HIGH PRECISION CLOCK-LESS ADC USING WAVELET NEURAL NETWORKS

By

Tamer Hussein Elsalahati

Submitted in partial fulfilment of the requirements for the degree of

MASTER OF APPLIED SCIENCE

at

DALHOUSIE UNIVERSITY

Halifax, Nova Scotia

December 2017

To my loving Parents

To my Wife and my Kids

To my Sibling and Friends

These humble works are a sign of my love for you!

Table of Contents

List of Tables

List of Figures

Abstract

A Clock-Less Analog to Digital Converter (ADC) system is proposed that convert analog input to a continuous-time (CT) digital representation without sampling and then processes the information digitally without the aid of a clock. As the conventional digital signal processing (DSP) suffers from aliasing and quantization noise, in this proposition a higher precision clock-less ADC using Wavelet Neural Network (WNN). The input signal will be encoded by a delta modulator without clock into a series of non-uniformed spaced emblems when a quantization level is crossed. These emblems are processed by the DSP in CT and converted to an analog output using a custom Digital to Analog Converter (DAC) that guarantees there are no glitches in the output waveform. The ADC quantizer resolution and the required number of emblems based on the rate of change of the input signal constitute a great challenge in CT.

The CT systems are suited for burst-like signals and low power applications such as those in hearing aids, ECG for monitoring and pacemakers, and neuron sensing for implantable prosthesis processing, as with an inactive input. The CT-ADC waits for a change in the signal while dissipating no dynamic power. Also, CT-DSP offers the advantages of noise immunity and programmability as in conventional digital systems but without the use of a clock. Furthermore, no sampling is used; thus, no aliasing occurs.

In this work, a new method is proposed to realize a high precision CT-ADC converter implementing a low precision CT-ADC in the first stage, then applying the WNN technique for calibration in the second stage. In the first stage, the input signal is converted to CT digital codes. In the second stage, the quantization error or residual signal of the CT-ADC and resample DAC are calibrated by WNN to get higher precision CT-DAC. The CT-ADC possesses strong nonlinearity due to the quantization error. Therefore, WNN is employed to remove errors from the ADC converter. Also, WNN incorporates the efficient learning ability and generalization of Neural Network (NN) and the good property of localization of wavelet transform.

This dissertation presents a novel four bits continuous-time ADC, which is simulated in MATLAB by (16-level) level crossing with an ideal comparator using ($1\ V_{pp}$) and a clock resolution (R=$10^{12.35}$). The modulator has a sampling clock of 80 Mhz. It achieves a dynamic range (DR) of 230.5 *dB* with Effective number of bits (ENOB) of 38.4 bits, a SQNR of 67.2 *dB*, and a SNDR of 232.9 *dB* over 1Hz input signal bandwidth.

List of Abbreviations and Symbols Used

Symbol	Property
WNN	Wavelet Neural Network
CT	Continuous Time
ADC	Analog to Digital Converter
DSP	Digital Signal Processor
DAC	Digital to Analog Converter
NN	Neural Network
Q	Quantization Error
SNR	Signal-to-Noise Ratio
SQNR	Signal-to-Quantization Noise Ratio
SFDR	Spurious-Free Dynamic Range
LSB	Least Significant Bit
RMS	Root-Mean-Square
P_{diss}	Power Dissipation
f	Frequency
LC-ADC	Level Crossing-Analog to Digital Converter
T_q	Time Quantizer / Time Interval / Clock Resolution / Time Resolution
Emblem	Non-Uniformed Data Space that Digitalized from CT-ADC
$V_{i/p}$	Voltage Input Signal
V_{ref}	Voltage of Reference Signal
ΔT	Time Constant of CT System
SINAD/SNDR	Signal-to-Noise and Distortion
EOB	Effective Number of Bits
THD	Total Harmonic Distortion
DR	Dynamic Range
Δ	Space Level (Delta)
R	Resolution Ratio
RMS	Root Mean Square

Symbol	Property
UPDN	UP/DOWN indicator signal
BPWNN	Back-propagation wavelet neural network
FT	Fourier Transform
MSE	Mean Squared Error

Acknowledgements

It would not have been possible to write this thesis without the help and support of the kind people around me, but I will acknowledge some of whom it is possible to give a particular mention here. Above all, I would like to thank my family, all over the world, for their support and inspiration. Special thanks to my father Hussein, who supported me financially towards approaching my master's degree and for his passionate encouragement, my sincere thanks for my kind mother Mona, who loves, supports and always pray for me. I would also like to thank my fabulous sister Dalia, and my awesome brother Hazem for their generous support and compassion during my master's journey. Also, appreciative thanks for my entire family, who has given me their explicit support throughout, as always, for which my mere expression of appreciation likewise does not suffice.

I would like to thank my thesis advisor, Professor Ezz El-Masry, for his guidance in research, for invaluable lessons in intuitive thinking and clear writing, for his patience, and for providing this interesting research topic. He always allowed me the space to work, as I desired. I also thank him for his support he has bestowed throughout my entire time at Dalhousie University.

I am truly thankful to Dr. Dalia El-Dib for her assistance, perceptive technical suggestions, and for her fabulous efforts to publish the paper for this thesis. Without her passionate participation and input, the paper could not have been successfully published. Also, I am gratefully indebted to her for her valuable review and comments on this thesis.

I would also like to thank all my colleagues, too many to name here, for all their encouragement, diversions, and enriching my life experience. I am glad to have met all my fellow colleagues at Dalhousie University, for the blackboard discussions, happy hours, and shared the enthusiasm for getting our degrees.

I would like to express my appreciation to my ex-wife, Nesreen for her great supporter to get me move to Halifax and start my master degree, and wishing her a great luck at her career and life.

I would like to extend my sincere thanks to my friend Khaled Gaber, I owe Khaled a great deal of gratitude for helping me publish my work at IEEE International Conference on Microelectronics (ICM) 2016.

I would also like to thank my friends in Halifax and in Toronto for their exceptional support particularly Adel Elshenawy for being my library mate, thanks for all the good times and for pushing me to excel and succeed. Taha Darwish, thank you for being there pushing me up to achieve outstanding progress in my life, thanks for being a wonderful friend and always being so enthusiastic and supportive.

Finally, the best outcome from these past few years is finding my best friend, soul-mate, and wife. I married the best person out there for me. Sara is the only person who can appreciate my quirkiness and sense of humor. There are no words to convey how much I love her. Sara has unconditionally loved me during my good and bad times.

Chapter 1

Introduction

During the past few decades the rapid evolution of digital integrated circuit technologies has led to ever more sophisticated signal processing systems. These systems operate on a spacious variety of clock-less system signals including speech, medical imaging, sonar, radar, electronic warfare, instrumentation, consumer electronics, and telecommunications. The main effective key to the success of these systems has been the advance in analog-to-digital converters (ADCs). There are many types of ADCs but all can be classified into conventional and continuous time ADCs.

1.1 Conventional vs. Continuous- Time ADCs

Conventional ADCs (synchronous/Nyquist ADC) is based on the uniform sampling mechanism with a sampling frequency according to constant sampling clock periodically triggers the conversion regardless of input signal variations. On the other hand, continuous-time ADCs start functioning only when a change in the input signal is sensed. There are many differences and similarities between both conventional and continuous-time ADCs as described in the following subsections.

1.1.1 Energy Efficiency

In conventional systems, sampling occurs at a fixed worst-case sampling frequency rate that is determined by the highest expected frequency. Hence, when the input signal is more relaxed (periods of silence, or even lower frequency content) the high sampling rate basically wastes power. Therefore, using the non-uniformed sampling with local sampling frequency adjusted to the signal properties is the best solution to avoid this issue. The non-uniformed sample is generated when the input change is enough to cross the precise modulation levels, thus saving average power in the ADC. In summary, conventional uniform sampling constantly generates the samples

from the sensed signal, resulting in a waste of system energy. Consequently, from the system perspective, conventional ADCs are less power-efficient for sparse signal recording if compared to continuous ADCs [1]-[4].

1.1.2 Under-sampling and Oversampling

Under-sampling is essentially sampling too slowly, or sampling at a rate below the Nyquist frequency of the signal. Under-sampling leads to aliasing, and the original signal cannot be properly reconstructed. However, under-sampling also requires less memory so that it may be useful in certain applications. Oversampling is sampling at a rate exceeding twice the highest frequency component of the signal and is usually anticipated. Since real-world signals are not perfectly filtered and frequently consist of frequency components greater than the Nyquist frequency, oversampling can be used to expand the folding frequency (one-half the sampling rates) so that these undesirable elements of the signal do not alias into the passband. Oversampling is also necessary when trying to capture fast edges, transients, and one-time events.

1.1.3 Aliasing signals

Aliasing happens when an input signal has frequency components at or higher than half the sampling rate. If the signal is not properly filtered to eliminate these frequencies, they will display as spurious lower frequency components or aliases that cannot be detected from valid sampled data. These errors in data are actually at a higher frequency, but when sampled, appear as a lower frequency, and thus, false information. The Nyquist Theorem states that at least two samples are needed per cycle for the signal to be recovered from its samples. That is; If the sampling frequency is below Nyquest rate aliasing occurs and if the sampling frequency is higher than Nyquest rate false images appear as mirror images of the original signal around the Nyquist frequency. This situation is called "aliasing back" [5].

For this reason, the conventional system is not working well with the sparks signals. If the frequency is set very high, it will waste the system energy, and if the frequency (f) is low, it will cause aliasing and glitches on the output signal.

1.1.4 Quantization

The continuous-time ADC has no quantization in time because the occurrences of samples may arrive anytime, so the time interval is continuous, and it always has a real-time value. However, the amplitude of samples is exact because a conversion is triggered by the crossing of a given amplitude value. The design of such CT-ADC is entirely different from traditional Nyquist ADCs. In the continuous-time ADC the input signal is quantized by the quantization levels that are spaced in amplitude by 1 Least Significant Bit (LSB) apart. However, their spacing in time is unknown and depends on the input signal behavior (slope). This is the main difference if compared to conventional ADCs, where the amplitude difference between consecutive samples is unknown, but the time space is fixed sampling period. This is obviously wast of power when the input signal is relaxed (lower frequency content, or even period of silence) and the fixed sampling frequency is high.

1.1.5. Performance Measures

Despite the variety in ADC's in general, their performances can be summarized by a relatively small number of parameters such as stated resolution (number of bits per sample), signal-to-noise and distortion ratio (SNDR) or signal-to-quantization noise ratio (SQNR) and quantization error (Q). Those parameters will be discussed in detail in Chapter 5, and then will be used to evaluate the results of each of the presented system and compare the performance results to previously published work.

1.2 Advantages of Clock-less ADC

The new clock-less ADC based on level-crossing sampling has the following promising advantages:

1) Low-amplitude and low-frequency input signals will be sampled in lower sampling rate and less densely in time than high frequency and high-amplitude input signals. Hence, the input signal will be sampled when the inputs cross the delta-modulation levels [6].
2) The output signal spectrum has no aliasing and contains only the harmonics of the input signal without the quantization noise [7]. Therefore, SNDR of Level crossing-ADC (LC-ADC) can exceed the theoretical limit of conventional systems with the same resolution in amplitude.
3) No external clock is required for clock-less ADC/Digital-Signal-Processing (DSP)/Digital-to-Analog Converter (DAC) system, which will save the fabrication size and power consumption, as the time quantizer (T_q) will count only the time for sampled signals, when the input signal crosses the threshed levels [7].
4) The output signal of the continuous-time clock-less ADC will be discrete in amplitude, and continuous in time, as the time resolution is infinite [7]. This diminishes quantization noise caused by the variability in the amplitude estimation.
5) In a clock-less ADC, the input signal is seemingly sampled at the accurately defined period, and the signal amplitude is estimated to equivalent digital value, which can be more precise when using a low supply voltage.
6) The clock-less system produces lower electromagnetic interference emission [6]-[8].
7) As the clock-less system is running without any trigger (external clock), the sudden input signal will be sampled and processed at the real-time without waiting for any trigger by the clock, and an output signal will be delivered at the real-time.

