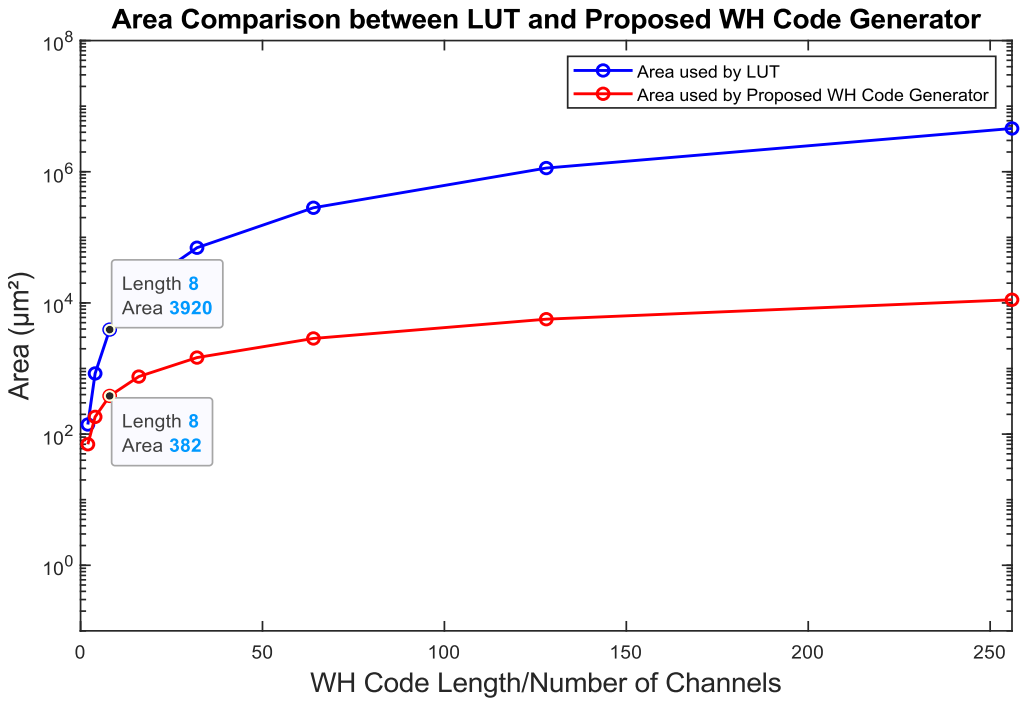


Fig. 28: Area Comparison between LUT and Proposed Circuit

Fig. 28 plots the area of the LUT and proposed circuit across different code lengths from 2 to 256. For shorter code lengths of 2 and 4 the area savings are minimal and there is no significant difference between a LUT and the proposed circuit. However, from a code length of 8 or greater, the area reduction begins to show sufficient improvement. For instance for a code length of 8 the proposed circuit is about 10 times smaller than using a LUT. The improvement is even more staggering for much longer code lengths where the area savings is exponential. For instance, for a code length of 128, the area is 200 times smaller as shown in Fig. 29.

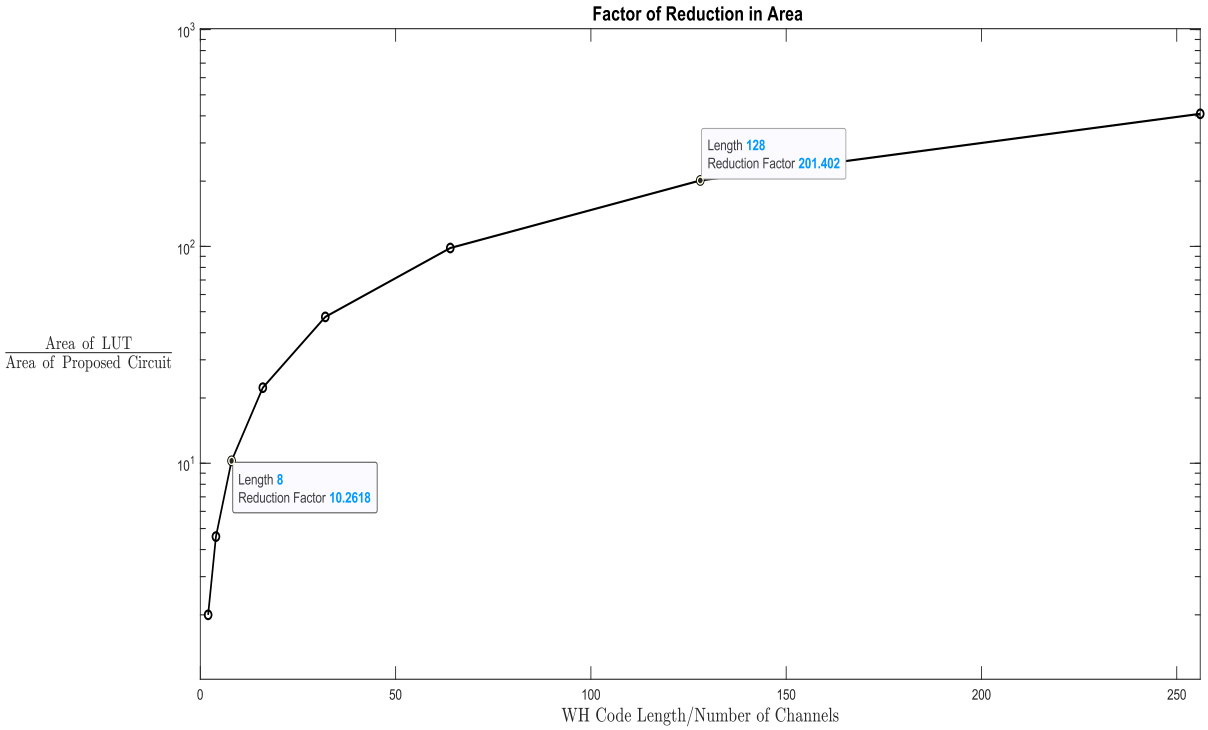


Fig. 29: Factor of reduction in area for different code lengths.

The enormous reduction in area for longer sequences, makes the WH code generator circuit comparable in area to that of LFSRs that are used to generate PRBS sequences as explained in Section 3.1. In fact, the circuit is simpler and uses fewer gates when compared to an LFSR. The reduction comes from the fact that a single flip-flop or XNOR gate generates each of the different sequences in a set. In contrast, LFSRs require a few flip-flops along with some combinational logic gates in feedback to generate each pseudo-random sequence as shown in Fig. 12. Since the area of the WH code generator is now in the same order of magnitude of an LFSR, it is more beneficial to use a WH code with the proposed circuit when the main

requirement of the multiplexing is to reduce the cross-correlation between channels. Further, WH codes use a much smaller bandwidth when compared to PRBS codes [5]. This is because for good cross-correlation properties, PRBS codes of much longer periods must be used. A code with a longer period requires a longer modulation frequency as explained in Section 4.4. Thus, when cross-correlation and efficient use of bandwidth is a key requirement for multiplexing, and the area is constrained, a WH on-chip code generator is a much better choice.

Table 7: Area Comparison between LUT and WH Code Generator

Sequence Length	WH Code LUT Area (μm^2)	WH Code Generator Area (μm^2)	Area Reduction Factor
2	140	70	2
4	840	183	4.59
8	3920	382	10.26
16	16800	753	22.31
32	69440	1468	47.30
64	582240	2871	98.31
128	1137920	5650	201.40
256	4569600	11181	408.69

Chapter 6

Circuit Implementation

The Walsh-Hadamard coded multiple channel system described in the previous chapters is designed and implemented using the 180nm BCD process by TSMC. Each circuit block with their properties and relevant specifications is described in the sub-sections below.

6.1 8 bit Walsh-Hadamard Sequence Generator

6.1.1 Schematic and Description

The Walsh Hadamard sequence generator proposed in Chapter 5, is constructed using the digital gates available in the standard logic cells library provided by TSMC. The schematic of the circuit is shown in Fig. 30. For the generation of 7 sequences, only 4 gates and 3 flip-flops are required.

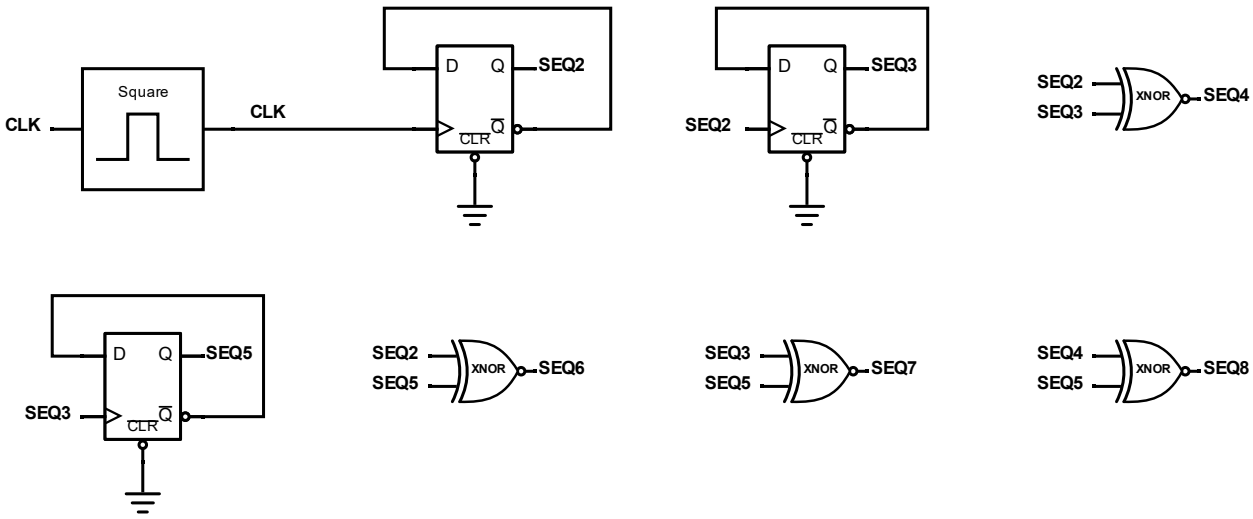


Fig. 30: Circuit Schematic of WH Code Generator

A test clock signal of 500kHz (period = 2µs) is used to validate the performance and check for correctness of the sequence generated. The frequency of the test clock is much higher than the

intended clock signal used to prove its robustness for higher frequencies as well. As the sequence of all ‘1’s is not used, it is neglected. Fig. 31 plots the waveforms of the sequences generated which match the waveforms from Fig. 27, where the algorithm was built using ideal blocks. However, there are certain spikes in Sequences 4, 6, 7 and 8 at certain instances of time which are undesirable. These spikes are a result of the XNOR operation performed when both the inputs are in transition from one logic state to another. For instance, in Sequence 4 at 4µs Sequence 2 is transitioning from ‘0’ to ‘1’ while Sequence 3 is transitioning from ‘1’ to ‘0’. The finite rise and fall times of the gates cause these glitches to occur. To fix the glitches, a negative-edge triggered flip-flop is connected to these outputs. This is done because the negative edge of the clock occurs during a steady-state output of the sequence as can be seen from Fig. 32. The resultant outputs match the outputs of the waveforms from Fig. 27 exactly without any glitches. This verifies the validity of the proposed approach on the circuit level.