1.3 Applications of Clock-less ADC

ADCs are widely used in implantable biomedical data acquisition systems, hearing aids, pacemakers, Electrocardiography (ECG) monitoring systems, wireless sensor networks and lots of other applications where low power/energy is a major concern.

In such systems, the power consumed during transmission usually dominates and is proportional to the overall data rate. Moreover, many bio-signals, for example, are sparse in the time domain, encompassing both long periods of low-frequency content and short periods of high-frequency information. These applications demand signal processing and transmission with as little power dissipation as possible, such as biomedical devices, wireless sensor networks, and portable communication devices. For these systems, asynchronous ADCs, also referred to as level crossing or continuous-time ADC, is most suitable.

The continuous-time system must be low-cost, low-noise, reduced-sized, and especially low power because they are always powered by batteries or remotely powered. As mentioned, the ADC block is the main key component of such systems. It is very critical for any suggested clock-less design to adhere to all these constraints. A promising alternative ADC for low power applications is called level-crossing continuous-time sampling, which is based on a non-uniformed data sampling technique called emblems, as the samples are generated by the signal crossings of the threshold levels while a timer measures the time between two consecutive samples. For example, low-amplitude and low-frequency inputs are sampled less densely in time than high-frequency and high-amplitude inputs; no aliasing occurs. Therefore, a much lower average sampling rate is achievable for several applications.

For example, the level-crossing sampling has been used in biomedical devices like hearing aids or in wireless ECG sensors, for continuous heart condition monitoring. In these kinds of applications, it facilitates continuous working for hearing or recording of ECG signal without causing extra inconvenience or disturbance to the patient. In such devices and applications a challenged ADC design is required that keeps the usage of wireless transceiver at a minimum with a decent size, weight, and power consumption that grants ability for extended battery life or self-powered sensor [9]. Hence, it becomes progressively important to focus on power optimization at all design levels for these kinds of sensors and devices.

In summary, clock-less systems operate on a wide variety of continuous-time signals include speech, medical imaging, sensors, sonar, radar, electronic warfare, instrumentation, consumer electronics, and telecommunications.

1.4 Thesis Overview

The new type of ADC proposed in this paper presents an asynchronous ADC design with level crossing data sampling that operates without any external clock. Nevertheless, an internal timer will be created whenever the input signal crosses the quantization levels. In fact, the input signal amplitude variations trigger the conversion process and drive the whole circuit. With this principle, when the analog input signal is quiet, the circuit is asleep, and there is no useless activity. Whereas, in the conventional systems the samples are triggered and recorded according to the sampling rate of time equispaced at either there is input signal activities or not.

Furthermore, this work will discuss the new technique of continuous-time ADC using a wavelet neural network. This system approaches a higher quantization resolution, without adding extra level crossing delta modulation. The massive improvement of the signal quality will preface to increase the applications usages for continuous-time technology. The target of the new design is designing a four-bit ADC (16-level) level

crossing with maximum input signal bandwidth of 20KHz. Such specifications make the new LC-ADC suitable for both voice and biomedical applications. The SNDR performance should be achievable without adding extra comparators competent by using WNN as adaptive resolution.

Chapter 2 presents the development of circuit level designs for ADC/DSP/DAC that have been published in the literature. Chapter 3 illustrates the WNN technique that is used for the proposed design to obtain a higher precession for targeted applications. In Chapter 4, the new contributions of this work and its architecture are outlined. Chapter 5 describes the ADC performance metrics and the measurement results of the proposed design. Finally, the conclusion and design performance comparisons to previously published work in the literature and suggestions for future work to improve the continuous-time output signal are outlined in Chapter 6.

Chapter 2

Clock-less System

Clock-less ADC systems (asynchronous) have been heavily researched in the last decade. Some researchers focused their studies on developing the circuit level of the CT system by reducing the number of components or by replacing them with other more efficient components aiming at reducing the circuit size and the power consumption. Others focused on enhancing the system level of the CT to reach a better scheme with higher resolution and more efficient systems that could be utilized in multiple applications. In this thesis, we study the system level of the CT-ADC system to achieve a higher resolution; we use MATLAB to build the structure of a CT-ADC system using WNN. In this chapter we present the main concept and topology for Clock-less ADCs/DSPs/DACs which compose the core techniques for a CT system.

2.1. Clock-Less ADC System

The CT ADC is the main stage of converting the analog signal to continuous digital signal. It basically depends on the delta modulation (DM) to detect the changes of the input signal by creating level-crossing sampling [7], [8] and [10].

As shown in Figure 2.1, when the input signal *VX(t)* crosses the pre-defined quantization reference levels or DM levels (dotted horizontal voltage levels), it gets quantized to $Vx_q(t)$. $Vx_q(t)$ is composed of non-uniformed samples, where each sample is called Emblem, Event-Driven, Data Token or Continuous Time (CT) Sample. Each Emblem is described by its time and amplitude (t_i , $Vx(t_i)$). An input signal with a faster slope generates more emblems compared to another input signals with a slower slope.

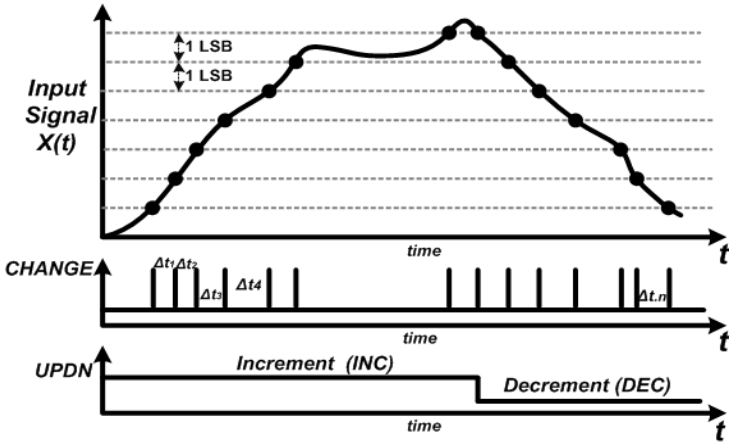

Figure 2.1: Level crossing quantization technique showingInput signal, Continuous-time change, Continuous-time UP/DOWN (UPDN)

The quantized signal $Vx_q(t)$ can be digitized in several ways including the continuous-time binary bits in a flash ADC or using data tokens consisting of change signal CHANGE and direction UPDN as in delta modulation. Both indicators, change, and direction, are dependent on the input signal behavior. Change signal is an active signal generated when the input signal *Vx(t)* crosses the quantization levels. It represents the time quantizer (T_q) of each generated Emblem. The direction indicator depends on the slope of the input signal; it is logic HIGH for a positive slope and is logic LOW for a negative slope. Both change and direction signals represent the digitized continuous-time signal.

In the conventional ADC, all the comparators are powered off by default until the sampling time arrives, then all comparators will be enabled to detect the unknown input frequency within whole full-scale voltage. Whereas in the clock-less ADC, all the comparators are powered off except for only two comparators (which need to be on) at any given time. The first comparator is the one that triggered the previous input signal from the last cycle, and it will compare the new input signal *Vx(t)* versus the next quantization level above, and the second comparator is comparing *Vx(t)* versus the next quantization level below. At inactive input signal, the clock-less ADC waits for a change in the signal while dissipating no dynamic power; which is the main advantage for the clock-less ADC, making it well suited for burst-like signals for low power applications.

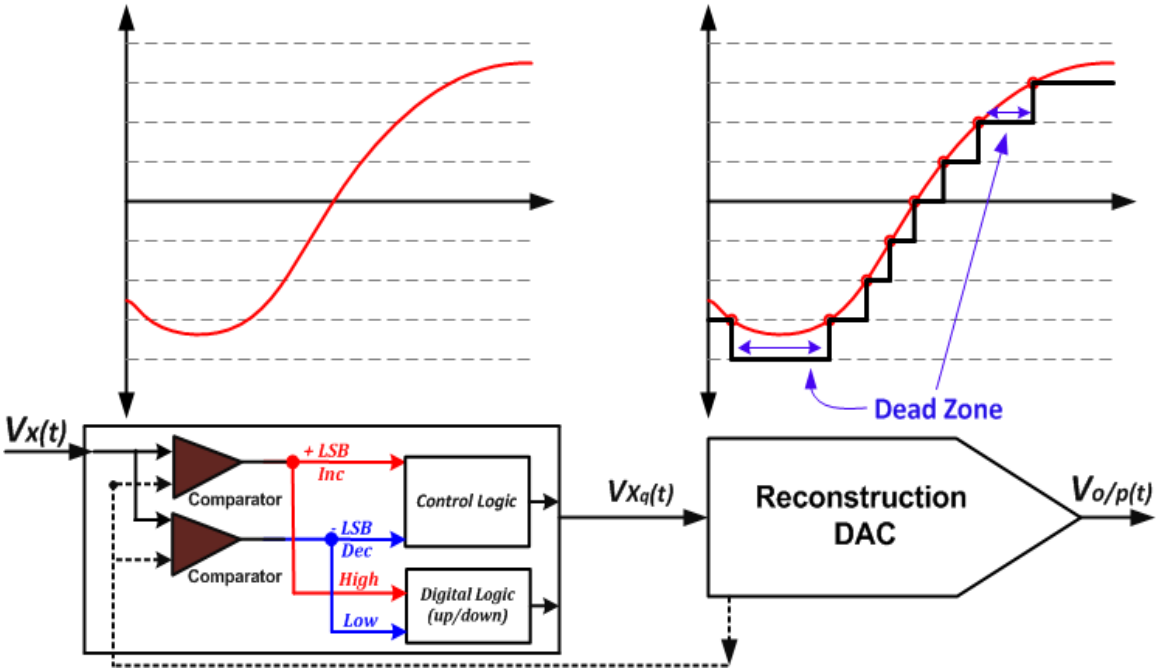

Figure2.2: The digital output $x_q(t)$ of the clock-less ADC. The continuous-time DAC reconstruction of the analog input from the samples

The resolution of the reconstructed signal CT-DAC is controlled by the slope of the input signal, which increases when the input signal crosses consecutive quantization levels rapidly and reduces when the input signal takes a longer time to cross any consecutive quantization levels. As Illustrated in Figure 2.2, the resolution of the clock-less system depends on the slope of the input signal; when the input signal crosses any consecutive quantization levels slowly, this will result in something called Dead Zone in the output signal in CT-DAC. Dead Zone is a blanked area which hides the real behavior of the input signal between any two consecutive quantization levels (Figure 2.2). The Dead Zone will be discussed further in Section 4.1.

The main block diagram for the CT-ADC system has been examined and implemented in [7], [10]. It is composed of two comparators (to create each pair of consecutive quantization levels), Control Logic, Digital Logic blocks (to record the signal behavior) and clock-less-DAC to feedback the comparators with the reference signal (V_{ref}) (Figure

2.2). In Chapter 4, a detailed description of each block in the clock-less ADC will be presented along with the enhanced WNN system.

The circuit design for clock-less ADC is very challenging. As a result of clock absence, the design of the comparators is difficult. The comparators compose the most critical block in the architecture, as its delay dominates the loop delay. An important consideration is the fact that comparison time depends on the rate of change of the input signal (Slope) and the comparator delay decreases with an increasing input slope. The clock-less ADC comparator has been presented in [7], [10]. The clock-less comparator is implemented as an amplifier structure, which is chosen for its ability to operate at low supply voltages. Two inverters are added at the output to allow for rail-to-rail (logic level) output swing due to the fact that the comparator output will interface with the control/digital logic and to prevent an increase in the decision time. The decision time of the comparator will depend on the slope of the input when it crosses the comparator threshold. Minimizing the time error and canceling comparator offset are discussed in [7], [10].

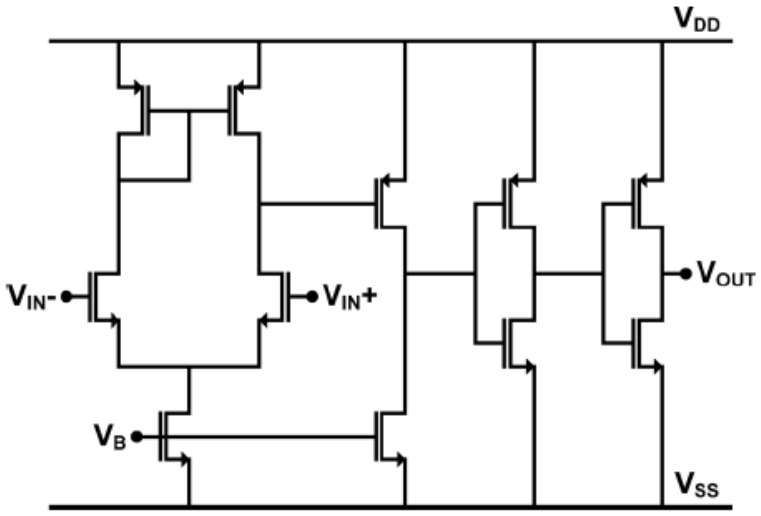

Figure 2.3: Clockless comparator [10]

Recently, new circuit designs of Clock-less ADC have been invented to improve the ability of the comparators to tolerate CT GHz systems as in [11]-[12].

2.1.2 Adaptive Resolution

The adaptive resolution is demonstrated in [8], [10], [13]. The main goal of adding an adaptive resolution is to increase the number of samples taken and obtain better SNDR to realize higher signal resolution. The adaptive resolution is varied dynamically depending on tracking the input signal behavior and reconstructed output signal due to the minor error between the input signal and sampled output. The signal improvement may lead to data compression, power and bandwidth saving, which will be well suited for clock-less DSP and their applications.

The adaptive resolution should be capable of efficiently sampling fast and slow varying signals; as well, the time difference between consecutive emblems should be long enough to match up with loop delay. In Chapter 4, we illustrate the two stages of adaptive resolution that have been developed to improve the reconstructed output signal ($V_{Combined}$) Consequently, the reconstructed output signal ($V_{Combined}$) is the input signal for WNN.

2.2. Clock-less DSP System

Clock-less DSP systems which perform DSP in CT are attractive for some applications like remote sensors, biomedical implants, hearing aids, audio and speech processing and telecommunications. Therefore, many researchers presented the implementation of CT-DSP considering the implementation costs and power consumption as in [5], [7]-[8], [12], [14]-[16].

Clock-less DSPs are well suited for real-time processing; the signals they process and products cannot be stored in a digital medium. Hence, the CT is a clock-less system and the time is not quantized, the respective signals are processed continuously in time without a clock, but they are discrete in amplitude. The time interval (T_q) (time granularity) is used to synchronize the digital signal samples at the CT-DSP signals. Thus, the signals within the CT-DSP system cannot be stored in the finite-resolution hardware of a memory, which limits the applications of the CT-DSP system mainly to the real-time

processing. Nevertheless, CT-DSP systems have some interesting properties, making them very attractive for several applications. The advantages of clock-less DSPs systems are:

(1) Clock-less DSP systems have no clock, and their power dissipation intensely decreases when input activity decreases.

(2) The entire clock-less system (ADC/DSP/DAC) does not suffer from aliasing. This reduces the in-band error power, which will have an improved in-band SDR over the conventional DSP system. Therefore, clock-less DSP systems have a significant performance advantage over the conventional DSP systems.

(3) Clock-less DSP systems still hold the benefits of digital techniques such as programmability and noise immunity.

(4) Clock-less DSP systems help keep electromagnetic emissions low as the advantage of no clock is used in signal processing.

(5) Clock-less DSPs react immediately to input changes, in contrast to the conventional DSP systems which may not catch such changes until the next sampling instant.

All these advantages make clock-less DSPs especially suited for real-time applications and high-speed digital control loops [17].

Many researchers studied the circuit design of the CT-DSP to improve their capabilities and reduce the application's limitation, especially in the GHz-range application [12]. As such, the conventional digital systems suffer from aliasing and require a complicated anti-aliasing filter or enormously high clock speeds with high power dissipation. The CT-DSP is alias-free, offers activity-dependent power dissipation and has lower EMI emissions. In [18], the author presented a new circuit design of CT-DSP and consider it as a candidate for wideband GHz low dynamic-range applications, like those found in pulse radio, spectrum sensing, and channel equalization. This scheme makes programmable CT digital filtering possible in the 0.8 to the 3.2 GHz range with graceful degradation at higher frequencies and no aliasing.

One of the main challengings of CT-DSP is the realization of the delay elements which must be implemented as quasi-continuous time delay lines, that requires large chip area. Since the time interval between the samples must also be preserved, the storage elements must be realized as continuous time delay lines. In [16], the author presented an architecture that allows reducing the implementation costs and power consumption of these elements. This has been achieved by granularity reduction of the delay elements (reducing the time interval), without sacrificing performance. The researcher used a technique to lessen the granularity by summing up the fixed time-intervals and replaced with the resulting summation, which reduced the total number of tokens and consequently the hardware complexity as well the power consumption.

In Summary, multiple CT-DSP designs have been studied to improve the circuit level by reducing the power consumption, time delay elements, and costs. In this work, we improve the system level of CT-DSP to realize higher signal resolution by using wavelet neural network as it presented in Chapter 4.

2.3 Clock-less DAC

Data converters from analog-to-digital (ADC) or from digital-to-analog (DAC) play a vital role in mixed signal circuits, allowing the analog signals to communicate with the digital systems and reassemble the digital signals to the analog world. These operations are necessary for sensor interfacing, biomedical, speech applications, and actuators interfacing. High-speed and high-resolution DACs are becoming a global requirement and are widely used for video, audio and communication systems. DACs are also required to generate precise spur for sensors and biomedical equipment. The output signal of CT-DACs connect to off-chip subsystems, which are generally time continuous. The time intervals where the signal is not valid are not allowed. The CT-DAC has been presented in many research [7], [10], [19]. They used the resistor string, hybrid switched-capacitor/resistor-string, or switched-capacitor to implement the CT-DAC.

Each design has its advantages and disadvantages according to application and resolution requirements.

The CT-DAC implemented in [7] using a shift registered and resistor-string DAC. Mainly, the shift register has only one of the 256 signals at logic *1,* and all others are at logic *0*. these signals apply directly to the switches that select the voltages from the resistor string. This technique is used because the input signal is continuously tracked and the increment (INC) and decrement (DEC) values of delta modulation are changing by one level only according to the estimated value of the level crossing levels. This scheme does not scale very well if the CT-ADC needs an extra bit of resolution. In fact, the size of the shift register and resistor string DAC will double in size with an increase in wiring complexity, power dissipation, and area requirements. The resistor-string requires $2^{\#\text{of bits}}$ resistors and $2^{\#\text{of bits of}}$ switches to be implemented. Hence, this design is not efficient for this work, because it will need a large number of resistors for the high precision output signal of the CT-WNN system.

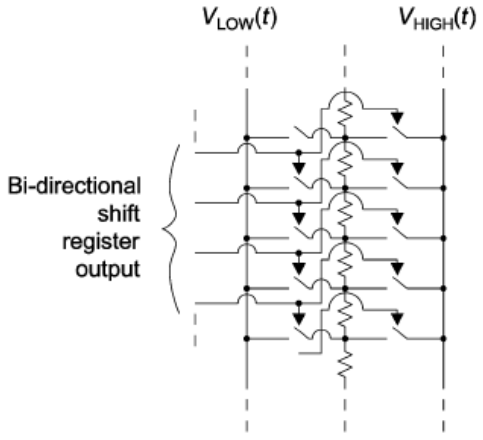

Figure 2.4 : Schematic of resistor string DAC [7]

The CT-DAC is implemented in [10] with 10-bit fully-differential hybrid switched-capacitor/resistor-string. Using the switched-capacitor decreases the accuracy of the output over the time due to the current leakage, which is referred to as output drift. To

eliminate output drift altogether, the DAC could be implemented as a resistor string. Therefore, the author mixed his design between the resistor string and switched-capacitor to reduces the static power consumption and output drift. This architecture is qualified for a maximum sample rate of approximately 50 Ks/s, or 100 Ks/s in the fully integrated design. Consequently, this scheme does not meet the requirements of the higher precision CT-ADC/DAC using WNN, as it still uses the resistor string consuming size in fabrication and has a low sample rate at the output signal.

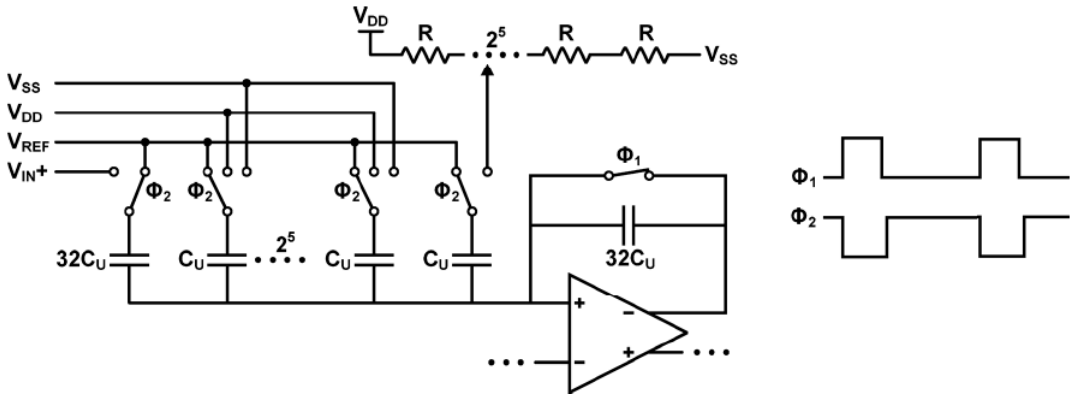

Figure 2.5: 10-bit fully-differential hybrid switched-capacitor/resistor-string DAC and example timing waveform [10]

In fact, the switched-capacitors DACs require TRACK and HOLD (T/H) block. In [19], The CT-DAC is executed on a 12-bit prototype and designed with 0.18µm CMOS technology. The authors presented an alternative architecture for capacitive DACs, which is capable of inventing an offset free output signal in all phases of the conversion cycle. The authors used an effective feedback chain circuit to control the op-amp based integrator, which can be designed to deliver the required current to the load directly.

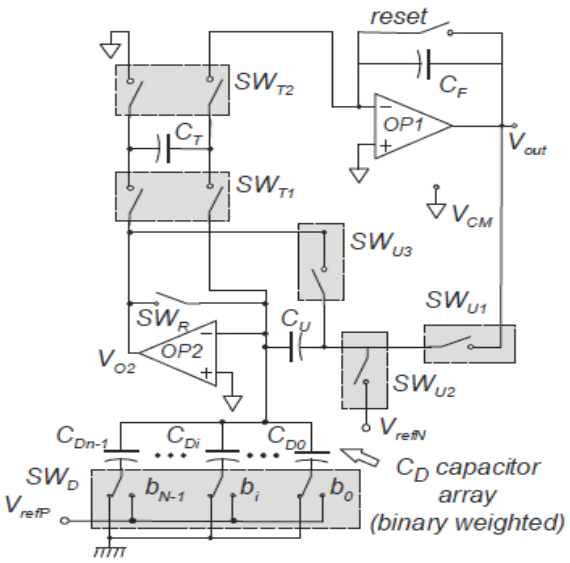

Figure 2.6: Block diagram of the proposed DAC architecture [19]

This architecture is capable of producing a continuous-time output free from errors due to offset and low-frequency noise contribution. The simulation results are obtained at a clock frequency of 2.5 MHz and results in a total time resolution of T=2.4µs. This high resolution will be suitable for this work's proposed higher resolution CT system using WNN. As a result, the switched capacitor DAC is a well-suited technique to use to implement the CT-DAC for the work presented in this thesis.

Chapter 3

Wavelet Neural Network

This chapter introduces the main concept of WNN. First, the importance of wavelet transform (WT) and the theory of continuous-wavelet transform (CWT) are described. The remainder of the chapter describes the training process for WNN to reach minimum error factor. Thus, this chapter covers the factors behind choosing WNN to enhance the CT-ADC signal.