6.1.2 Verification and Waveforms

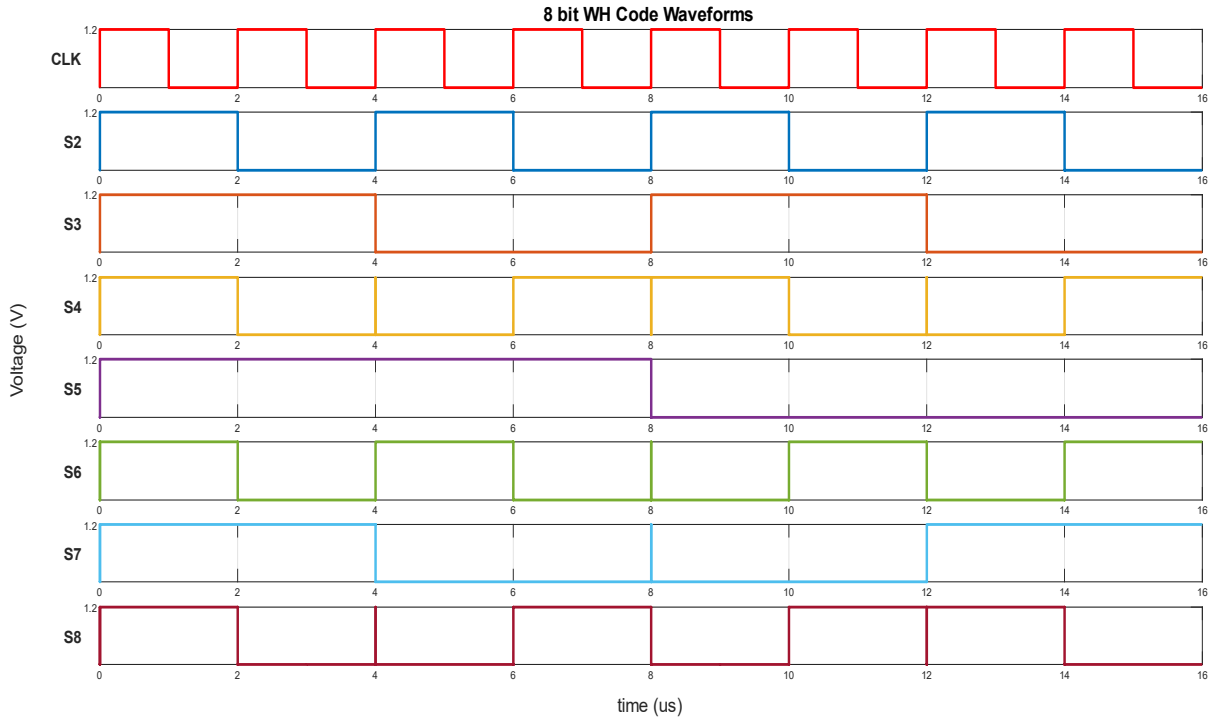


Fig. 31: WH8 Code Waveforms with glitches

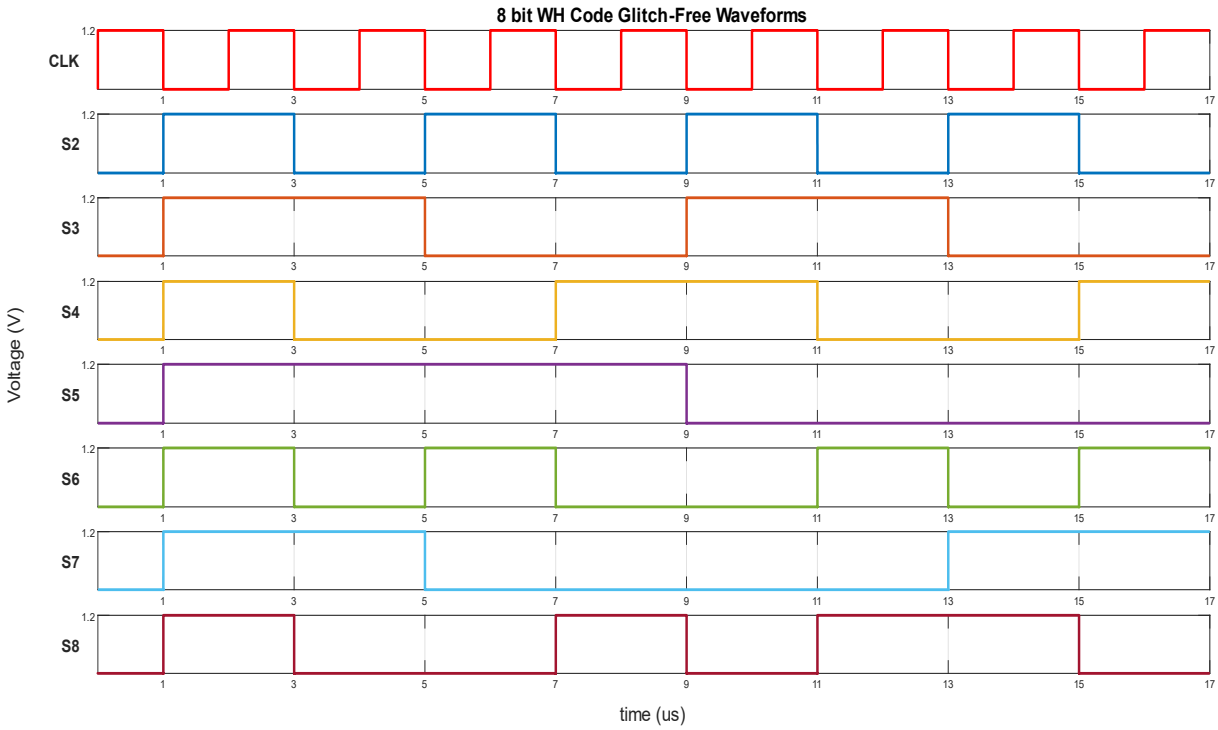


Fig. 32: WH8 Code Waveforms without any glitches.

The outputs of the sequence generator are fed as an input to a non-overlapping clock generator circuit as shown in Fig. 33, to drive the switches of the modulator.

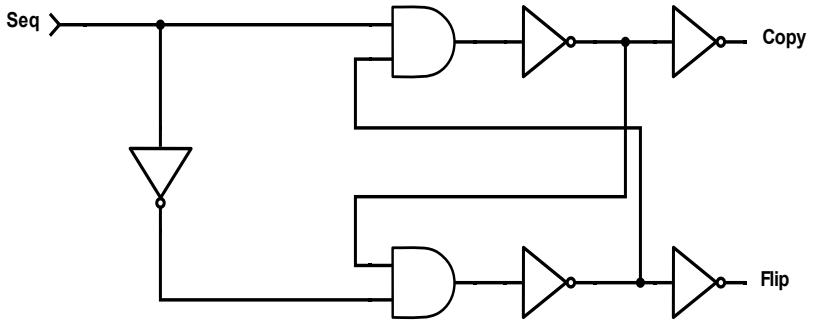


Fig. 33: Non-Overlapping Clock Generator

6.2 Amplifier

The top level schematic of the summing amplifier is depicted in Fig. 34. The sub-sections that follow explain the design strategy of the OTA and capacitors.

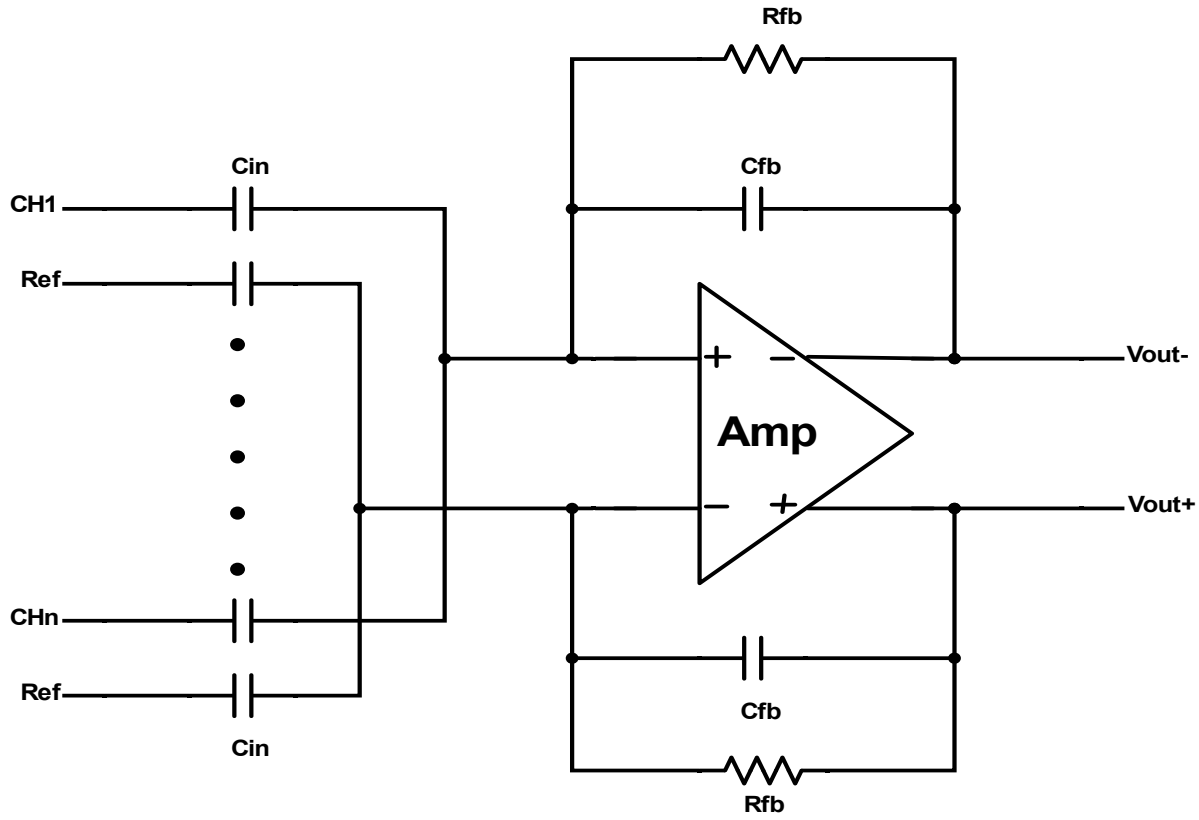


Fig. 34: Summing Amplifier Schematic

6.2.1 Schematic and Description of OTA

The operational transconductance amplifier implemented is a fully-differential inverter-based amplifier as shown in Fig. 35. It is alternatively called a current reuse topology as the input signal is fed into the PMOS and NMOS input terminals. This topology is used due to its high gain and g_m efficiency. Given the threshold of the transistors in this technology, by using a supply voltage of 1.2V, the concept of self-biasing can be used. This eliminates the requirement for external biasing circuitry as well as an additional common mode feedback circuit. Thus, it helps to reduce the power and area consumption while also simplifying the design.

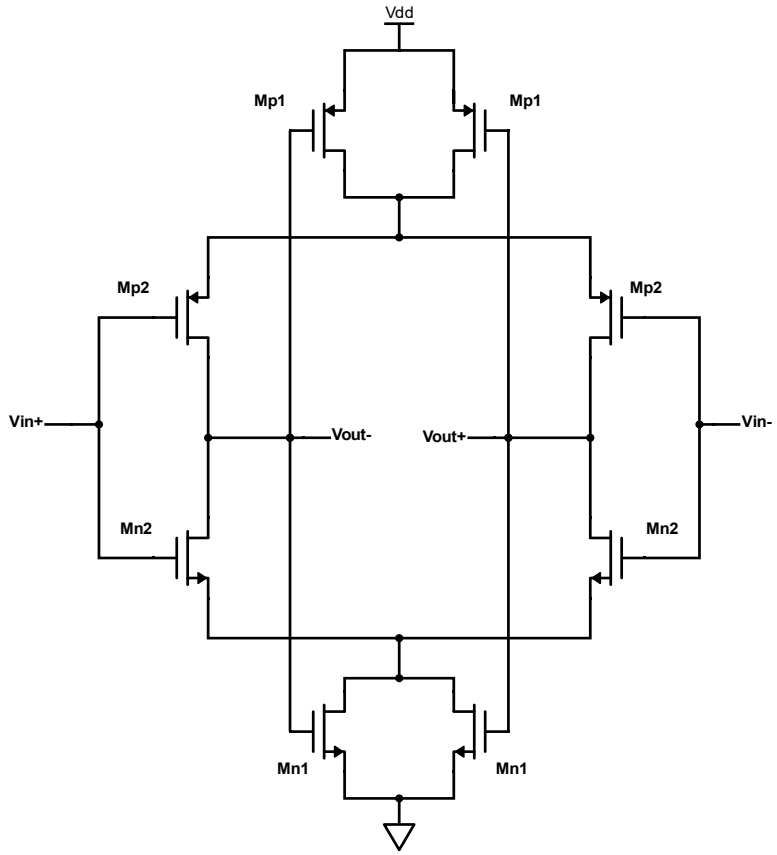


Fig. 35: Self-Biased Inverter based OTA

The voltage gain of the amplifier is given by:

$$A_V = -(g_{Mn} + g_{Mp})(r_{on} \,||\, r_{op})$$

(6.2.1)

The thermal input referred noise is:

$$V_{in}^2 = \frac{8\,K\,T\,\gamma}{g_{Mn} + g_{Mp}}$$

(6.2.2)

Thus, the integrated input noise is:

$$V_{In,Total} = \sqrt{\frac{8\,K\,T\,\gamma\,.\frac{\pi}{2}.BW}{g_{Mn} + g_{Mp}}}$$

(6.2.3)

From the above equations, it can be seen that the total gain and noise see an improvement because the effective g_m is increased as both the NMOS and PMOS pair contribute to the total g_m. Thus for the same noise specification, a lower current can be used leading to a better power efficiency. For proper output common mode control, the topology makes use of self-biasing where the output node is directly connected to the gates of the tail current sources to implement the common-mode feedback. While a single PMOS or NMOS tail current source can also be implemented, in this case since the total required output voltage swing is low, two tail current sources are used. The two tail current sources once again reuse the same current for a stronger negative feedback loop as the effective g_m in the feedback loop is the sum of the g_m of the PMOS and NMOS current source. Hence, for a given current using both the g_{ms} the common mode loop gain is larger which is beneficial in stabilizing the output common mode.

6.2.2 Design Strategy and Sizing

Before sizing the transistors, since the concept of self-biasing is employed, the constraints on the bias voltages must be noted. The desired output common mode is at 0.6V, which is half the supply voltage. As the input pair is biased through feedback using a resistor, the input common mode is set to 0.6V as well. Additionally, since the common mode feedback is performed by directly connecting the output nodes to the tail current sources, the gates of the tail current sources are at 0.6V too. Thus, the V_{GS} of the two tail current sources = 0.6V.

First, the size the signal transistors is chosen. Since a higher intrinsic gain requires a large output impedance longer transistors must be chosen to achieve sufficient gain. The design does not use cascodes to eliminate any external biasing. From Fig. 36 and Fig. 37 a length of 3µm is chosen for the NMOST and PMOST to achieve a gain of greater than 50dB. Putting the signal transistors in moderate or closer to weak inversion ensures a high g_m efficiency. Additionally, since the signal transistors and the current sources have the same gate bias, it is necessary for the signal transistors to be in weak inversion so that the bottom current sources can remain in saturation to achieve a good CMRR. As annotated in Fig. 36 and Fig. 37, for the NMOST a Gm/Id of 20 was chosen as going further causes a dip in the intrinsic gain. For the PMOST, a Gm/Id of 23 is chosen as it is the maximum point on the plot. From the current (Id) set by the tail sources, the lengths and the Gm/Id value, the widths of the transistor can be computed. Table 8, summarizes the widths and lengths of all the transistors of the amplifier.