3.1 Wavelet Transform

Wavelet transforms (WTs) are based on small waves, called wavelets, which provide frequency and time information concurrently. Therefore, it is a significant tool for signal representation. Presently, it is being used in several applications like signal processing, image processing, and data compression. Wavelets allow complex data such as images, music, speech, and patterns to be decomposed into fundamental forms, called wavelet coefficients, *C*, at different positions and scales. These wavelet coefficients represent a family of wavelets that are generated from a single function called “mother wavelet” by translation and dilation operations. The data are subsequently reconstructed with high precision [20].

WTs are mathematical functions that decompose the data into different frequency components, then study each component with its resolution matched to its scale. WTs have many advantages over traditional Fourier methods in analyzing burst-like signals where the signal contains discontinuities and sharp spikes. Therefore, it is a good tool to determine where the low-frequency area and high-frequency area is. Also, it can tell us how a given signal changes from one period to the next, as the window size, can be modified to determine more accurately either time or frequency, as shown in Figure 3.1.

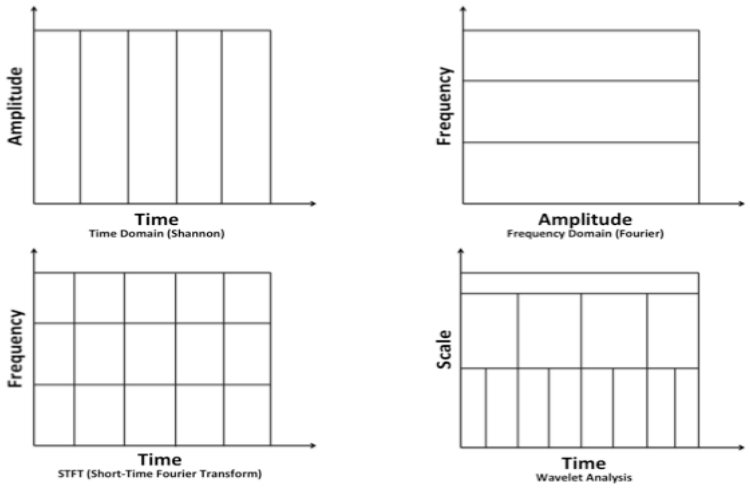

Figure 3.1: Comparison between wavelet transform window and other Fourier transforms (FT)

The time resolution and frequency resolution are inversely proportional, that means the wavelet transform at high frequencies will provide rapid changing details (good time resolution). All of this leads to a compressed wavelet with a low scale (poor frequency resolution). Consequently, the wavelet transforms at low frequency will have slow changing (limited time resolution) that will present a stretched wavelet with high scale (good frequency resolution). In short, the narrow box gives a better resolution, as shown in the Figure 3.2.

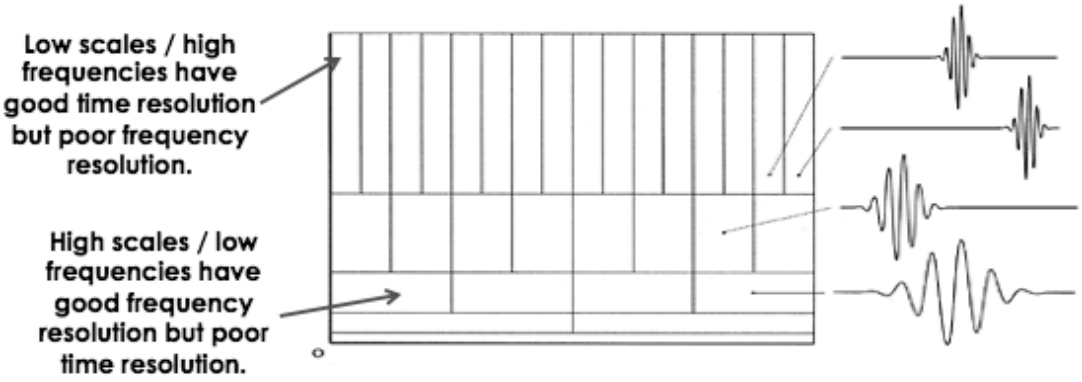

Figure 3.2: Wavelet signal analysis at low frequency and high frequency

As our scope is on continuous-time signal, we will study the continuous wavelet transform (CWT) technique that proved its suitability for non-stationary signals.

3.2 Continuous Wavelet Transform

The CWT is defined as the summation of overall time domain evidence of the signal multiplied by scaled and shifted positions of the mother wavelet function. Although CWT uses discretely sampled data, by dividing the overall input signal into discrete sampled windows, the continuous shifting process is operating smoothly across the full domain of the input signal analyzed. The scaling can be defined anywhere between the minimum (original signal scale), and the maximum signal is chosen by the user, thus giving a much greater resolution [20].

Either real or complex analytic wavelets can be used. Complex analytic wavelets can separate amplitude and phase components, while real wavelets are often used to detect sharp signal transitions. The result of the CWT is a set of wavelet coefficients, *C*, which is a function of scale and position. Multiplying each coefficient by the appropriately shifted and scaled mother wavelet returns the essential wavelets of the original signal. As the original signal can be represented in terms of a wavelet expansion, by using the wavelet coefficients in a linear combination of the wavelet functions, data operations can be performed using only the corresponding wavelet coefficients [21].

The CWT is used to transform the quantized CT signal and studies their time and frequency localization simultaneously. Meanwhile, the translation parameter *b* is controlling the translation (shifting) of function by inspecting the quantized CT signal at different time steps of time localization. Also, the dilation (scaling) parameter *a* is controlling the scale of function by inspecting the quantized CT signal at various frequencies. The idea is to change the scaling and shifting parameters such that we can measure how the wavelet function, $\Psi_{a,b}(t)$ fits the signal *X(t)*. Thus, *b* is related to the location of the window, as the window is shifted through the signal. It corresponds to time information in the wavelet transform. Whereas, *a* corresponds to frequency information. Hence, low frequencies (large scales) expand the signal and provide non-

detailed information about the signal, whereas high frequencies (low scales) compress the signal and provide detailed information about the signal [22].

The CWT is defined as:

$$\mathbf{CWT}_{\boldsymbol{x}}(\boldsymbol{a},\boldsymbol{b}) = \int_{-\infty}^{\infty} \frac{\mathbf{1}}{\sqrt{|a|}}\ \boldsymbol{\Psi}^{*}(\boldsymbol{t})\, \boldsymbol{X}(\boldsymbol{t})\ \boldsymbol{dt} \qquad \textbf{(3.1)}$$

$$\mathbf{CWT}_{\boldsymbol{x}}(\boldsymbol{a},\boldsymbol{b}) = \int_{-\infty}^{\infty} \frac{\mathbf{1}}{\sqrt{|a|}}\ \boldsymbol{\Psi}^{*}\left(\frac{\boldsymbol{t}-\boldsymbol{b}}{\boldsymbol{a}}\right)\ \boldsymbol{X}(\boldsymbol{t})\ \boldsymbol{dt} \qquad \textbf{(3.2)}$$

Where *X(t)* is the input signal to be analyzed, $\Psi_{a,b}(t)$ is the mother wavelet scaled by *a* and shifted by *b*.

$$\boldsymbol{\Psi}_{\boldsymbol{a},\boldsymbol{b}}(\boldsymbol{t}) = \frac{\mathbf{1}}{\sqrt{\boldsymbol{a}}}\ \boldsymbol{\Psi}\left(\frac{\boldsymbol{t}-\boldsymbol{b}}{\boldsymbol{a}}\right) \qquad \textbf{(3.3)}$$

Where *a* >0 and $= -\infty < b < \infty$

In the time domain, the position of the wavelet is given by its translation *b*; while its scale *a* gives its position in the frequency domain. Therefore, the WT maps the original series into a function of *a* and *b*, which gives us simultaneous information on time and frequency. The formulas of the WT and the (FT) are very similar. Nevertheless, the main difference is that the FT does not have a time localization parameter and has *cosine* and *sine* functions instead of a wavelet function.

3.2.1 Mother Wavelet

The main issue when using the WT is the choice of the appropriate mother wavelet, $\psi(t)$. This choice depends on the goal of study and the characteristics of the analyzed signal [23]. The correct choice depends on what information to be extracted from the analyzed signal, as each wavelet function characterizes different features of the signal. This section discusses the seven main mother wavelet functions that build a wave function library; their expressions are as follows:

1) Haar wavelet function:

$$\psi(t) = \begin{cases} 1, & 0 \le t \le \frac{1}{2} \\ -1, & \frac{1}{2} \le t \le 1 \\ 0, & other \end{cases} \quad (3.4)$$

2) Gaussian wavelet function:

$$\psi(t) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-t^2}{2}\right) \quad (3.5)$$

3) Morlet wavelet function:

$$\psi(t) = cos\ (1.75t) \exp\left(\frac{-t^2}{2}\right) \quad (3.6)$$

4) Mexican Hat (Mexihat) wavelet function:

$$\psi(t) = c\ (1-t^2) \exp\left(\frac{-t^2}{2}\right) \quad c = \frac{2}{\sqrt{3}}\ \pi^{-1/4} \quad (3.7)$$

5) Shannon wavelet function:

$$\psi(t) = \frac{\sin \pi\ (t-1/2) - \sin 2\pi\ (t-1/2)}{\pi\ (t-1/2)} \quad (3.8)$$

6) Meyer wavelet function (approximate formula):

$$\psi(t) = 35t^4 - 84t^5 + 70t^6 - 20t^7 \quad (3.9)$$

7) Wavelet function GGW constructed by the authors:

$$\psi(t) = sin(3t) + sin(0.3t) + sin(0.03t) \quad (3.10)$$

Table 3.1: Prediction results of different mother wavelet function

Wavelet function	Haar	Gaussian	Morlet	Mexihat	Shannon	Meyer	GGW
MSE	2.10×10^{-2}	1.03×10^{54}	1.23×10^{-3}	1.27×10^{-3}	1.11×10^{55}	3.90×10^{57}	1.60×10^{-2}
Running time (s)	4.74	4.76	4.85	4.88	4.84	5.24	4.88

The mean squared errors (MSE) for the mother wavelet functions are as follows in descending order: Meyer> Shannon> Gaussian > Haar> GGW> Mexican Hat> Morlet, as shown in Table 3.1 [24]. The running time and MSE of Morlet and Mexihat are only slightly different, but the MSE of Shannon, Meyer, and Gaussian are extremely high. For that reason, the Mexihat or Morlet are the most popularly mother wavelet functions used to construct wavelet neural network. In this thesis it was decided to use the Morelet mother wavelet function for our design because it is faster at run time, and has better localization properties in both time and frequency domains.

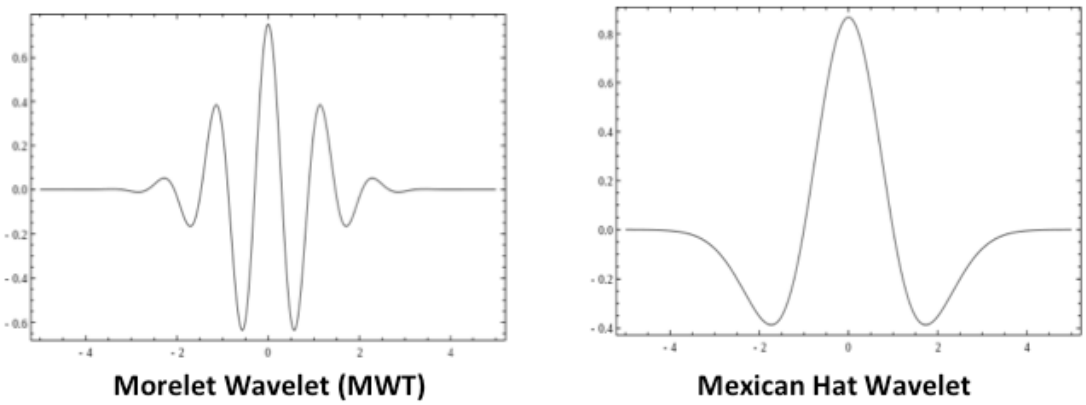

Figure 3.3: waveform of Morelet wavelet and Mexican Hat wavelet

The Morelet wavelet is a plane wave modulated by a Gaussian function. The functional set obtained on the basis of the Morelet wavelet is well localized in both time and frequency domains. With raising the value of wavelet parameter, the resolution in frequency domain increases, whereas the time localization is reduced. Therefore, CWT uses different scales to achieve high resolution in frequency and time domain [25].