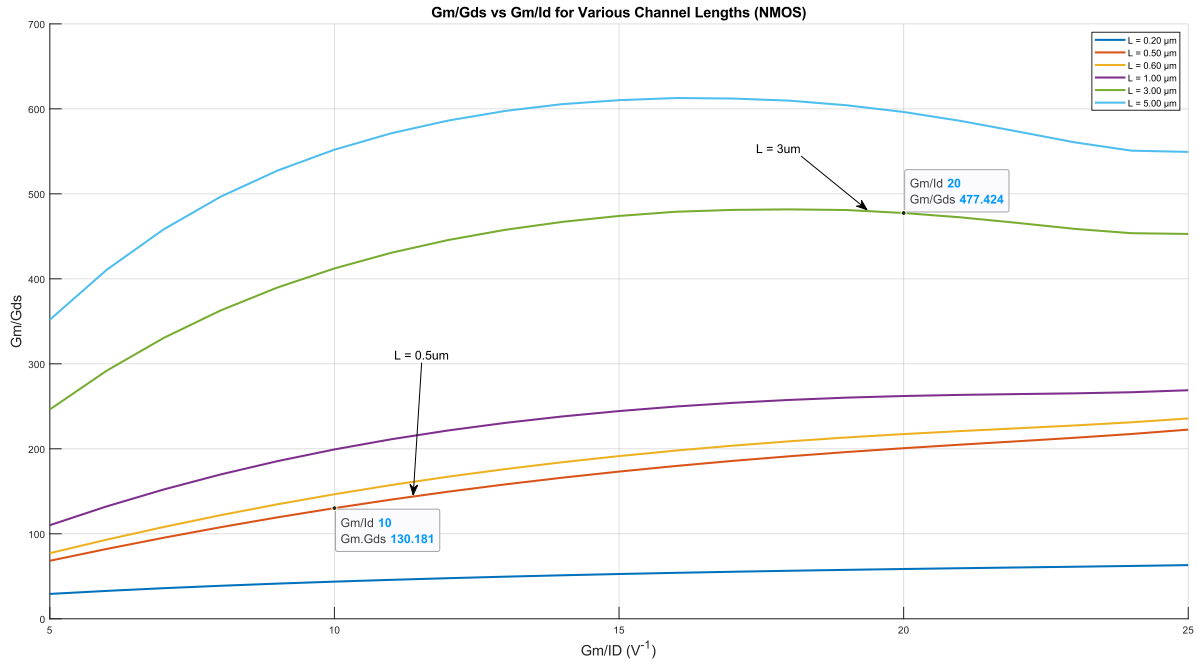


Fig. 36: Gm/Gds vs Gm/Id of NMOS transistor for different channel lengths

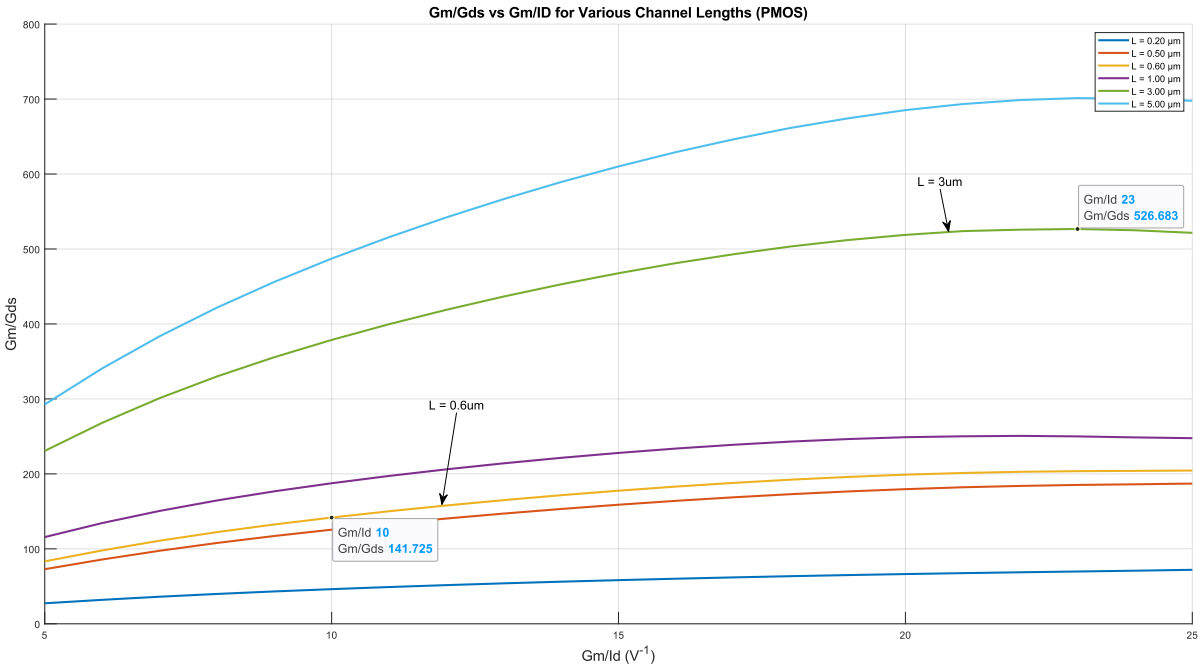


Fig. 37: Gm/Gds vs Gm/Id of PMOS transistor for different channel lengths

The next step is to size the two tail current sources. Once again, since the output impedance of the tail current sources determine the common mode rejection ratio, its length is chosen based on the output impedance. Longer transistors provide a larger output impedance but at the same

time using longer transistors also increases the $V_{D,SAT}$ that will eat into the headroom. A low Gm/Id of 10 is chosen, as it is desirable for a current source for better noise efficiency. From Fig. 34 and Fig. 35 for a Gm/Id of 10, a length of 0.5µm for the NMOST and 0.6µm for the PMOST is chosen to have sufficient output impedance with a $V_{D,SAT}$ of less than 150mV. The width of the NMOS tail current source is set to the minimum and the width of the PMOST is 5 times bigger to match the current set by the NMOS transistor.

Table 8: OTA Transistor Sizes

Transistor	Gm/Id	Size (µm)
Mn1 (Current Source)	9.2	0.24 / 0.5
Mp1(Current Source)	9.2	1.15 / 0.6
Mn2 (Signal Transistor)	20	32.5 / 3.0
Mp2 (Signal Transistor)	23.2	160 / 3.0

The feedback resistor R_{fb} is used to set the bias of the input pair. A resistor of 1GΩ is used so that the output impedance of the amplifier is not affected and does not kill the gain. For moderate accuracy the feedback capacitor (C_{FB}) is chosen to be 200fF. The input capacitors (C_{IN}) are implemented by a capacitor bank, so that a variable gain can be used for the varying input voltage that ranges from 1-10mV per channel. The input capacitors are 600fF, 1.2pF, 2.4pF and 4.8pF for gains of 3,6,12 and 24 respectively.

Smaller capacitors can be used, but they were significantly affected by the loading of the amplifier itself. For an accurate gain ratio, increasing the capacitor value helps as the loading of the amplifier itself becomes less dominant. However, making them excessively big in the order of a few picofarads has two effects. Firstly, the area increases significantly. Secondly, a bigger input capacitor reduces the input impedance which is an undesired effect.

Although the capacitor values chosen experience much less loading in comparison to using minimum size capacitors, there is still some effect which leads to a gain error. However, as the input signal itself is highly varying, the variable gain array allows for larger amplifications to make use of the full scale range of the ADC. In the next sub-section, the simulation results of the amplifier are given to verify the sizing procedure followed above.

6.2.3 Verification and Waveforms

Fig. 38 and Fig. 39 show the gain and phase margin of the amplifier. As expected from the above sizing a DC gain of 52dB was achieved with a sufficient phase margin of 83.4°.

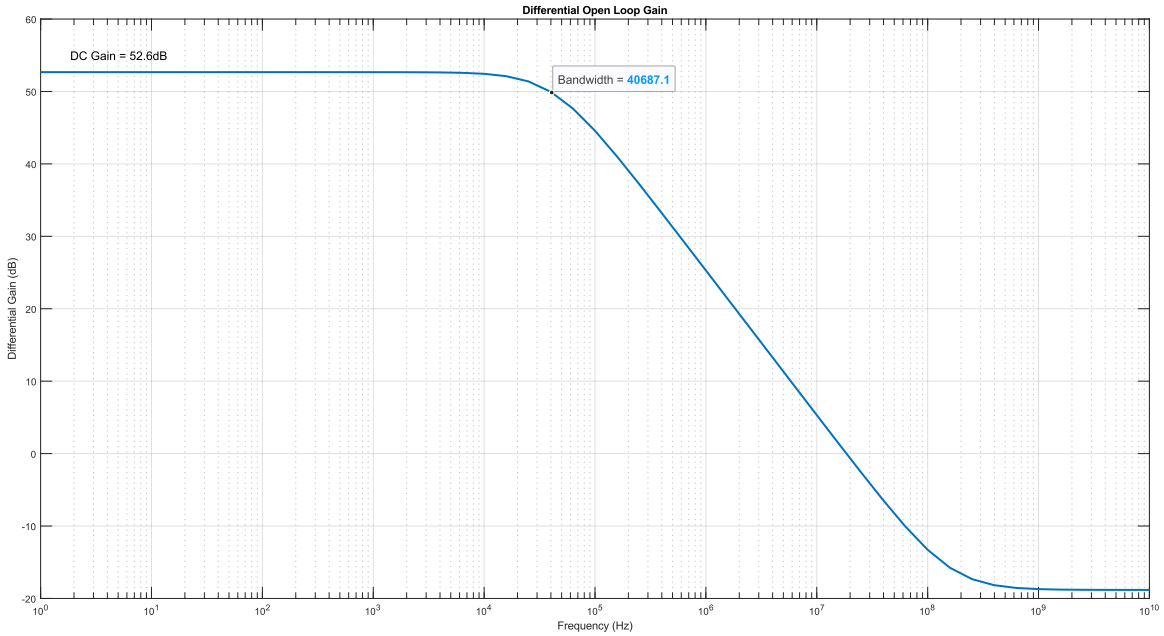


Fig. 38: Differential Open Loop Gain

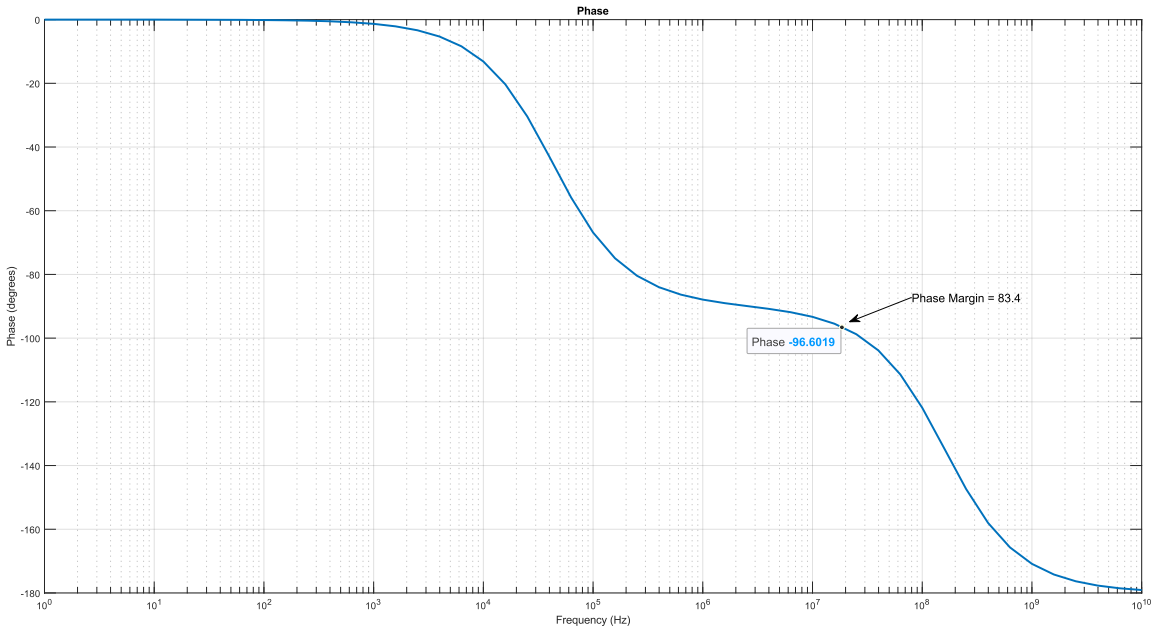


Fig. 39: Phase Margin

The common mode rejection ratio (CMRR) of the amplifier is a critical part and the limiting factor of a multi-channel system that uses a shared electrode. As explained in Section 6.2.2, the amplifier designed used longer length transistors for the current sources for better output impedance to achieve a good CMRR. Fig. 40 and Fig. 41 plot the common mode gain and CMRR respectively. It can be seen that a sufficiently high CMRR of 77dB is achieved [19].

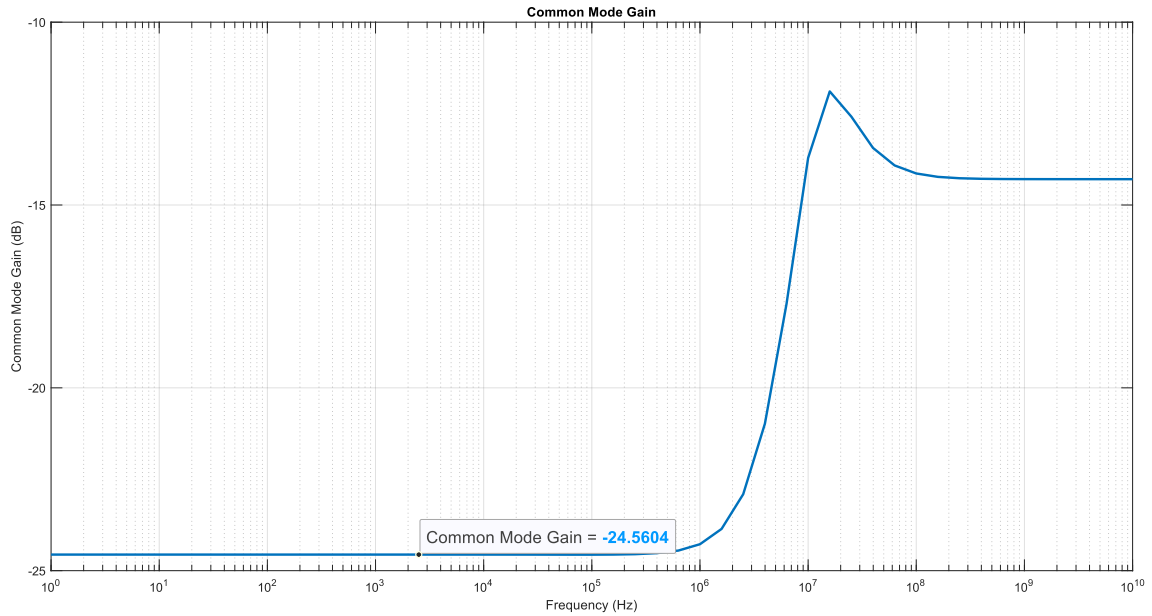


Fig. 40: Common Mode Gain

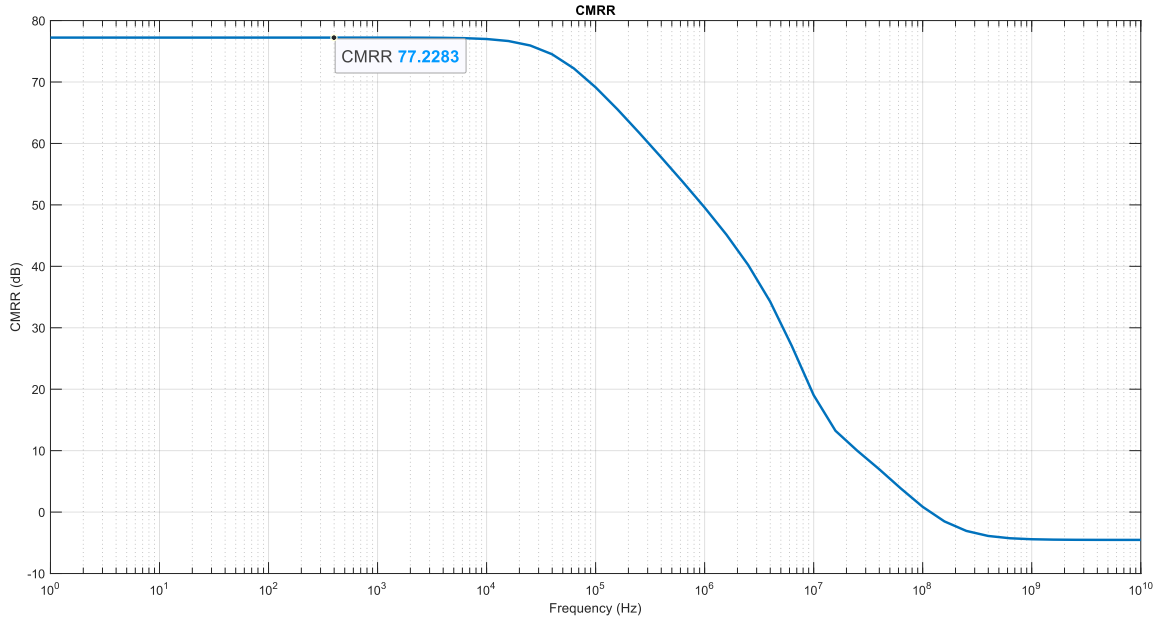


Fig. 41: Common Mode Rejection Ratio

6.3 Analog to Digital Converter

6.3.1 Description and Specification

The analog to digital converter implemented is a ramp or counter type converter. This type of ADC is chosen since the application runs at very low frequencies (10s of kHz) and requires a low resolution of 6 bits.