3.2.1 Continuous Wavelet Transform Algorithm

The main steps to create continuous wavelet transform are presented as follows [22]:

Step 1. Use one of the mother wavelet functions $\psi(t)$, for example Morlet wavelet, and compare it to a small segment (window) at the start of the original input signal *x(t)*.

Step 2. Calculate the wavelet coefficient (C) that illustrates how closely correlated the wavelet function is with this section of the input signal *x(t)*. A higher wavelet coefficient reflects a higher similarity [22]. For example, if the signal energy and the wavelet energy are equal to one, that means the corresponding *C* may be interpreted as a correlation coefficient.

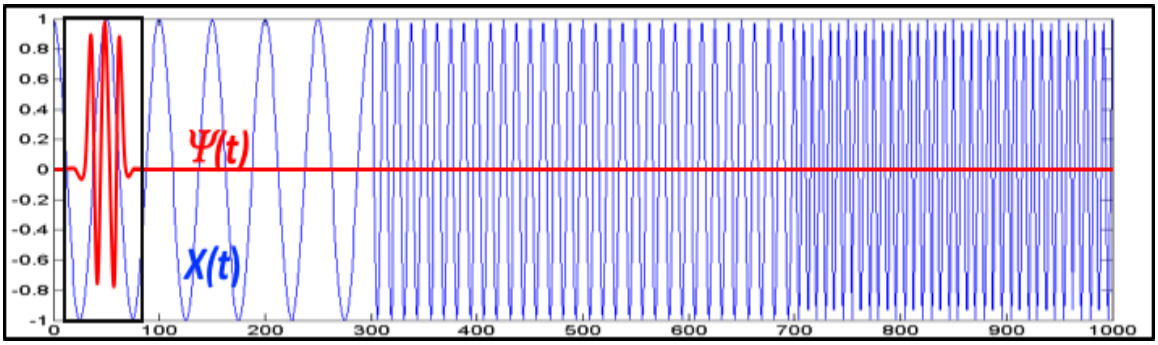

Figure 3.4: Step 2 of CWT algorithm x(t): Input signal $\psi(t)$: wavelet function

For example, the wavelet coefficient in Figure 3.4 is C=0.0302, which is very low. As described, the CWT coefficients explicitly depend on the analyzing wavelet. Therefore, the CWT coefficients are different when the CWT for the same signal is computed using different wavelets [24].

Step 3. Shift the wavelet function to the right and repeat steps one and two until it has covered the entire signal.

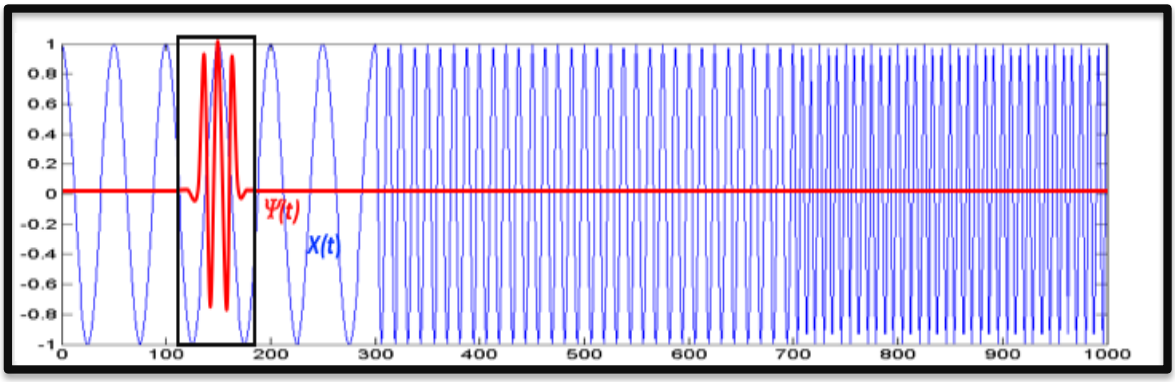

Figure 3.5: Step 3 of CWT algorithm x(t): Input signal $\psi(t)$: wavelet function

Step 4. Rescale (stretch) the wavelet function and repeat steps 1 through 3.

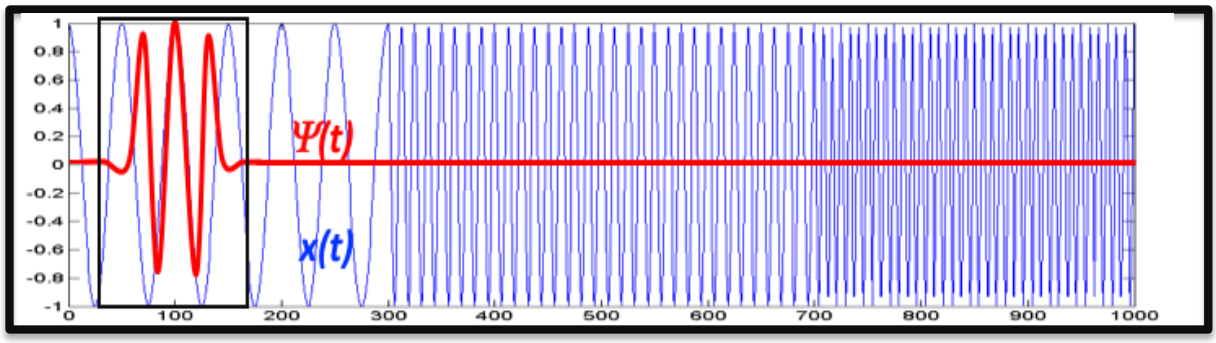

Figure 3.6: ***Step 3 of CWT algorithm*** **x(t):** ***Input signal*** **ψ(t):** ***wavelet function***

Following to the same example, the wavelet coefficient from step 4 is C=0.3267.

Step 5. Repeat steps one through four for all scales until it is done, then we will have all wavelet coefficients generated at different scales by different sections of the signal. Those wavelet coefficients can then constitute the regression of the input signal. These five process steps will be aborted if the correlation coefficient is between wavelet function $\psi(t)$. Moreover, the input signal *x(t)* is equal to 1 or (1 – resolution number). For example, if the resolution of the input signal is 0.05 and the correlation between the input signal and wavelet function is 0.95 or higher, then the process of continuous wavelet transform will be terminated, and the regression signal will be that signal that correlated 0.95 with the input signal.

3.3 Neural Networks

Neural networks are mainly devices of parallel and distributed processing components for any statistical assumptions. They consist of many interconnected neurons (hidden layers), for which the associated weights determine the strength of the signal passed through them. There is no parametric procedure, exact structure is assumed theoretically; rather, the strengths of the connections are processed in a way

that captures the essential features in the data [21]. There is a learning algorithm called Gradient Descent Algorithm that will be assigned to do iterative algorithms. The main power of neural networks accrues from their capability for universal function approximation. There are many studies that have shown that neural networks with one-hidden-layer can approximate randomly any continuous function very well, including the function's derivatives. Also, the recent academic researchers attracted to the gold feature of the neural network in nonlinear modeling techniques, with neural networks assuming a prominent role, which proven to be a powerful tool for modeling and fitting nonlinear systems using numerical data.

There are two basic categories of neural networks: Feed-forward networks and back propagation networks. A feed-forward network is a forward flow of data from the input elements to the output units. The processing of the data can be extended to multiple layers of hidden units, but there are no feedback connections. Thus, there are no connections from the outputs of the units to the inputs of the elements of the same or previous layers, whereas, the back-propagation networks have a feedback connection. [26] In this paper, the wavelet function replaces the role of sigmoid function in the hidden unit of the standard feed-forward neural network as we shall discuss next.

3.4 Wavelet Neural Network

Wavelet networks are the result of a new technique of networks that combine sigmoid neural networks and wavelet analysis. Wavelet networks have been used with great success in several applications. On the one hand, wavelet analysis has proven to be a valuable tool for analyzing a wide range of time series. Thus, it has been successfully used in image processing, signal de-noising, density estimation, signal and image compression, and time-scale decomposition. Hence, wavelet analysis is a powerful tool for representing nonlinearities [26]. The major drawback of wavelet analysis is that it is limited to applications of small input dimension. The reason for this is the structure of a wavelet basis is computationally expensive when the dimensionality

of the input vector is relatively high [27]. On the other hand, neural networks have the capability to approximate any deterministic nonlinear process, with slight knowledge and no assumptions regarding the nature of the process. However, sigmoid neural networks have a series of drawbacks. The initial values of the neural network's weights are chosen randomly. Random weight initialization is conveyed by extended training times. Furthermore, when the transfer function is sigmoidal, there is always a significant chance that the training algorithm will converge to a local minimum.To conclude, there is no analytical link between the specific parameterization of a sigmoidal activation function and the optimal network architecture.

Wavelet networks were proposed as a substitute to feed-forward neural networks, which would improve the weaknesses mentioned above associated with each method [28]. The wavelet networks are a generalization of radial basis function networks. Thus, wavelet networks are one hidden layer networks that use a wavelet as an activation function instead of the sigmoidal function. The nodes (wavelons) of wavelet networks are wavelet coefficients of the function expansion that have a significant value. The main reason for wavelets being used instead of any other transfer functions is that wavelets have strong compression abilities. Also, calculating the value at a single point or updating the function estimate from a new local measure involves only a small subset of coefficients [26]. Moreover, wavelet networks allow for effective procedures that efficiently initialize the parameters of a network. By using wavelet decomposition, a wavelet library can be constructed. In turn, each wavelon can be constructed using the best wavelet in the wavelet library. As a result, wavelet networks provide information for the relative participation of each wavelon to the function approximation and the predictable dynamics of the generating process. The main characteristics of these constructive procedures are (1) convergence to the global minimum of the cost function, and (2) the initial weight vector being in proximity to the global minimum, resulting in drastically reduced training times [28].

Finally, well-organized initialization methods will approximate the same vector of weights, and minimize the loss function each time.

In summary, WNN is a combination of wavelet analysis and neural network system, which has the benefit of time-frequency localization of wavelet transform and the powerful learning function of neural networks. Therefore, WNN has optimized coverage of functions based on wavelet frame theory and the time-frequency localization of wavelet transform through suitably choosing the dilation and translation parameters and adjusting wavelet coefficients.

3.5 Training a Wavelet Network with Back-propagation

After the initialization stage the network has to be trained further to find the weights that minimize the cost function. Since the wavelet is a function whose energy is well localized in time and frequency [27], we will use the wavelet network with back-propagation technique. Back-propagation wavelet neural network (BPWNN) is an artificial neural network that is integrated with wavelet techniques and has been used successfully in numerous fields.

As mentioned earlier, the wavelet coefficients of mother wavelet using dilating and translating, have great characteristics of time precision in high-frequency precision in high-frequency domains, and frequency precision in low-frequency domains. Therefore, the ability of a WNN in mapping complicated nonlinear functions is improved considerably, instead of using conventional nonlinear sigmoid transfer functions [29]. Nevertheless, we used the Morelet wavelet and the sigmoid transfer functions in our WNN design to achieve highest precision output signal.