The input signal is compared to a ramp signal that increments by one LSB for each comparison. As long as the input voltage is greater than the reference ramp voltage the output of the comparator is '1' and the counter keeps running. When the ramp voltage crosses the input voltage the output of the comparator outputs a logic '0' and the counter stops running. The counters value at this point is the digital output and it is read out using a register. Fig. 42 is a top level schematic of the ramp ADC followed by the specifications in Table 9.

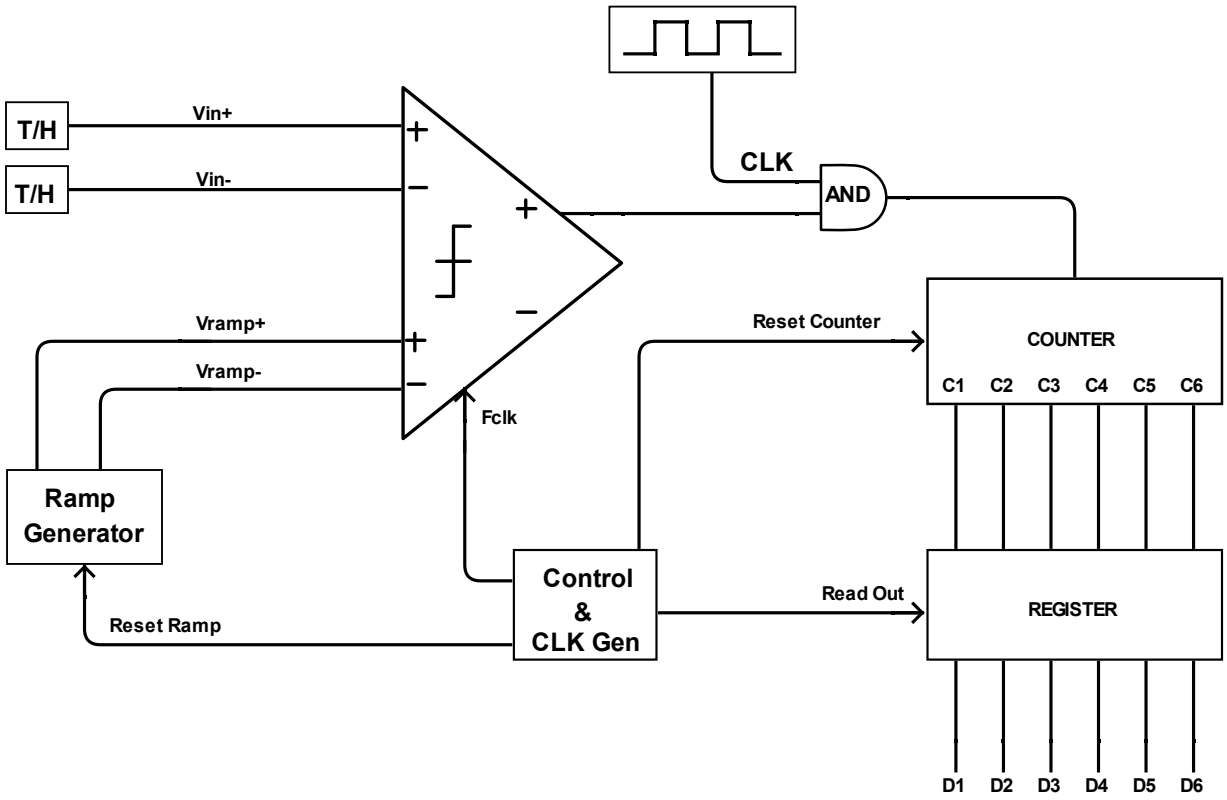


Fig. 42: Top Level Schematic of Ramp/Counter ADC

The sampling clock and modulation clock signals are shown in Fig. 43. The sampling frequency is 4 times the bandwidth of the signal and the sampling edges do not overlap with the transitions of the modulating clock. The comparator is clocked at 2.24MHz where 64 clock cycles are for comparison, 4 clock cycles for resetting the counter and 2 for reading out the digital output at the end of the conversion.

Table 9: ADC Specifications

Specification	Value
Signal Bandwidth (f_B)	4kHz
Sampling Frequency (f_S)	16kHz
Comparator Clock (fclk)	2.24MHz
Number of Bits (N)	6 bits
Full Scale Range (V_{REF})	150mV_{pk-pk}
LSB	2.34mV

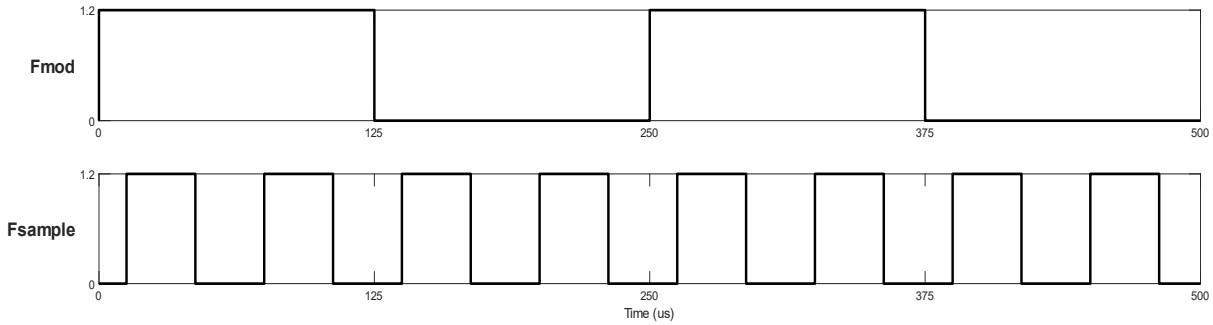


Fig. 43: Modulation and Sampling Clock Signals

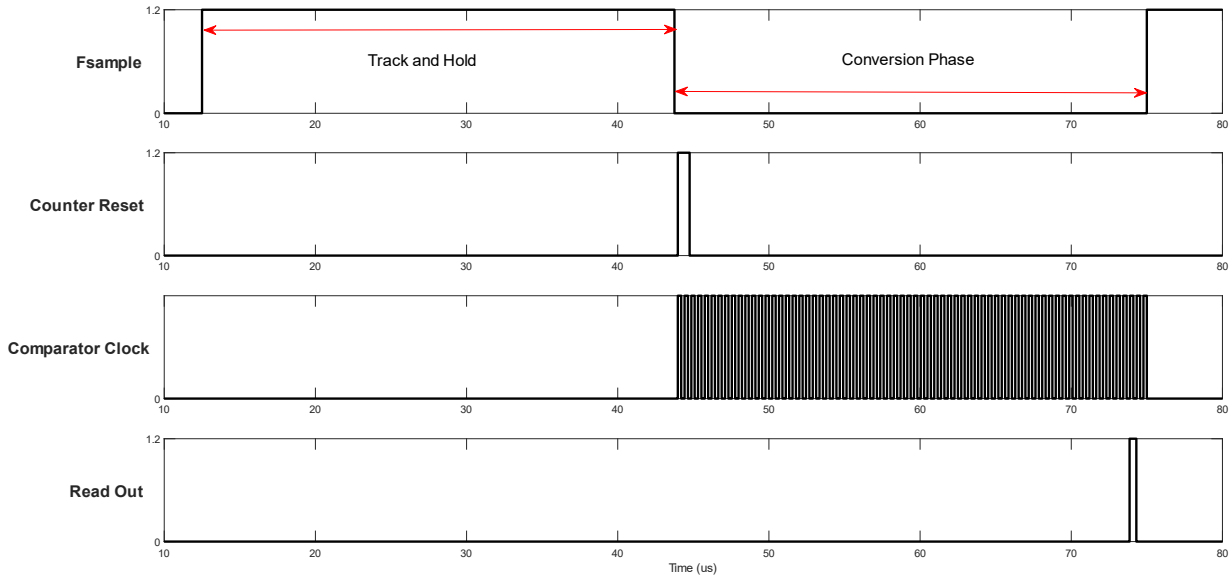


Fig. 44: ADC Control Signals for one cycle

6.3.2 Comparator

A dynamic StrongArm Latch comparator is used due to its low power consumption as a current is drawn only during the regeneration phase. The comparator is sized to minimize the input referred offset to less than half an LSB value.

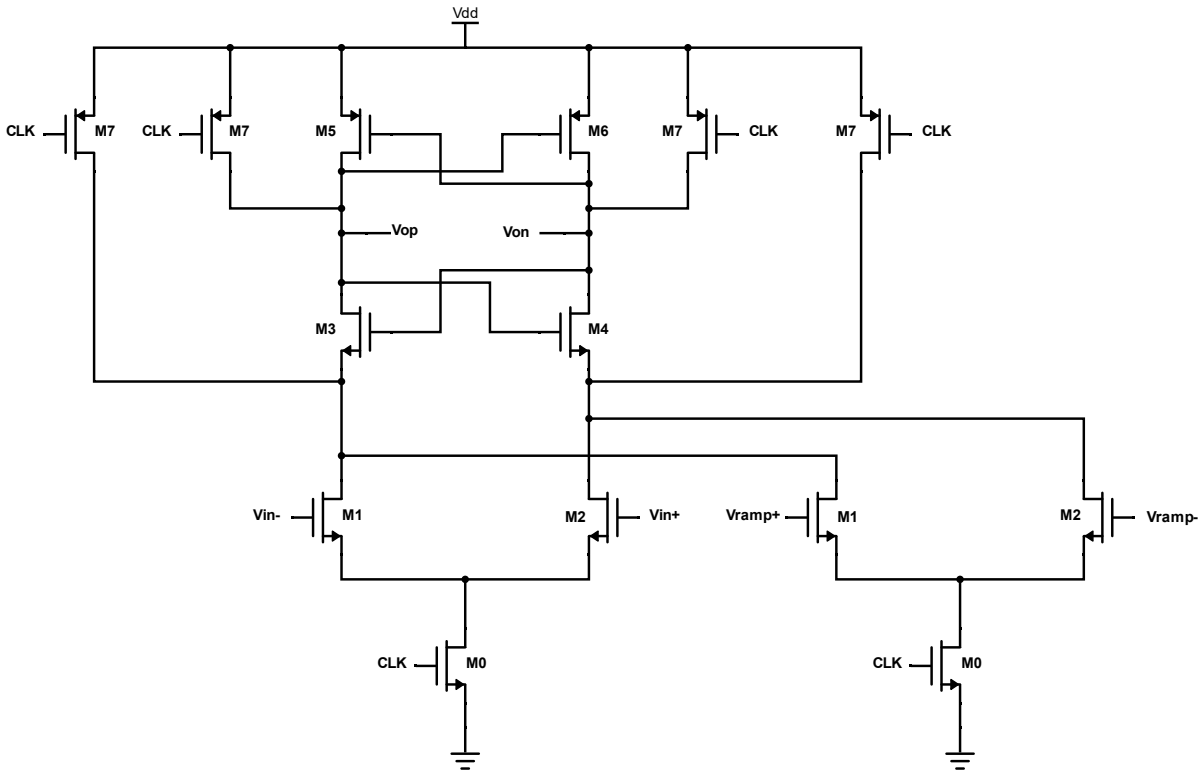


Fig. 45: Strong Arm Latch Comparator

The four reset switches M_7 need to be able to pull up either of the nodes from ground to VDD within half a clock cycle before the next comparison is to be done. As the comparator is clocked at a relatively low speed, setting them to the minimum size achieves the target. Similarly, the PMOS cross-coupled pair is set to minimum size as the offset contribution to the input is minimal as it is attenuated by the gain of $M_{3,4}$ and $M_{1,2}$.

The input pair $M_{1,2}$ and the NMOS cross-coupled pair $M_{3,4}$ are sized to minimise offset. For the input pair an initial length of 2µm and a width of 4µm is chosen. The length of $M_{3,4}$ is set to minimum size and its width is scaled in each iteration in proportion to the input pair. After running a few Monte Carlo iterations, the input pair width is determined to be 8µm and the width of $M_{3,4}$ is 2.5µm.

To determine the input referred offset, first the comparator is given a 0mV input difference and a Monte Carlo is run to check if the decisions are equally balanced. The comparator's first decision is sampled at the same instant for every run to ensure that the effect of hysteresis does not affect the calculation. Fig. 46 plots the distribution and it can be seen that there are an equal number of 1s and 0s which indicates the mean is centered around 0.

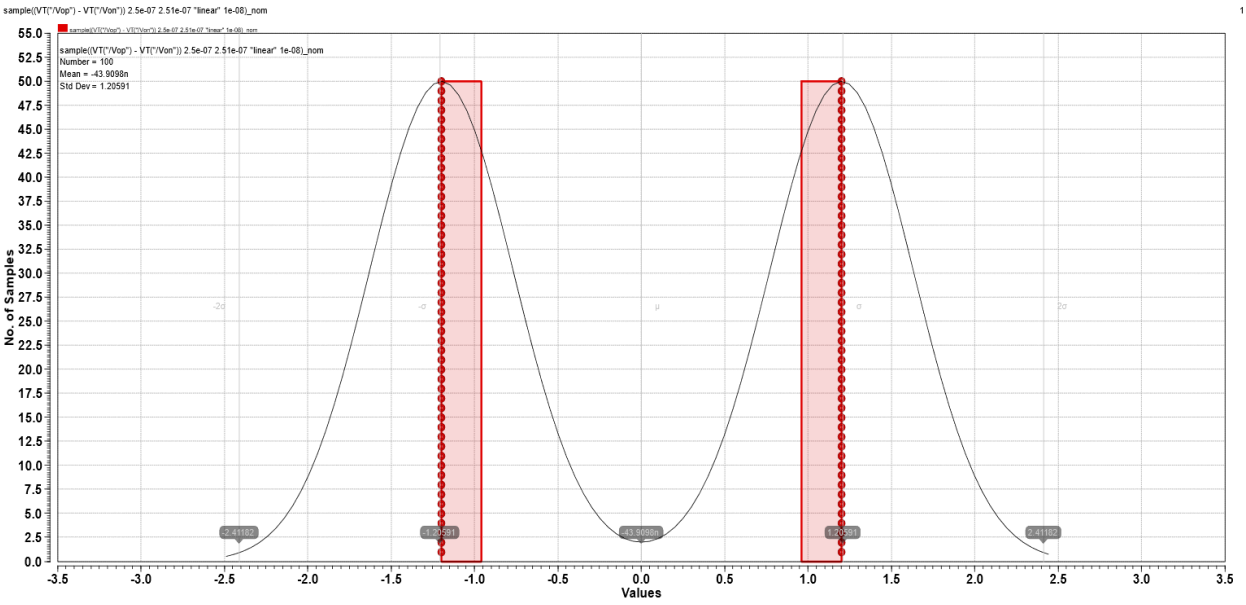


Fig. 46: Number of 1s and 0s for a balanced input (0mV input difference)

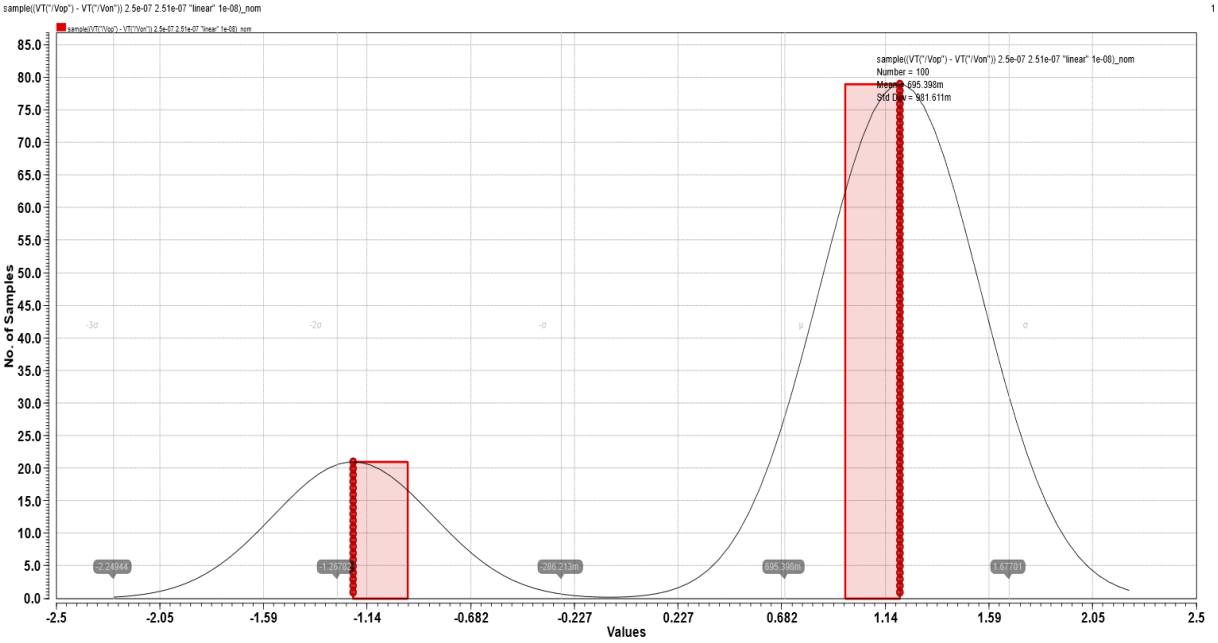


Fig. 47: Distribution of 1s and 0s with a +1mV skewed input

Consequently, the comparator is skewed by feeding in a 1mV difference. From the distribution in Fig. 47, it can be seen that 78% of the decisions are a '1' and 22% of the decisions are a '0'. From this data, the standard deviation can be calculated which gives an indication of the input referred offset. The assumption made here is that the offset is a gaussian distribution. This is validated by skewing the comparator with 1mV and 2mV differences in both directions. From all the distributions, the standard deviation was computed and it turned out to be around the same value. The distributions and calculations for the other inputs are given in Appendix B.2.

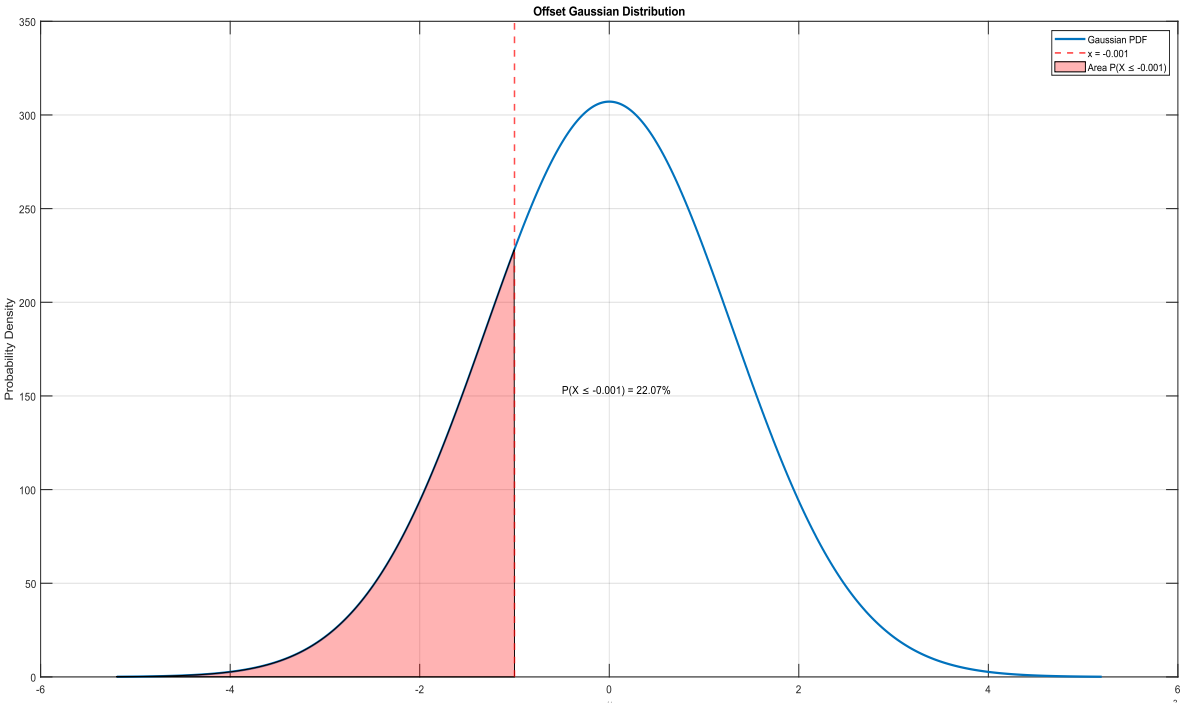


Fig. 48: Calculation of Offset from the probability distribution at a 1mV difference.

By plotting the distribution as shown in Fig. 48, the standard deviation is computed by using the Z table available in Appendix B. For a probability distribution of 22% up to -0.001 (1mV), the z score is -0.77. The z score is given by:

$$z = \frac{(x - \mu)}{\sigma} \tag{6.3.1}$$

Using equation 6.3.1, σ can be calculated

$$\sigma = \frac{(-.001 - 0)}{-0.77}$$

$$\sigma = 0.00129 = 1.29mV$$

Thus, the input offset of the comparator meets the requirement as it is around half an LSB.

The sizes of all the transistors of the comparator based on the design strategy above are summarized in Table 10.

Table 10: Comparator Transistor Sizes

Transistor	Size (µm)
$M_{1,2}$ (Input Pair)	8.00 / 2.00
$M_{3,4}$ (NMOS XCP)	2.50 / 0.18
$M_{5,6}$ (PMOS XCP)	0.25 / 0.18
M_7 (RST Switches)	0.25 / 0.18
M_0 (Current Source)	0.25 / 2.00

6.3.3 Track and Hold

The track and hold circuit used is a simple NMOS switch with a sampling capacitor as shown in Fig. 49. The input signal is tracked for half the clock period when the clock signal is high, and a sample is taken on the falling edge of the clock. A 0.5pF sampling capacitor is used with a switch of width 1µm and length 0.18µm.

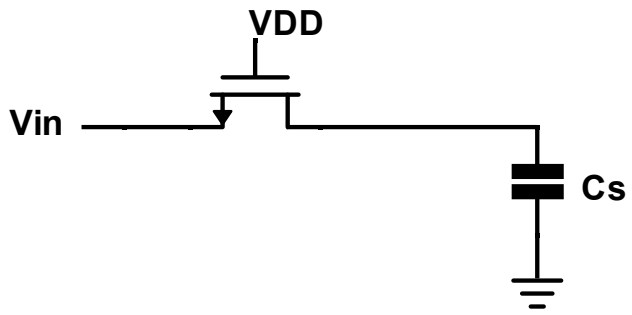


Fig. 49: Track and Hold Circuit

6.3.4 6-bit Register

At the end of the counting during each conversion phase, the value stored in the counter is readout using a shift register as shown in Fig. 50.

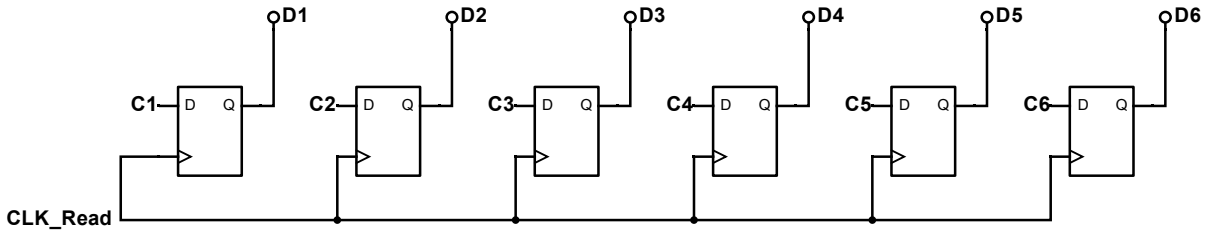


Fig. 50: 6-bit Register to save the counters output value

6.3.5 6-bit Counter

During the conversion phase, the counter runs till the reference voltage crosses the input voltage. The counter implemented is an asynchronous counter as shown in Fig. 51, that is built using the standard digital logic cells from the TSMC library.

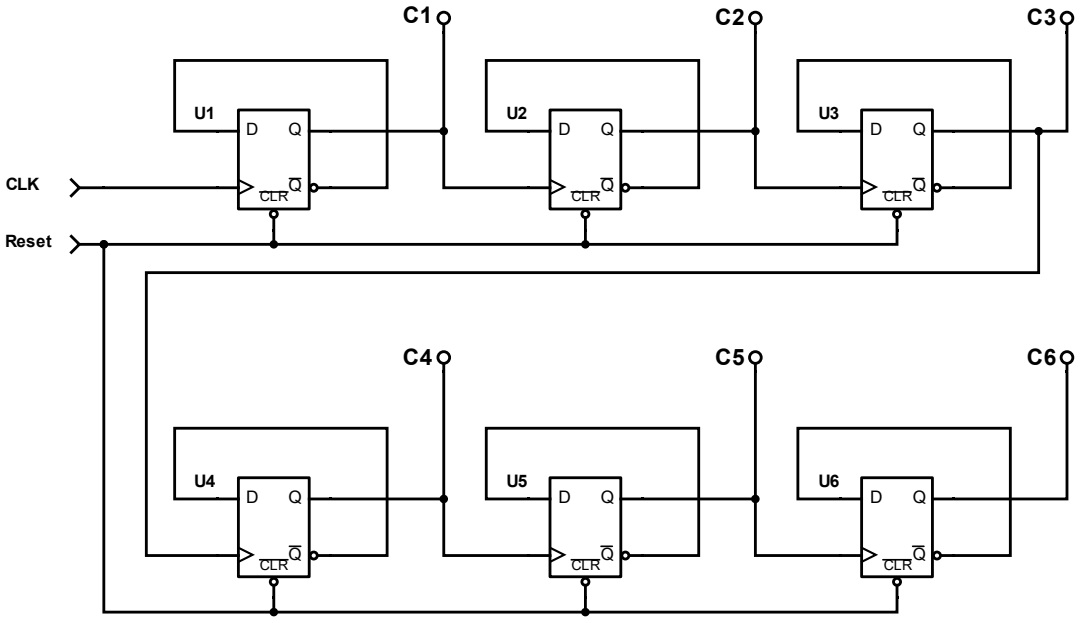


Fig. 51: 6 bit Counter

6.3.6 Verification and Waveforms

The performance of the ADC was characterized by simulating it with an input sine wave of 150mV$_{pk-pk}$ at 4kHz, as this is the optimum bandwidth for a 7 channel system as derived in Section 4.4.

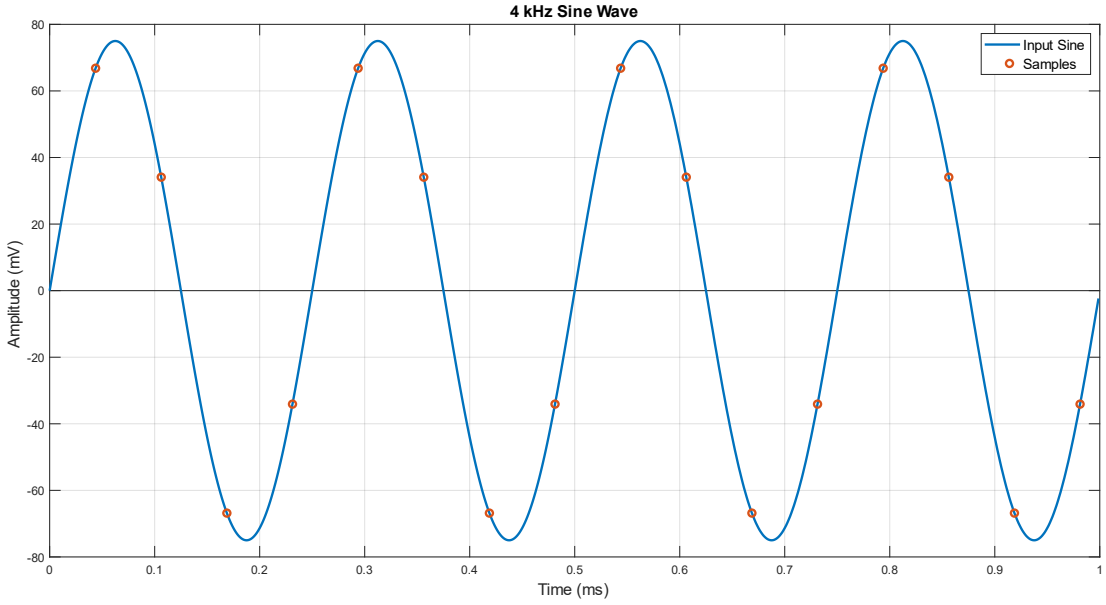


Fig. 52: 150mV$_{pk-pk}$ 4kHz input signal sampled at 16kHz. The dots represent the samples taken.