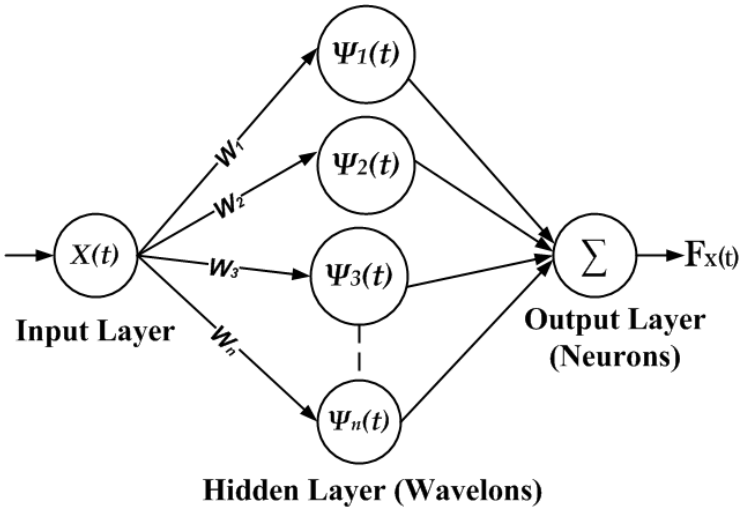

Figure 3.7: Schematic diagram of wavelet neural network [30]-[31]

The basic structure of BPWNN with a single hidden layer is shown in Figure 3.7 [29]. The neural network operates in three stages: An input layer, a hidden layer, and an output layer. The input layer receives the input information of WNN from the external sources to the network for processing. The hidden layer collects information from the input layer and processes the information, after that the output layer receives processed information from the network and sends the results out to the output result. Where *X(t)* is the input signal, w_n is the weight of (n^{th}) neuron between the input signal and the hidden layer, and $\Psi(t)$ is the mother wavelet function. The Function $F_x(t)$ is the output of sigmoid transfer function.

The mother wavelet *($\Psi(t)$)* is a series of daughter wavelets that can be developed through dilating and translating function, where $\Psi_{a,b}$ can be expressed as in equation

$$\boldsymbol{\Psi_{a,b}(x) = \frac{1}{\sqrt{a}}\ \Psi\left(\frac{x-b}{a}\right)} \qquad \textbf{(3.11)}$$

Where (*b)* is the translation factor, and (*a)* is the dilation factor.

In the aim for high resolution in both time and frequency domains, we opt in this work to implement the Morelet wavelet, discussed in the preceding section, as the transfer function of the BPWNN [32].

Morelet wavelet function is:

$$\boldsymbol{\Psi(t) = cos(1.75t)\ exp\ (-\frac{t^2}{2})} \qquad \textbf{(3.12)}$$

The sigmoid transfer function of the neural network is

$$\boldsymbol{\sigma\ (z) = \frac{1}{1+\exp(-Z)}} \qquad \textbf{(3.13)}$$

The main formula of the wavelet neural network function and the output layer transfer function G(x) of the sigmoid transfer function is:

$$\boldsymbol{G(x) = \sigma\ (\sum_{i=1}^{N} W_i\ \Psi\left(\frac{x-b_i}{a_i}\right))} \qquad \textbf{(3.14)}$$

Where w_i, a_i and b_i are weight, translation and dilation coefficients for each daughter wavelet, and (*i*) is the number of network nodes.

The Main idea of back-propagation is to find the percentage contribution of each weight to the error. The error *E* for a pattern (n) is simply the difference between the desired signal (d_n) and the network output (g_n). By squaring and multiplying it by $\frac{1}{2}$, we take the equivalence error E_n, which is used in network training:
The error equation is:

$$\boldsymbol{E_n = \frac{1}{2}\left(d_n - g_n(x)\right)^2} \qquad \textbf{(3.15)}$$

Where, $(n = 1, 2 \ldots\ldots N)$

The network will be trained until a vector of weight W_i(n) that minimizes the proposed cost function is found. During the training, the predicted output $(g_n(x))$ is compared with the desired output (d_n), and the mean square error E_n will be calculated at each pattern (*n*). If the mean square error is more than a prescribed limiting value, it is back propagated from output to input, and weights are further adapted till the error or number of iterations is within a proposed limit. The training algorithm will be discussed in the following section.

3.5.1 Sigmoid Transfer Function

As discussed above, any artificial neuron consists of a summing function with an internal threshold, and "weighted" inputs (see Figure 3.8). For a neuron receiving some inputs (n), each input signal x_i (i = 1 to n) is weighted by multiplying it with their weight w_i. The sum of the $w_i x_i$ products outcomes the net stimulation of the neuron. This stimulation value is subjected to a transfer function to produce the neuron's output. Therefore, one of the most important characteristics of any neural network is the type of transfer function used to calculate the output of node from its net activation [26].

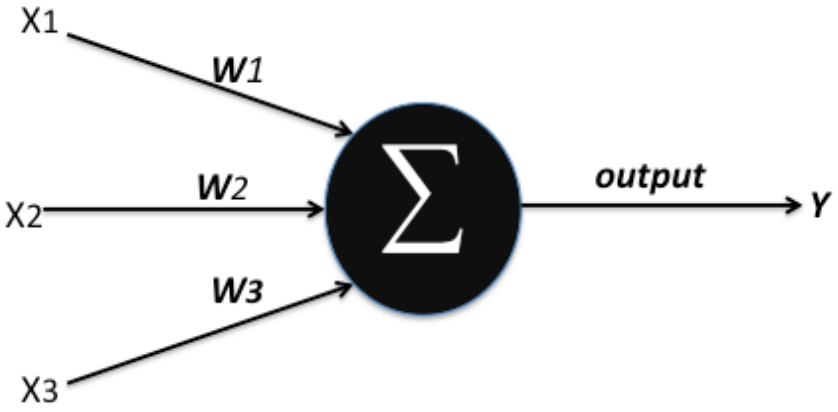

Figure 3.8: Artificial neuron network

The most popular transfer function that produces a continuous value in the range 0 to 1 is the sigmoid transfer function is a mathematical function having an "S" shape (sigmoid curve). Sigmoid function illistrate the special case of the logistic function shown in Figure 3.9 and is defined by the formula

$$\boldsymbol{Output_i = \frac{1}{1+e^{-\,gain\ activiation_i}}} \qquad \textbf{(3.16)}$$

If the gain activation is high, this will lead to hard activation. Thus, if the gain activation is low, that means it is close to identity function, and it affects the slope of the transfer function around zero.

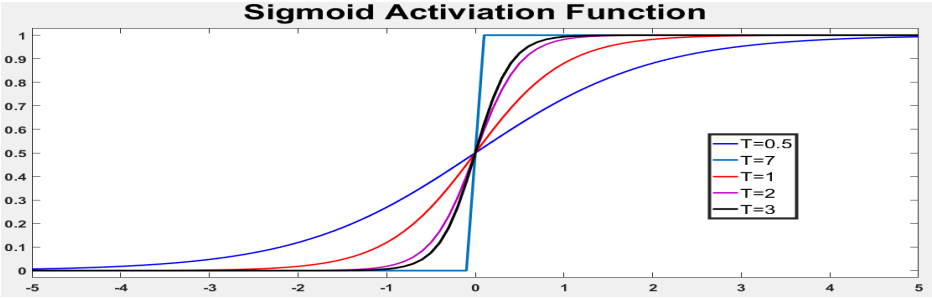

Figure 3.9: Family of sigmoid (Log-Sigmoid) transfer functions (MATLAB Function: a= logsig(n)

There are multiple transfer functions (see Table 3.2), which could be used for the neural network but the sigmoid transfer function is best suited to use in multiple-input networks trained with back propagation. The output signal will be in the range between 0 and 1.

Table 3.2: Types of transfer functions

Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1+e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n$		poslin
Competitive	$a = 1$ neuron with max n $a = 0$ all other neurons	C	compet

3.5.2 Calculus of Variation

The calculus of variations is a mathematical analysis that deals with minimizing or maximizing the paramters to optimizing the values to the minimal errors , which, are mappings from a set of functions to the real numbers. Calculus of variation function is often expressed as definite integrals involving functions and their derivatives.

As we have three variable parameters ($w_i, b_i, \& \, a_i$) for the wavelet neural network, we need to find the best-suited values for each of them that lead to the smallest *E*. Therefore, we will use in calculus of variations, the Euler–Lagrange equation (Euler's equation) to find the shortest distance between any two points. We will calculate the differential equation of ($w_i, b_i, \& \, a_i$) in regard to *E* to find the optimum values of ($w_i, b_i, \& \, a_i$) that lead to the smallest error.

The calculus of variation to find the partial derivatives of the functional *g(x)* concerning ($w_i, b_i, \& \, a_i$) is composed of the following formulas:

Morelet wavelet function

$$\boldsymbol{\varphi(t) = \cos(1.75t)\, exp\ (-\frac{t^2}{2})} \qquad \textbf{(3.17)}$$

The sigmoid transfer function of the

neural network

$$\boldsymbol{\sigma\,(z) = \ \frac{1}{1+\exp(-Z)}} \qquad \textbf{(3.18)}$$

The main formula of wavelet neural network function

$$\boldsymbol{g(x) = \sigma\ (\sum_{i=1}^{N} W_i\ \ \varphi\ \left(\frac{x-b_i}{a_i}\right))} \qquad \textbf{(3.19)}$$

$$\boldsymbol{g(x) = \frac{1}{1+exp\left(-\sum_{i=1}^{N} W_i\ \ cos\left(1.75\,\frac{x-b_i}{a_i}\right) exp\left(-\frac{\left(\frac{x-b_i}{a_i}\right)^2}{2}\right)\right)}} \qquad \textbf{(3.20)}$$

Error equation (E) is

$$\boldsymbol{E = \frac{1}{2}\ (d - g(x))^2 \quad (n = 1, 2, \ldots . N)} \qquad \textbf{(3.21)}$$

$$\boldsymbol{E = \frac{1}{2}\ (d - \frac{1}{1+exp(-\sum_{i=1}^{N} W_i\ \ cos\left(1.75\frac{x-b_i}{a_i}\right) exp\ (-\frac{(\frac{x-b_i}{a_i})^2}{2}))})^2}$$

where $(n = 1,2, \ldots . . N)$

$$\boldsymbol{E = \frac{1}{2}\left(d - \frac{1}{1+exp\left(-\sum_{i=1}^{N} W_i\ \ cos(1.75\,\tau)\ exp\left(-\frac{(\tau)^2}{2}\right)\right)}\right)^2} \qquad \textbf{(3.22)}$$

where $(n = 1,2, \ldots . . N)$ and, $\boldsymbol{\tau = \frac{x-b_i}{a_i}}$

$$\Delta(\boldsymbol{W_i}) = \boldsymbol{\frac{dE}{dW_i} = -\left(d - g(x)\right) g'(x) \cos(1.75\tau)\ exp(-\frac{\tau^2}{2})} \qquad \textbf{(3.23)}$$

$$\Delta(\boldsymbol{b_i}) = \boldsymbol{\frac{dE}{db_i} = -\left(d - g(x)\right) g'(x) \frac{w_i}{a_i} exp(-\frac{\tau^2}{2})(1.75 \sin(1.75\tau) + \tau\, cos(1.75\tau))} \quad \textbf{(3.24)}$$

$$\Delta(\boldsymbol{a_i}) = \boldsymbol{\frac{dE}{da_i} = -\left(d - g(x)\right) g'(x)\, \tau \frac{w_i}{a_i} exp\left(-\frac{\tau^2}{2}\right)(1.75 \sin(1.75\tau) + \tau\, cos(1.75\tau)}$$

$$\boldsymbol{\frac{dE}{da_i} = \tau \frac{dE}{db_i}} \qquad \textbf{(3.25)}$$

3.5.3 Gradient Descent Algorithm

Gradient Descent Algorithm is an optimization algorithm for finding the nearest minimum of a function which presumes that the gradient of the function can be computed. It starts at a point A, then moves to point B by moving minor steps along the negative of the gradient function to reach a global minimum.

The gradient algorithm for the wavelet neural network equation is defined as a vector of partial derivatives for three dimensions E ($w_i, b_i, \& \ a_i$)

$$\boldsymbol{\Delta E = [\frac{dE}{dW_i}, \frac{dE}{db_i}, \frac{dE}{da_i}]} \qquad \textbf{(3.26)}$$

Where ΔE is the Gradient.

That means (ΔE) and can be evaluated at any particular point W_i, as well as a_i and b_i. The calculated values of W_i, a_i and b_i could lead to fast changing in each direction to reach to the minimum value of ΔE. To reach to $\boldsymbol{\frac{dE}{dW_i} = \frac{dE}{db_i} = \frac{dE}{da_i} = 0}$ or the minimum value.

Therefore, the value of W_i, a_i and b_i will be changed to minimize the function $\frac{dE}{dW_i}$, $\frac{dE}{db_i}$, $\frac{dE}{da_i}$ as follows:

If $\frac{dE}{dW_i}$>0 then dW_i increases as W_i increases, so the gradient ΔE should decrease W_i

If $\frac{dE}{dW_i}$<0 then dW_i decreases as W_i increases, so the gradient ΔE should increase W_i

If $\frac{dE}{dW_i}$=0 then dW_i is at the minimum, so the gradient ΔE should not change W_i

The gradient descent rule

$$\boldsymbol{W_{new} = W_{old} - \eta\ \Delta(W_i)} \qquad \textbf{(3.27)}$$

Where, $-\Delta(w)$ is the gradient and η is the learning rate (small, positive).

The learning rate (η) determines how far the result will be.

General Gradient Descent Algorithm

The steps for optimizing any function using gradient descent algorithm starts by defining the objective function $E(W_i)$ that will be optimized, and then finding the vector of values (W_i) that will minimize E (W_i). First, an initial set of weights (W_i) will be picked up randomly. Then, $\Delta(W_i)$ will be evaluated for each value of (W_i), as shown in Figure 3.10, and subsequently all the weights will be updated with (W_{new}) (equation 3.27).

The weight will keep moving and updating their values at weight spaces until the $\Delta(W_i)$ is approximately equal to zero (converged to a flat minimum).

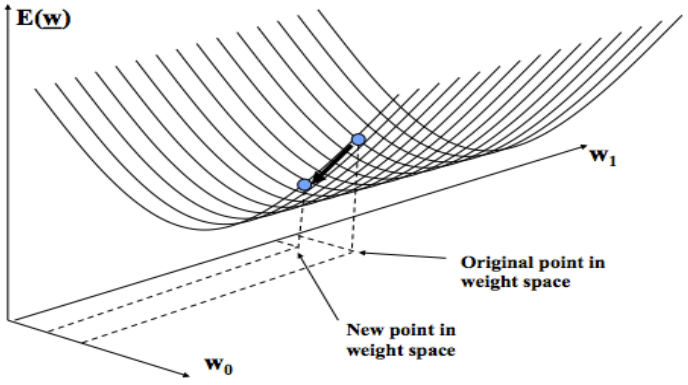

Figure 3.10: Calculating gradient descent algorithm

In order to implement the gradient descent algorithm, a learning rate value (η) and the number of learning iterations (i) need to be chosen. The learning rate (η) determines how fast the algorithm attempts to converge. The gradients for each parameter are multiplied by (η) before being used to modify that parameter, as formulated in equations (3.28) – (3.30).

The learning iterations (i) determine how many times the training data should be fed through the learning procedure. The higher this value is, the closer the convergence of the network to the minimum should be. The calculation time will increase [30]-[31].

In summary, the gradient descent algorithm applied will use the following formulas:

Weight update rule

$$\Delta \boldsymbol{w_i}(\boldsymbol{t+1}) = -\boldsymbol{\eta}\frac{\boldsymbol{dE}}{\boldsymbol{dW_i}} + \boldsymbol{\beta W_i}(\boldsymbol{t}) \qquad \textbf{(3.28)}$$

$$\Delta \boldsymbol{b_i}(\boldsymbol{t+1}) = -\boldsymbol{\eta}\frac{\boldsymbol{dE}}{\boldsymbol{db_i}} + \boldsymbol{\beta b_i}(\boldsymbol{t}) \qquad \textbf{(3.29)}$$

$$\Delta \boldsymbol{a_i}(\boldsymbol{t+1}) = -\boldsymbol{\eta}\frac{\boldsymbol{dE}}{\boldsymbol{da_i}} + \boldsymbol{\beta a_i}(\boldsymbol{t}) \qquad \textbf{(3.30)}$$

Where β is the momentum factor to the weights, η is the learning rate (small, positive) that determines the length of the weight update.

Gradient descent rule

$$\boldsymbol{w_i}(\boldsymbol{t+1}) = \boldsymbol{w_i}(\boldsymbol{t}) + \Delta \boldsymbol{w_i}(\boldsymbol{t+1}) \qquad \textbf{(3.31)}$$

$$\boldsymbol{b_i}(\boldsymbol{t+1}) = \boldsymbol{b_i}(\boldsymbol{t}) + \Delta \boldsymbol{b_i}(\boldsymbol{t+1}) \qquad \textbf{(3.32)}$$

$$\boldsymbol{a_i}(\boldsymbol{t+1}) = \boldsymbol{a_i}(\boldsymbol{t}) + \Delta \boldsymbol{a_i}(\boldsymbol{t+1}) \qquad \textbf{(3.33)}$$

Local Minimum

The gradient descent algorithm has a problem that emerges during the network training. As shown in Figure 3.8, we see there is more than one minimum point (local minimum, global minimum). The back-propagation algorithm can reach one of these

solutions, depending on the starting point. The way that gradient descent is implemented does not afford any guarantee that the output solution will be the optimal global minimum, which corresponds to the lower overall error level.

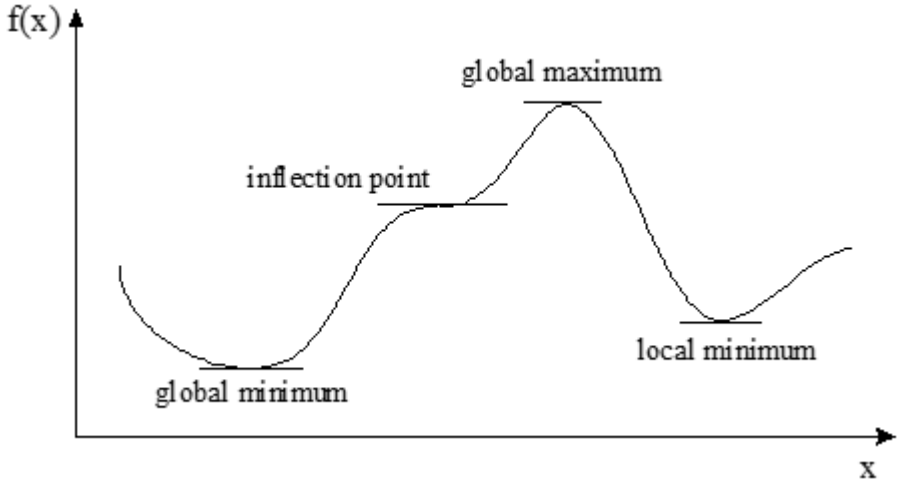

Figure 3.11: Local minima at gradient descent algorithm

By default, the gradient descent goes to the closest local minimum, and it could have multiple local minima at the weight. Therefore, a simple procedure followed to increase the probabilities of finding the global minimum is called weight jogging. This method depends on setting a random restart points at multiple places when an initial solution is found. Those resets are small changes added up to the weight vector, then the training process continues until the algorithm converges to a new solution [21]. However, as mentioned previously, there is no guarantee that this process will lead to a better solution as it is only improving the search for the global minima.

3.6 Summary Steps for CT-WNN

In this chapter, we studied WTs and their advantages in analyzing burst-like signals and low-frequency signals, which significantly fit continuous time systems.

Step 1: Insert the digitalized continuous time signal, V_{ref}, to the Morelet wavelet function (equation 3.17), then apply the CWT by changing the weight, translation, and

dilation for each wavelet coefficients (equation 3.19). After this, the signal will be reformed by the sigmoid transfer function (equation 3.18), to be represented by (equation 3.20).

$$g(x) = \frac{1}{1+exp\left(- \sum_{i=1}^{N} W_i \ \ cos\left(1.75 \frac{x-b_i}{a_i}\right) exp\left(-\frac{\left(\frac{x-b_i}{a_i}\right)^2}{2} \right) \right)} \quad \textbf{(3.20)}$$

Step 2: The WNN signal, $g(x)$, will be subtracted from the input signal (*d*) to calculate the *E*, (equation 3.21) between the original input signal and WNN pattern signal.

$$E = \frac{1}{2} (d - g(x))^2 \quad (n = 1, 2, \ldots . N) \quad \textbf{(3.21)}$$

The lowest error, E, that will be achieved from WNN training will be equal to the signal resolution. For this reason, the goal of WNN training in the next step is to achieve the lowest error. Hence, the signal resolution of WNN is a variable resolution and is dependant on the signal behavior and WNN training. Usually, the circuit designer sets the resolution range for the circuit design, and calibrates it due to the design requirements as a higher resolution output signal will require more training iterations, which consumes power and time delay for the output signal.

Step 3: Applying gradient descent algorithm to train WNN, by calculating the error factor and adjusting the weight, translation, and dilation to achieve the lowest error.

$$w_i(t+1) = w_i(t) - \eta \frac{dE}{dW_i} + \beta W_i(t) \quad \textbf{(3.28), (3.31)}$$

$$b_i(t+1) = b_i(t) - \eta \frac{dE}{db_i} + \beta b_i(t) \quad \textbf{(3.29), (3.32)}$$

$$a_i(t+1) = a_i(t) + -\eta \frac{dE}{da_i} + \beta a_i(t) \quad \textbf{(3.30), (3.33)}$$

Chapter 4

The Architecture of Clock-less ADC

The architecture of a Clock-less ADC system is at the heart of our proposed system in addition to many improvements and procedures that have been followed to apply WNN to improve the performance of the continuous-time output signal. In this chapter, we will describe the three stages of the system and their theory of operation; stage one is the delta modulation to convert the input analog signal to a continuous-time digital signal. The second stage combines the quantization error with the continuous output signal of delta modulation, and the third stage applies the WNN to the combined signal.

4.1 The Architecture of Clock-less ADC System Using WNN

The block diagram displayed in Figure 4.1 illustrates the full suggested system. A sinusoidal input signal ($V_{i/p}$) is applied on the left of the diagram to the amplitude shift stage, then many stages follow. The rest of this chapter will describe the theory of operation for each block and its impact on the design. For the description of each block, please refer back to the full system diagram in Figure 4.1.