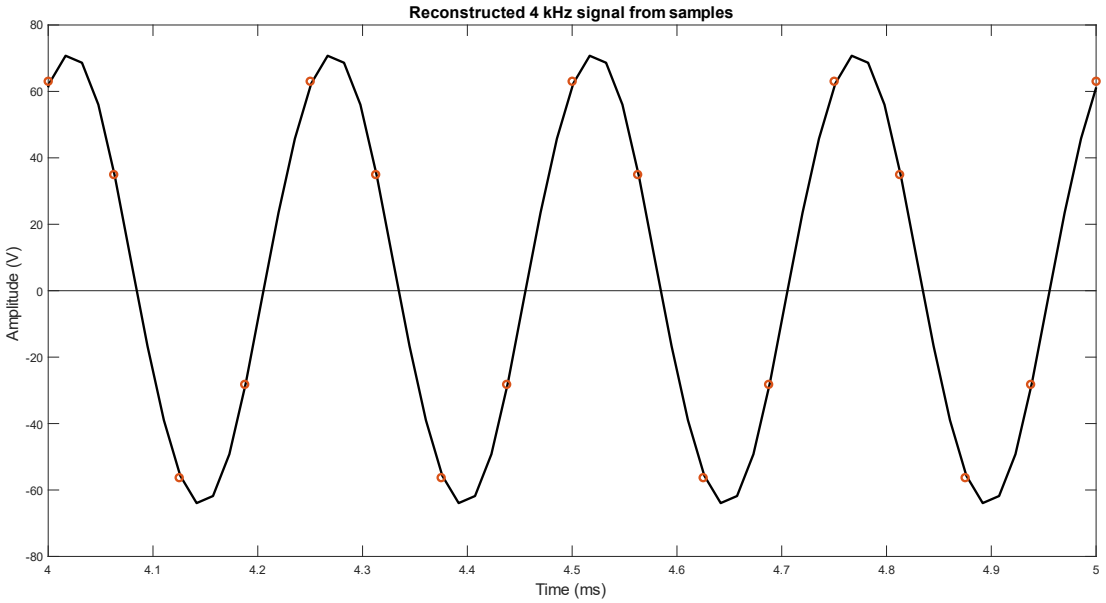


Fig. 53: Reconstructed analog signal from the digital output.

The input signal with the samples taken at the sampling instants are shown in Fig. 52. The quantized samples are digitized for 256 cycles and the digital data is exported to MATLAB for

reconstruction. Fig. 53 is the reconstructed signal from the digitized points. From these data points a smooth analog signal is reconstructed by using the resampling function in MATLAB with an up sampling factor of 4.

An FFT is performed on the reconstructed analog signal with 4096 points which is plotted in Fig. 54. The main signal tone is present at 4kHz with distortion component at 12kHz which is an odd multiple of the input signal. The even order harmonics are supressed. From the FFT data, the SNR, SNDR and SFDR are extracted. The ADC performance parameters are summarised in Table 11.

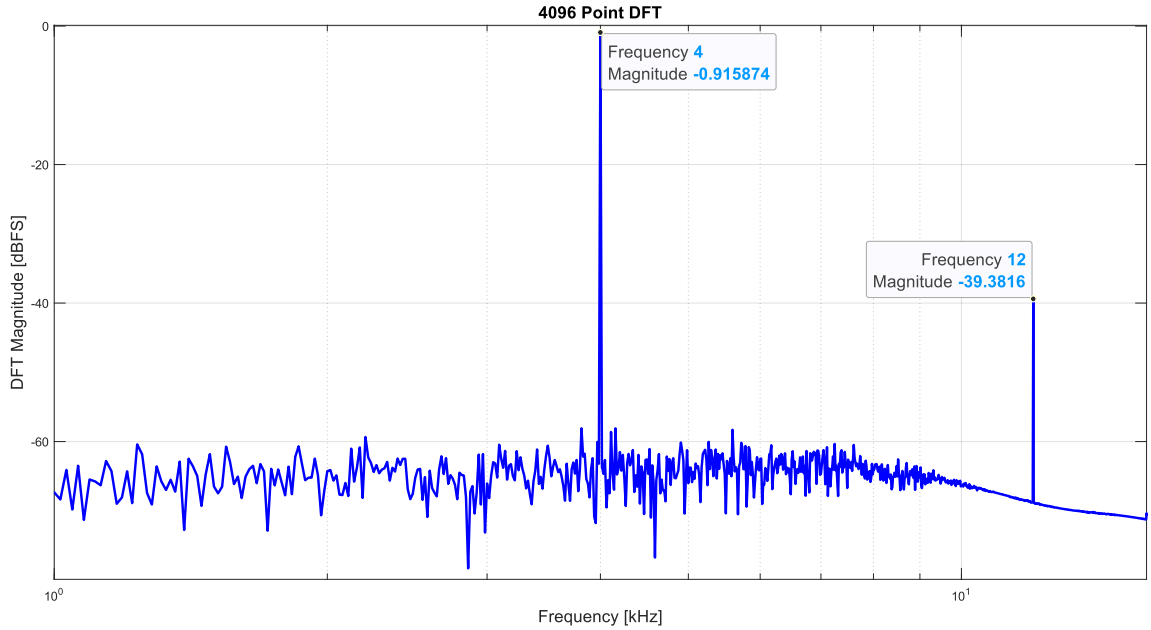


Fig. 54: FFT of Reconstructed Signal

Table 11: ADC Performance Metrics

Parameter	Value
Sampling Rate	16 kS/s
SNR	34.17 dB
SNDR	32.77 dB
SFDR	38.46 dB
THD	-19.18 dB
ENOB	5.15 bits

For a 6 bit ADC oversampled at twice the Nyquist rate, the Ramp ADC that was realised achieves an SNDR of 32.77dB which corresponds to an ENOB of 5.15 bits. In Section 4.2, the required resolution was determined to be 5 bits for the atrial electrogram signals to be reconstructed with a PRD of less than 3%. Operating at an optimal bandwidth of 4kHz for a 7 channel system, the ADC provides a sufficient resolution to reconstruct the signal with sufficient accuracy.

Chapter 7

Conclusion and Discussion

7.1 Conclusions

With the ever increasing number of electrodes for recording signals invasively, multiplexing of channels to share the front-end recording circuits becomes crucial to realize a power and area efficient chip. Time, frequency, and code multiplexing are the available techniques available to do so. Among the three techniques, code multiplexing makes efficient use of the total bandwidth of the front-end, provided that the coding scheme used has the capability to minimize cross-talk to a great degree. The aim of this thesis was to investigate the limitations in a code-multiplexed system and design a power and area efficient system by reducing the complexity and addressing bottlenecks at the system level. In this regard, the following questions as mentioned in the problem statement are answered:

1) How do various multiplexing coding schemes contrast in their capability to minimize cross-talk between channels while utilizing an optimal bandwidth?

Pseudo-random and orthogonal Walsh-Hadamard sequences were contrasted by considering the properties of the codes themselves such as cross-correlation and their ease of generation on chip. WH codes have better cross-correlation properties and require lower modulation rates, thus reducing the overall bandwidth requirements of the analog front end.

2) How can generation of these multiplexing codes be achieved on-chip with minimal form factors that show a significant improvement from existing techniques?

While pseudo-random codes have poor cross-correlation properties, they can be realized relatively easily using LFSRs on chip. This makes them useful when area is a concern. Walsh-Hadamard codes on the other hand have good cross-correlation properties but have the limitation of storage or computation on chip. However, in this thesis, a novel method of deriving WH sequences from a clock signal was proposed and implemented using elementary digital logic. The algorithm proposed has considerable area savings in comparison to an LUT especially for codes that have a length of 8 or greater.

3) To what degree can digital signal processing (DSP) off-chip help alleviate the recording constraints so that an area and power efficient chip can be realised?

As the entire demodulation happens in the digital domain, standard signal processing techniques can be deployed to recover the signal. In this thesis, an ideal digital to analog conversion was performed to recover the signal followed by the de-spreading of the individual channels with the code sequences. Once the different channels are separated, applying a moving average and low pass filter help remove any higher order frequencies present in the signal. By deriving the optimum parameters of each step in the signal chain, as explained in Chapter 4, the on-chip recording can use the minimal required bandwidth. A smaller signal bandwidth helps save power and area as the front-end amplifier and ADC can operate at a low bandwidth and conversion rate respectively.

4) What circuit architectures can be deployed for higher efficiencies to reduce power and area that are also suitable for advanced technology nodes with shrinking voltages?

One of the focuses of this thesis is to design circuits that can easily be adapted to newer technology nodes with shrinking supply voltages. In that aspect, digitally inspired analog blocks were investigated, as they are more power efficient and compatible with smaller nodes with smaller supply voltages. For front-end amplifiers inverter based architectures can provide higher noise and power efficiencies due to their current reuse. The challenge with smaller nodes and lower voltages is that achieving high gains is difficult due to the lower intrinsic gains and the use of cascodes being difficult or impossible. Hence, for higher gain requirements, multiple stage amplifiers will be required. Additionally, depending on the supply and threshold voltages of the transistors, self-biasing maybe a viable option to reduce power consumption and area as biasing circuits can be eliminated.

7.2 Performance Comparison

The Walsh-Hadamard code generator, low-noise amplifier and ramp ADC designed consume a total power of 78.4µW for 7 channels. This corresponds to 11.2 µW per channel. Table 12 summarizes the current and power consumption of each of the circuit blocks. The supply voltage used for the analog as well as digital blocks is 1.2V.

Table 12: Power Consumption Summary

Circuit Block	Current	Power
WH Code Generator	3.49 nA	4.20 nW
Amplifier	3.94 µA	4.73 µW
ADC	61.39 µA	73.67 µW
Total	65.33 µA	78.40 µW

Table 13 compares the performance of this work with state of the art designs that use code-division multiplexing to share a number of channels. In comparison to the work down in [5], the power consumption has significantly improved by a factor of 3.3. However, the power consumption per channel is considerably higher than that reported in [6]. The limiting factor in terms of power consumption is the ADC. This is due to the large number of comparisons made in a counter ADC in comparison to that of a SAR ADC.

Table 13: Performance Comparison

	JSSC'20 [6]	BioCAS'23 [5]	This Work
Modulation/Multiplexing	WH/CDM	PRBS/CDM	WH/CDM
Reference Electrode	Dedicated	Shared	Shared
Power/Channel (µW)	1.97	37.26	11.2
Shared Blocks	LNA, ADC	LNA, ADC	LNA, ADC
ADC Architecture	Async. SAR	$\sum \Delta$	Ramp
Look-Up Table	Yes	No	No
On-Chip Code Generator	No	Yes	Yes
Number of Channels (N)	15	4	7
Supply (V)	1.2 Analog/1.8 Digital	1.8	1.2
Channel Gain(dB)	40 - 56	12 - 30.1	9.5 - 27.6
Process(µm)	0.18	0.18	0.18 BCD

7.3 Future Work and Recommendations

1) In this thesis, for the code-division multiplexing only Pseudo-Random and Walsh-Hadamard codes were explored. The use of Walsh-Hadamard codes proved to be better due to the orthogonality of each code in the set. However, there are many other sets of codes that exist that could possibly have better properties and allow for lower modulation rates and lower bandwidths or even allow for more channels. Exploring other types of codes might lead to a more optimized efficient design.
2) Some of the system level parameters in this design were derived from the MIT-BIH database of atrial electrograms. These are ambulatory recordings from only two channels. By testing the system on actual recorded data, the system level parameters may vary to a certain degree and the optimum values could be different. Hence, these design parameters can be revised depending on the type of signal recorded and its characteristics.
3) The front-end amplifier designed consisted of a single stage with a relatively low open loop gain of about 54dB. Due to the low open-loop gain the gain error was higher and larger capacitor ratios (closed-loop gain) had to be used to make use of the full scale range of the ADC. A straightforward method of increasing the open-loop gain of the amplifier can make the ratios more accurate and reduce the capacitor values thereby reducing the area.
4) To reduce the total area due to the input capacitor bank for each channel, a better solution is to split the amplifier into 2 closed-loop stages with a variable capacitor bank after the first stage. Doing so will reduce the input capacitance which will increase the input impedance that the channels see. Additionally, since the channels are already summed in the first amplifier, the capacitor bank of the second amplifiers input can only be a single bank of capacitors as opposed to the current design wherein each channel consists of a separate capacitor bank. This leads to massive area savings but will consume more power as two amplifier stages are used. For larger number of channels, this solution will be more viable as opposed to using a capacitor bank for each channel.
5) The limiting factor in terms of power consumption is the ADC. By deploying a SAR ADC in contrast to a Ramp ADC the power consumption can be reduced. This is because the number of conversion steps performed in a SAR is much less than that of a ramp.

Appendices

A.1 MATLAB Codes for Generation of Polynomials

A.1.1 Generation of Primitive Polynomials

```
%% Generation of primitive polynomials
% This program finds the primitive polynomials for  a given order.
% The primitive polynomials are a maximum length sequence PRBS generators.
% The terms present are the taps of the LFSR.

clear;
clc;
close all;

%% Input from user:
deg = input('Enter the degree of the polynomial '); % The degree of the polynomial.
N = (2^deg)-1; % The maximum length of the sequence that will be generated.
Possiblities = (2^(deg-1)); % Number of polynomials to test.


%% Computation of polynomials
starting_point = (2^deg)+1; % The decimal starting point for the polynomial. Check
the odd ones only. Skip every 2.
end_point = (2^(deg+1))-1; % The decimal ending point for the polynomial. Last
polynomial to be checked.

syms x;
poly_max = x^N + 1; % The maximum degree polynomial. Find the prime factors of this
to get ML sequences;

polys_to_check = [starting_point:2:end_point]'; % The polynomials to be checked.
primitive_polys = isprimitive(polys_to_check); % The primitive polynomials ( need to
convert this to a symbolic polynomial).
num_of_polys = sum(primitive_polys == 1); % The total number of primitive
polynomials in the set.

primitive_polys_bin = decimalToBinaryVector(polys_to_check(primitive_polys==1));
% Polynomial converted to tap positions


%% Conversion to symbolic representation (Just for visual appeal and ease of use)
poly_symbolic = cell(size(primitive_polys_bin, 1), 1); % Symbolic Polynomials array
initialised.

for i = 1:num_of_polys
    poly_symbolic{i} = poly2sym(primitive_polys_bin(i,:), x); % Convert each co-
efficient polynomial to symbolic expression. This is a cell array.
end
poly_symbolic = cell2sym(poly_symbolic); % Symbolic expression in normal array.
Converted from cell to normal array.
```

A.1.2 Finding the Preferred Pairs of Polynomial

```
%% Finding Preferred Pairs of polynomials
% This code finds the preferred polynomials for a given degree by calculating the
correlation between them.
% Preferred polynomials are those which have certain values of correlation.
% Correlation values: t(n) = 1 + 2^[(n+2)/2]. Values taken are -1, -t(n) and t(n)-2.

%% Call "Generation of Primitive Polynomials" Script and calculate 3 valued
spectrum.
Generation_of_Primitve_Polynomials; % Call "Generation of Primitive Polynomials.m"
from A.1.1 to generate all the polynomials for a given order.

t_n_odd = 1 + 2^(floor((deg+2)/2)); % Odd order correlation value
ideal_3_values = [-t_n_odd, -1, t_n_odd-2]; % The three correlation values that make
up a preferred pair.

ideal_4_values = [-(1+2^((deg+2)/2)), -(1+2^((deg)/2)), -1, -(1-2^((deg)/2))]; % The
four correlation values that make up a preferred pair for polynomials mod(n,4)==0.


%% Use maximum length sequence in built function to generate a ML sequence.
initial_condition = [zeros(1,deg-1), 1]; % Defining an initial condition for LFSR.

counter=1;
for i=1:num_of_polys-1
    for j =i+1:num_of_polys
        pnSequence1 =
comm.PNSequence(Polynomial=primitive_polys_bin(i,:),InitialConditions=initial_condit
ion, SamplesPerFrame=N);
        pnSequence2 =
comm.PNSequence(Polynomial=primitive_polys_bin(j,:),InitialConditions=initial_condit
ion, SamplesPerFrame=N);
        data1 = pnSequence1()';
        data2 = pnSequence2()';
        [correlation, lags] = gf_corr_improved(data1, data2,deg);
        %[correlation, lags] = gf_corr(data1, data2);
        if mod(deg,4)==0
            spectrum_4_values = unique(correlation);
                if isequal(ideal_4_values,spectrum_4_values)
                preferred_pairs(counter,:) =
[poly_symbolic(i,:),poly_symbolic(j,:)];
                counter=counter+1;
                end
        else
            spectrum_3_values = unique(correlation);
                    if isequal(ideal_3_values,spectrum_3_values)
                    preferred_pairs(counter,:) =
[poly_symbolic(i,:),poly_symbolic(j,:)];
                    counter=counter+1;
                    end
        end

    end
end

%% Export Data to Excel File.
pairs_export = arrayfun(@char,preferred_pairs, 'uniform', 0);
xlswrite('Degree 13 Polynomials.xlsx', pairs_export);
```

A.1.3 Cross Correlation Function Calculator.

This function is created to compute the correlation function between two sequences. The way a correlation is calculated between two sequences is by taking the dot product of one sequence with a shifted version of the other. The dot product of two binary sequences is defined as: Number of positions where bits match – Number of positions where bits mismatch.

```
%% GF2 Correlation Function
% It takes 2 inputs as vectors and computes the correlation between them.
% If you want autocorrelation, put both inputs as same vector.

%% Function definition and statements
function [corr_spectra, lags] = gf_corr(input1,input2)
input1_len = length(input1);
input2_len = length(input2);

if input1_len ~= input2_len
    error('Lengths of both vectors must be equal')
else
    lags = 0:input1_len-1;
    matching = zeros(1, input2_len);
    non_matching = zeros(1, input2_len);
    corr_spectra = zeros(1, input2_len);

    for i = 1:input2_len
        data_shifted_i = circshift(input2, lags(i));
        matching(i) = sum(input1 == data_shifted_i);
        non_matching(i) = sum(input1~= data_shifted_i);
        corr_spectra(i) = matching(i) - non_matching(i);
    end
end
```

A.1.4 Cross Correlation Function Calculator for faster computation.

This function is an optimised calculator for a preferred pair of polynomials. It skips the computations once a value is not the preferred values. This function can be strictly used for computing the preferred pairs only and not to find a correlation in general.

```
%% GF2 Correlation Function Improved Version
% It takes 2 inputs as vectors and computes the correlation between them.
% If at any instant the correlation exceeds preferred values it terminates. This
is to speed up computation time.
% If you want autocorrelation, put both inputs as same vector.
% Output is [corr_spectra, lags]. The first argument is the correlation spectrum.
Second output is the shifts.

%% Function definition and statements
function [corr_spectra, lags] = gf_corr_improved(input1,input2,deg)
input2_len = length(input2);

t_n_odd = 1 + 2^(floor((deg+2)/2));
```

```
14
15  lags = 0:input2_len-1;
16  matching = zeros(1, input2_len);
17  corr_spectra = zeros(1, input2_len);
18
19  for i = 1:input2_len
20      data_shifted_i = circshift(input2, lags(i));
21      matching(i) = sum(input1 == data_shifted_i);
22
23      corr_spectra(i) = -input2_len + 2*matching(i);
24      if ~((corr_spectra(i) == (-t_n_odd)) || (corr_spectra(i) == (t_n_odd-2)) ||
25  (corr_spectra(i) == -1))
26          break;
27      else
28          continue;
29      end
30  end
```

A.2 MATLAB-LTSpice Automation for WH Code Generation

This script interfaces MATLAB and LTSpice to build the Walsh Hadamard Code generator for a given degree. It automates the writing of the netlist and then runs a transient simulation. The transient simulation results for each of the outputs are then imported into MATLAB and compared with the original WH matrix to see if the correct sequence has been generated.

For the script below to run, the LTSpice2Matlab toolbox that is available in the link: https://github.com/PeterFeicht/ltspice2matlab

The instructions to setup the link between MATLAB and LTSpice to write netlists, open and close them are available in the following link: https://medium.com/@amattmiller/running-ltspice-from-matlab-630d551032cc

```
1   %% Generate LTSpice Netlist and Check for any N bit Walsh Hadmard Sequence
2   % Ask the user for size of WH sequence required. Store it in size.
3   % Generate a Hadamard matrix in matlab.
4   % Generate a netlist in LTSpice and do the necessary circuits to generate a WH code.
5   % Sample and import the data to matlab and check if they are equal. If yes, the
6   algorithm devised holds true.
7
8   %% Useful Paths to Have
9   % Path to LTSpiceCall Batch File:  @    W:\Documents_W\Matlab_Spice\LTSpiceCall.m
10  % Path to Netlist File:            @    W:\Documents_W\TUD Masters PG\Thesis\Spice
11  Tries\WH\name.net
12  % x86 LTSpice1                     @    C:\Program Files\LTC\LTspiceXVII\XVIIx86.exe
13  % x64 LTSpice2                     @    C:\Program Files\LTC\LTspiceXVII\XVIIx64.exe
14
15  %% Initial Clearing
16  clear;
17  clc;
18  close all;
19
```

```
%% Accept size & create N bit Walsh Hadmard Sequence in Matlab using 1s(1) & 0s(-1).
size = input('Enter the size of Walsh Hadamard Codes to be generated '); % Size of
WH Sequence required by user

if(mod(log2(size),1)~=0)
    disp("Enter a power of 2");
    return;
else
    H_mat = hadamard(size);   % Walsh Hadmard where each row is a sequence. Matlab
generated array.
    H_mat(H_mat==-1) = 0;     % Convert -1s to 0s for ease of comparison with
generated waveforms

    %% Create LTSpice Netlist
    % Setup
    netlist = sprintf('C:\\Users\\Dion\\Documents\\Docs_Work\\TUD Masters
PG\\Thesis\\Spice Tries\\WH\\WH_automation\\WH_%d.net',size); % Create netlist file

    sim = sprintf('.tran 0 %d 100m 100m\r\n',size);                  % simulation
duration set by size. Default frequency is 1Hz with 50% duty cycle.
    save_wav = sprintf('.wave WH%d.wav 16 1 V(Seq1) V(Seq2) ',size); % Sequence
waveforms to be saved.

    A=strings(1,size);% Netlist code for each code sequence. Stored in array of size
N.
    A{1} = 'A1 Seq1 0 CLK 0 0 0 Seq1 0 DFLOP\r\n';
    A{2} = 'A2 N2 0 CLK 0 0 N2 Seq2 0 DFLOP Vhigh=1.8 Vlow=0\r\n';

    Connections = strings(1,size+log2(size)); % TXT File with connections to
generate WH Code
    Connections{1} = 'Sequence 1 ==> Just DC 1';
    Connections{2} = 'Sequence 2 ==> CLK / 2';
    Connections{3} = '------------------------------';
    starting_point = 2;


    % Generator starts from 2x2 and then builds up 2x at a time. So from 2 to 4 to 8
etc.
    if (size>2)
        for i = 1:log2(size)-1  % Go from 2 to 4 to 8 etc all the way upto N. Sub
operations have to be done log2(size)-1 times

            for j = starting_point+1:2*starting_point   % Start from N+1 upto 2N.
                save_wav=[save_wav,sprintf('V(Seq%d) ',j)];
                if (j==(starting_point+1)) % N+1 term = (N/2)+1 term/2 in frequency

                    A{j} = sprintf('A%d N%d 0 Seq%d 0 0 N%d Seq%d 0 DFLOP Vhigh=1.8
Vlow=0\r\n',j,j,floor((j/2)+1),j,j);
                    Connections{j+i} = sprintf('Sequence %d ==> Half the frequency
of Sequence %d',j,floor((j/2)+1));
                else   % R(N+x) to R(2N) = R(N+1) XNOR R(x). 2<=x<=2N
                    A{j} = sprintf('A%d Seq%d Seq%d 0 0 0 Seq%d 0 0 XOR Vhigh=1.8
Vlow=0\r\n',j,floor(j-starting_point),starting_point+1,j);
                    Connections{j+i} = sprintf('Sequence %d ==> Sequence %d XNOR
Sequence %d',j,floor(j-starting_point),starting_point+1);
                end
                Connections{j+i+1} = '------------------------------';
            end
```

```
            starting_point = 2*starting_point; % Once 2N is created from N, shift N
to 2N to create next iteration of next 2N etc.
        end
    end


    % Netlist Writing
    code = strjoin(['This is WH code \r\n'...
        'VDD Seq1 0 1.8\r\n'...
        'V§CLK CLK 0 PULSE(0 1.8 0 100p 100p 0.5 1)\r\n'...
        A...
        sim...
        save_wav,'\r\n'...
        '.save V(CLK) V(Seq*)\r\n'...
        '.backanno\r\n'...
        '.end\r\n']);

 % Create the new netlist
    fid = fopen(netlist,'w+');
    fprintf(fid,code);
    fid=fclose(fid);

    %% Call and close LTSpice
    fileID = fopen('LTSpice_call.bat', 'w+');
    fprintf(fileID,'%s',sprintf('start "C:\\Program
Files\\LTC\\LTspiceXVII\\LTspice.exe -b" "C:\\Users\\Dion\\Documents\\Docs_Work\\TUD
Masters PG\\Thesis\\Spice Tries\\WH\\WH_automation\\WH_%d.net"',size));
    fclose(fileID);

    dos('LTSpice_call.bat');  % the dos command launches the .bat file
    pause(size);              % Alllows LTSpice to finish simulating
    dos('LTSpice_end.bat');   % Closes LTSpic after the .raw file is created

    %% Reading Data from LTSpice and checking if an actual WH is generated
    [WH_Generated, Fs] = audioread(sprintf('WH%d.wav',size)); % Read the waveform
from LTSpice
    WH_Generated=round(WH_Generated); % Convert the decimal values to whole numbers
for comparison
    truth = isequal(WH_Generated,H_mat); % Check if Circuit Generated Sequence and
Matlab sequence are equal

    raw_data=LTspice2Matlab(sprintf('WH_%d.raw',size));
    %plot(raw_data.time_vect, raw_data.variable_mat(4,:))

    %% Writing Connection Instructions to txt file
    % Specify the file name
    fileName = fopen(sprintf('WH_%d_Connections.txt',size),'w+');

    % Write the string array to a text file
    fprintf(fileName, '%s\n',Connections);
    if truth
        fprintf(fileName,'Succesfully Generated WH Sequence :)');
    else
        fprintf(fileName,'Unsuccesful :(');
    end

    % Close the file
    fclose(fileName);
end
```

B.1 Z-Score Table for Gaussian Distribution Calculations

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
–3.4	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0002
–3.3	.0005	.0005	.0005	.0004	.0004	.0004	.0004	.0004	.0004	.0003
–3.2	.0007	.0007	.0006	.0006	.0006	.0006	.0006	.0005	.0005	.0005
–3.1	.0010	.0009	.0009	.0009	.0008	.0008	.0008	.0008	.0007	.0007
–3.0	.0013	.0013	.0013	.0012	.0012	.0011	.0011	.0011	.0010	.0010
–2.9	.0019	.0018	.0018	.0017	.0016	.0016	.0015	.0015	.0014	.0014
–2.8	.0026	.0025	.0024	.0023	.0023	.0022	.0021	.0021	.0020	.0019
–2.7	.0035	.0034	.0033	.0032	.0031	.0030	.0029	.0028	.0027	.0026
–2.6	.0047	.0045	.0044	.0043	.0041	.0040	.0039	.0038	.0037	.0036
–2.5	.0062	.0060	.0059	.0057	.0055	.0054	.0052	.0051	.0049	.0048
–2.4	.0082	.0080	.0078	.0075	.0073	.0071	.0069	.0068	.0066	.0064
–2.3	.0107	.0104	.0102	.0099	.0096	.0094	.0091	.0089	.0087	.0084
–2.2	.0139	.0136	.0132	.0129	.0125	.0122	.0119	.0116	.0113	.0110
–2.1	.0179	.0174	.0170	.0166	.0162	.0158	.0154	.0150	.0146	.0143
–2.0	.0228	.0222	.0217	.0212	.0207	.0202	.0197	.0192	.0188	.0183
–1.9	.0287	.0281	.0274	.0268	.0262	.0256	.0250	.0244	.0239	.0233
–1.8	.0359	.0351	.0344	.0336	.0329	.0322	.0314	.0307	.0301	.0294
–1.7	.0446	.0436	.0427	.0418	.0409	.0401	.0392	.0384	.0375	.0367
–1.6	.0548	.0537	.0526	.0516	.0505	.0495	.0485	.0475	.0465	.0455
–1.5	.0668	.0655	.0643	.0630	.0618	.0606	.0594	.0582	.0571	.0559
–1.4	.0808	.0793	.0778	.0764	.0749	.0735	.0721	.0708	.0694	.0681
–1.3	.0968	.0951	.0934	.0918	.0901	.0885	.0869	.0853	.0838	.0823
–1.2	.1151	.1131	.1112	.1093	.1075	.1056	.1038	.1020	.1003	.0985
–1.1	.1357	.1335	.1314	.1292	.1271	.1251	.1230	.1210	.1190	.1170
–1.0	.1587	.1562	.1539	.1515	.1492	.1469	.1446	.1423	.1401	.1379
–0.9	.1841	.1814	.1788	.1762	.1736	.1711	.1685	.1660	.1635	.1611
–0.8	.2119	.2090	.2061	.2033	.2005	.1977	.1949	.1922	.1894	.1867
–0.7	.2420	.2389	.2358	.2327	.2296	.2266	.2236	.2206	.2177	.2148
–0.6	.2743	.2709	.2676	.2643	.2611	.2578	.2546	.2514	.2483	.2451
–0.5	.3085	.3050	.3015	.2981	.2946	.2912	.2877	.2843	.2810	.2776
–0.4	.3446	.3409	.3372	.3336	.3300	.3264	.3228	.3192	.3156	.3121
–0.3	.3821	.3783	.3745	.3707	.3669	.3632	.3594	.3557	.3520	.3483
–0.2	.4207	.4168	.4129	.4090	.4052	.4013	.3974	.3936	.3897	.3859
–0.1	.4602	.4562	.4522	.4483	.4443	.4404	.4364	.4325	.4286	.4247
–0.0	.5000	.4960	.4920	.4880	.4840	.4801	.4761	.4721	.4681	.4641

B.2 Distribution of Comparator Output for various inputs

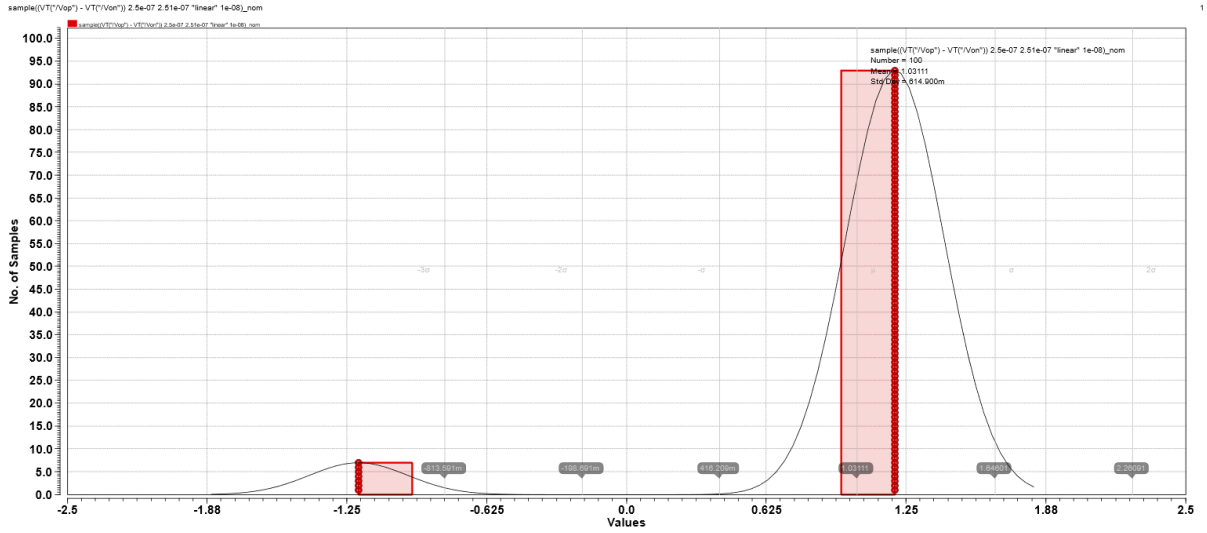


Fig. 55: Distribution of 1s and 0s with a +2mV skewed input

For a +2mV input difference, the distribution is such that 93% of the inputs are a '1' and 7% are a '0'. From this the standard deviation can be computed using 6.3.1:

$$\sigma = \frac{(.002)}{1.48} \quad \rightarrow \quad \sigma = 0.00135 = 1.35mV$$

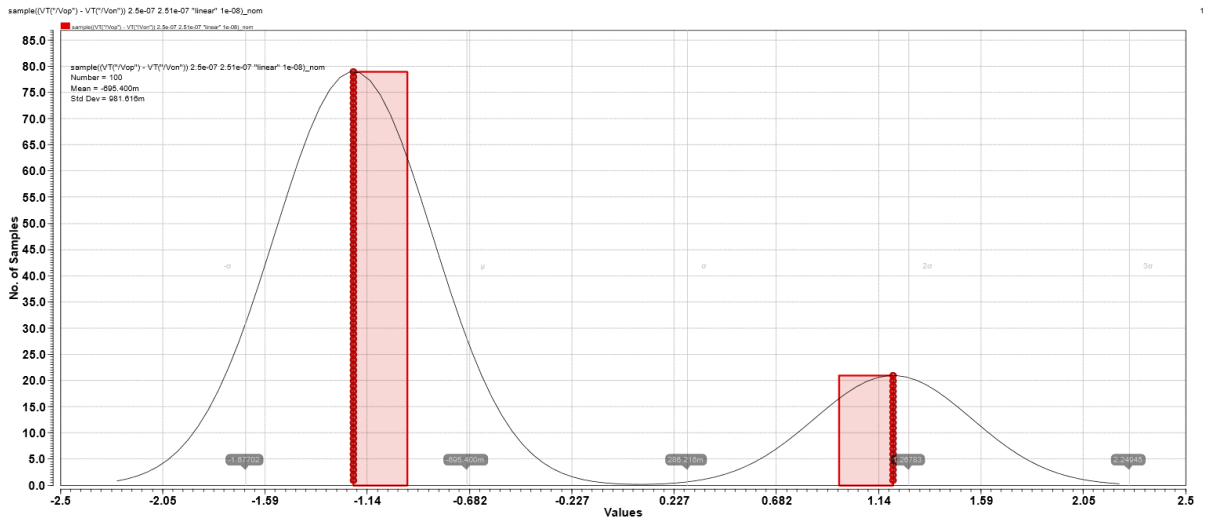


Fig. 56: Distribution of 1s and 0s with a -1mV skewed input

The distribution is similar to that of Fig. 48 where 79% are '0's and 21% are '1's. Using 6.3.1:

$$\sigma = \frac{(.001-0)}{0.81} \quad \rightarrow \quad \sigma = 0.001235 = 1.235mV$$

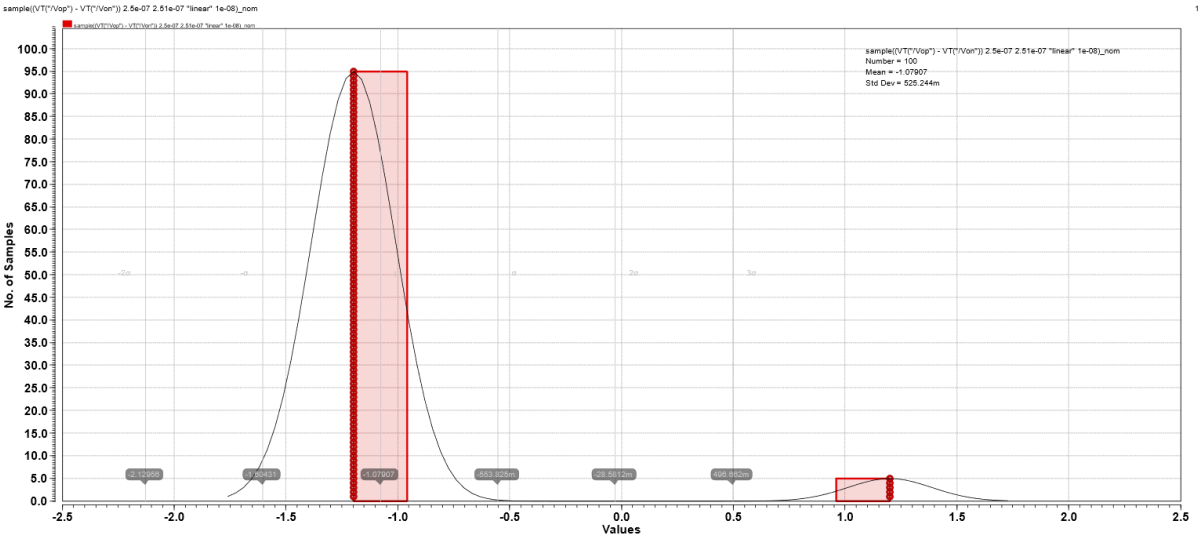


Fig. 57: Distribution of 1s and 0s with a -2mV skewed input

For a -2mV input difference, the distribution is such that 95% of the inputs are a '0' and 5% are a '1'. The standard deviation is:

$$\sigma = \frac{(.002)}{1.645} \rightarrow \sigma = 0.001216 = 1.216mV$$

References

[1] L. Arbeloa-Gómez, J. Álvarez-Vidal, and J. L. Izquierdo-García, “Further Advances in Atrial Fibrillation Research: A Metabolomic Perspective,” *Applied Sciences*, vol. 12, no. 6, p. 3201, Mar. 2022, doi: 10.3390/app12063201.

[2] R. Silva, A. Fred, and H. P. Da Silva, “Morphological autoencoders for Beat-by-Beat atrial fibrillation detection using Single-Lead ECG,” *Sensors*, vol. 23, no. 5, p. 2854, Mar. 2023, doi: 10.3390/s23052854.

[3] B. Burle, L. Spieser, C. Roger, L. Casini, T. Hasbroucq, and F. Vidal, “Spatial and temporal resolutions of EEG: Is it really black and white? A scalp current density view,” International Journal of Psychophysiology, vol. 97, no. 3, pp. 210–220, Sep. 2015, doi: https://doi.org/10.1016/j.ijpsycho.2015.05.004.

[4] F. H. Noshahr, M. Nabavi, and M. Sawan, “Multi-Channel Neural Recording Implants: A review,” *Sensors*, vol. 20, no. 3, p. 904, Feb. 2020, doi: 10.3390/s20030904.

[5] S. Rout, B. Monna, F. Pareschi, G. Setti, and W. A. Serdijn, “Spread-Spectrum modulated Multi-Channel biosignal acquisition using a shared analog CMOS Front-End,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 17, no. 4, pp. 872–884, Aug. 2023, doi: 10.1109/tbcas.2023.3317188.

[6] J. H. Park *et al.*, “A 15-Channel orthogonal code chopping instrumentation amplifier for Area-Efficient, Low-Mismatch Bio-Signal acquisition,” *IEEE Journal of Solid-state Circuits*, vol. 55, no. 10, pp. 2771–2780, Oct. 2020, doi: 10.1109/jssc.2020.2991542.

[7] A. Yaksh *et al.*, “A novel intra-operative, high-resolution atrial mapping approach,” *Journal of Interventional Cardiac Electrophysiology*, vol. 44, no. 3, pp. 221–225, Oct. 2015, doi: 10.1007/s10840-015-0061-x.

[8] H. Chandrakumar, “A 0.6 µW/channel, Frequency Division Multiplexed Amplifier for Neural Recording Systems,” M.S. thesis, University of California, Los Angeles, 2012. [Online].Available:https://escholarship.org/content/qt2ss944rw/qt2ss944rw_noSplash_49984ecc8ca80e7719b9e6b12d688967.pdf?t=nru1mv

[9] L. Dong et al., "A Multi-Channel 1.52 µVrms Front End with Orthogonal Frequency Chopping for Neural Recording Applications," 2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Bangkok, Thailand, 2019, pp. 389-392, doi: 10.1109/APCCAS47518.2019.8953173.

[10] J. Warchall, P. Theilmann, Y. Ouyang, H. Garudadri and P. P. Mercier, "Robust Biopotential Acquisition via a Distributed Multi-Channel FM-ADC," in IEEE Transactions on Biomedical Circuits and Systems, vol. 13, no. 6, pp. 1229-1242, Dec. 2019, doi: 10.1109/TBCAS.2019.2941846.

[11] R. L. Pickholtz, D. L. Schilling, and L. B. Milstein, “Theory of Spread-Spectrum Communications - a tutorial,” *IEEE Transactions on Communications*, vol. 30, no. 5, pp. 855–884, May 1982, doi: 10.1109/tcom.1982.1095533.

[12] D. V. Sarwate and M. B. Pursley, “Crosscorrelation properties of pseudorandom and related sequences,” Proceedings of the IEEE, vol. 68, no. 5, pp. 593–619, Jan. 1980, doi: 10.1109/proc.1980.11697.

[13] Vahid Majidzadeh, A. Schmid, and Yusuf Leblebici, “A 16-channel, 359 μW, parallel neural recording system using Walsh-Hadamard coding,” Sep. 2013, doi: https://doi.org/10.1109/cicc.2013.6658512.

[14] T. W. Cusick and Pantelimon Stanica, Cryptographic Boolean Functions and Applications. Academic Press, 2009, pp. 5-24.

[15] A. L. Goldberger, “PhysioBank, PhysioToolkit, and PhysioNet”, *Circulation*, vol. 101, no. 23, Jun. 2000, doi: 10.1161/01.cir.101.23.e215.

[16] Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P. C., Mark, R., ... & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220.

[17] “The normal ECG,” *The Student Physiologist*, Aug. 14, 2016. https://thephysiologist.org/study-materials/the-normal-ecg/

[18] M. Qian, K. Zhao, B. Li, H. Gong, and A. Seneviratne, “Survey of Collision Avoidance Systems for Underground Mines: sensing protocols,” *Sensors*, vol. 22, no. 19, p. 7400, Sep. 2022, doi: 10.3390/s22197400.

[19] Omid Malekzadeh-Arasteh, Ahmad Reza Danesh, A. H. Do, Zoran Nenadic, and P. Heydari, “An Analysis of CMRR Degradation in Multi-Channel Biosignal Recording Systems,” IEEE Transactions on Circuits and Systems Ii-express Briefs, vol. 68, no. 1, pp. 151–155, Jan. 2021, doi: https://doi.org/10.1109/tcsii.2020.3011180.