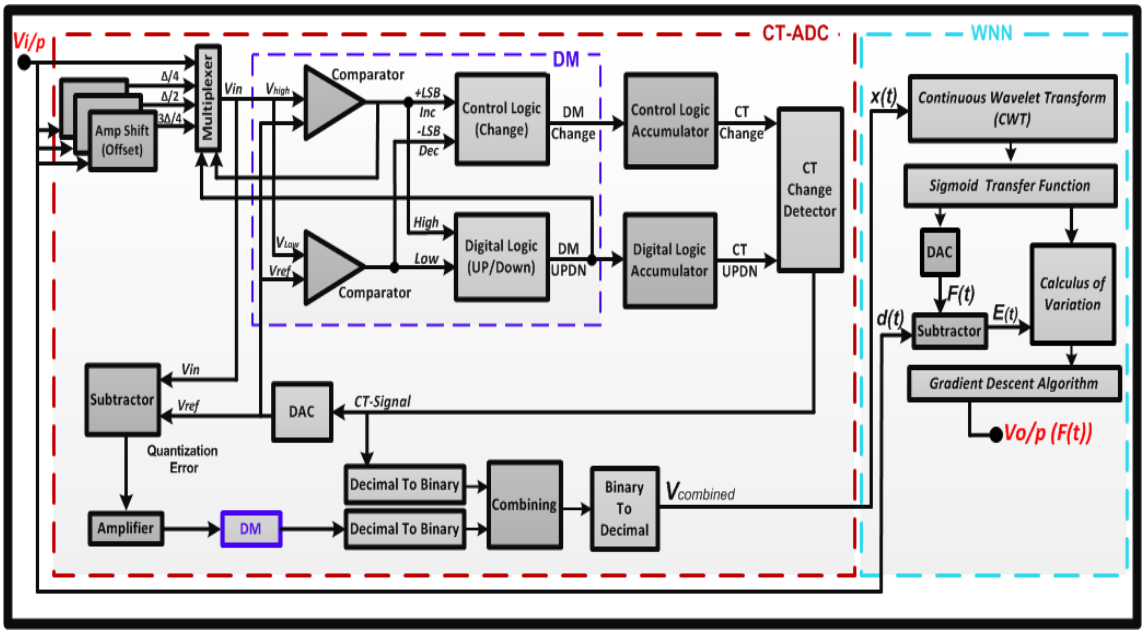

Figure 4.1: A block diagram of the CT-ADC using wavelet neural network

4.2 Amplitude Shifting (Offset)

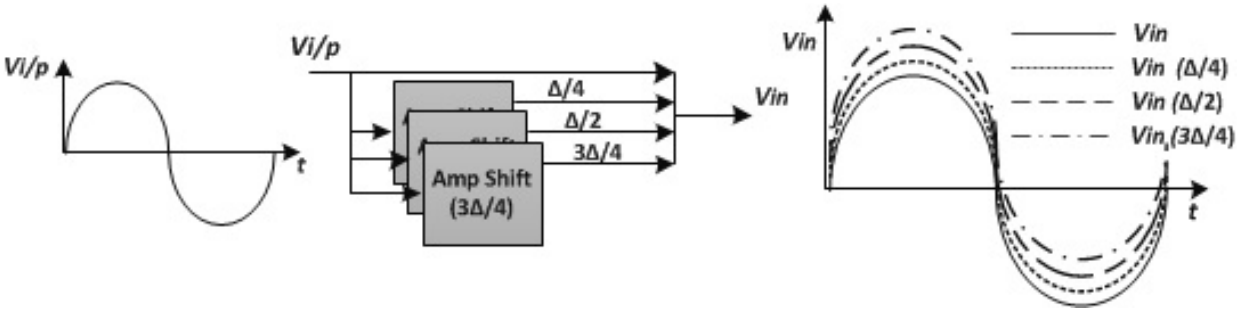

Figure 4.2: Amplitude shifting for analog sine wave input signal

A sinusoidal input signal will be applied to three amplitude shifters to shift the input signal into three different levels above the main signal level. The magnitude of shifting depends on the delta level value (Δ). The first amplitude shifter will shift the input signal by ($\Delta/4$), the second amplitude shifter will shift it by ($\Delta/2$). Moreover, the third amplitude shifter will shift it by ($3\Delta/4$). Thus, the three shifters will duplicate the same input signal with a negligible time delay between the three levels by an amount equal to a quarter delta (Δ). The goal of this block is to get multiple copies of the same input signal and apply these four signals, (the main input sine wave and the three shifted signals), to the continuous time ADC. Then compose four digitalized signals from the single input signal; the four signals will be analyzed individually according to the amplitude shifter identified by their delta shift. This point will be discussed further when it comes to the accumulator block.

4.3 Multiplexer

The multiplexer block is a regular four input one output multiplexer with two selection signal (S_0, S_1) as shown in Figure 4.3. The input signal ($V_{i/p}$) and their three replicated signals will be applied to the multiplexer and the output signal (V_{in}) will be produced according to the two selection. By default, the input signal ($V_{i/p}$) will pass through the MUX, then once this signal passes the comparator block, the comparator will send partial feedback to the multiplexer to move forward to the next signal to pass through the multiplexer.

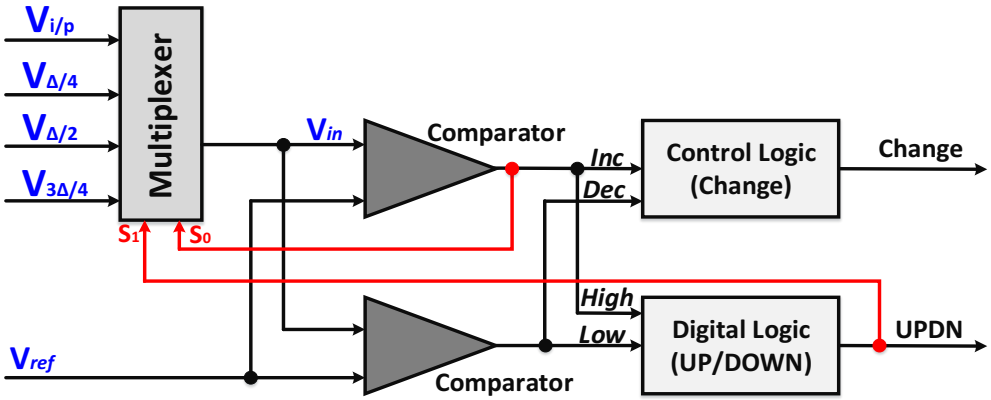

Figure 4.3: Multiplexer for CT-ADC

When the signals reach through the multiplexer and the comparator to the digital logic, the digital logic block will also partially control the next signal to go through the multiplexer. If the digital logic sends an UP signal, the multiplexer will pass the signal which is higher than the last signal by an increment of $\Delta/4$. Similarly, if the digital logic sends a DOWN signal, the multiplexer will pass the decremented version of the last signal going through the multiplexer. This will be elaborated even more during the next few subsections.

4.4 Comparator

The comparator is the main component in clock-less ADC circuit architecture, and it is the most challenging building block because it should meet several requirements. Its gain needs to be high enough to resolve small voltage differences, and it should be low power because the minimum power consumption of the system is set by the static power consumption of both comparators. The comparator will be used to compare the input signal (V_{in}) and reference signal (V_{ref}) (the voltage signal of the latest input signal), and then convert the analog signal to non-uniformed digital data signals called emblems, by using quantization levels as will be described shortly.

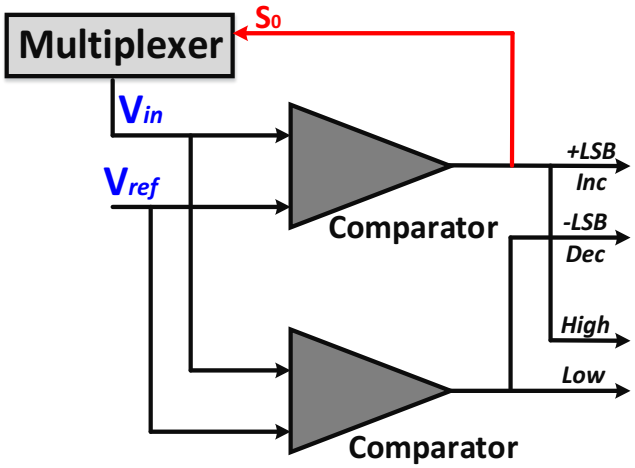

Figure 4.4: Continuous time ADC comparator

The comparators operate as follows: When the input signal is within the range of two quantization levels as shown in Figure 4.5, then both comparators initially have outputs of logic 0, and nothing happens. When the input signal moves outside this range and crosses on of the quantization levels, one of the comparators will detect the change of this signal and will create a logic 1 on either INC or DEC, depending on the direction of movement [7].

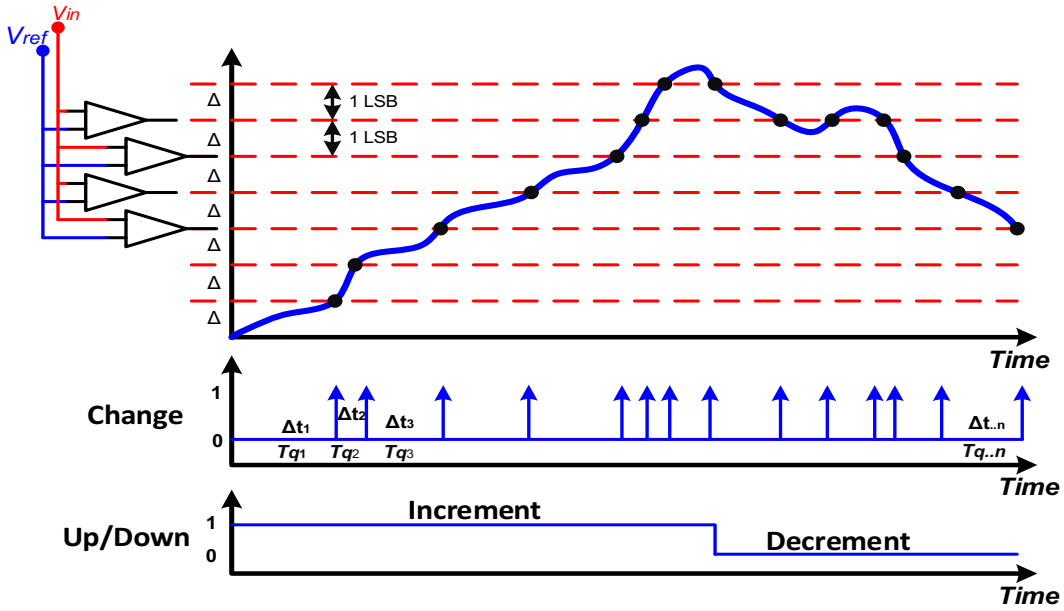

Figure 4.5: Example of waveform showing how the CT-ADC operates

When either INC or DEC becomes logic 1, the control logic and digital Logic will together generate an indication of the Emblem. This Emblem consists of two binary signals: CHANGE and UP/DOWN. The Emblem is fed back to the DAC with the updated changes to generate a new reference signal (V_{ref}). After such an update, the signal is once again within the range. This behaviour is demonstrated in Figure 4.5.

4.4.1 Delta Modulator Architecture

In clock-less systems, the input signal is digitized by generating emblems through delta modulation (amplitude levels), so the time is continuous (infinite), and the amplitude is well-known by considering the threshold-level resolution. However, in conventional systems, the input signal is digitized by cut off frequency as shown in Figure 4.6.

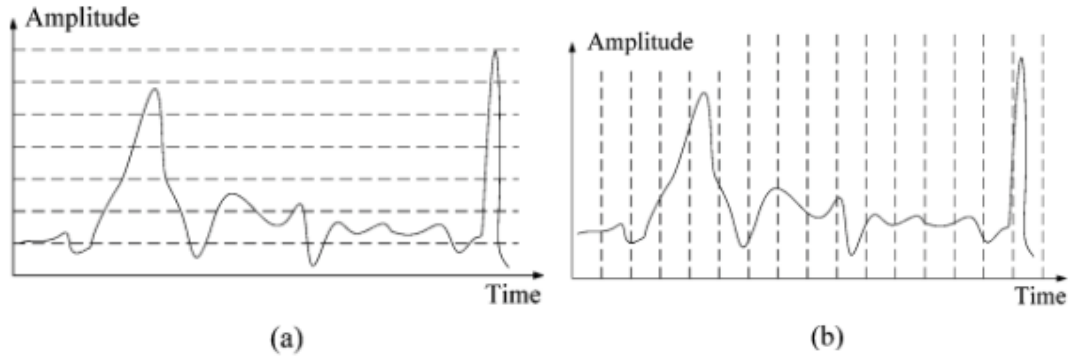

Figure 4.6: (a) Non-uniformed sampling (delta modulation) (b) Uniform sampling

Any initial input signal will have initial amplitude around the zero point in range period [-0.5Δ, 0.5Δ]. The total number of quantization levels (threshold levels) for an N-bit quantizer is 2^N, but because of the alignment of a mid-tread quantizer to space the levels and keep odd symmetry in the relationship of the input and output it will be only 2^N-1 levels.

The following equations show the relationship between the threshold levels and the number of bits [7].

Number of quantization level (delta modulation levels) = 2^N-1 **(4.1)**

Delta step space size (weighting of LSB): $[\Delta]$ (LSB) = $\frac{V_{max}}{2^N}$ **(4.2)**

V_{max} is the difference between the maximum reference voltage (+0.5) - the minimum reference voltage (-0.5) (V_{max}= V_{max}high - V_{max}low) [33]

N (bit) = $\frac{Log\frac{1}{\Delta}}{\log 2}$ **(4.3)**

The maximum amplitude of the input signal that retains odd symmetry is represented as

$$X_{\max Peak} = 1 - \frac{1}{2^N} \quad \textbf{(4.4)}$$

The signal to noise ratio (SNR) for the conventional system can be denoted by this equation.

$SNR_{(MAX)}$ = (6.02 × N + 1.76) $_{dB}$ **(4.5)**

where N is number of bits. The SNR calculation will discussed in (section 5.1.3).

The below table exposes the relationship between the number of bits, the number of level crossing with the weighting of LSB, and the SNR. We displayed the above chart because we use the four-bits circuit with 16 level crossings to enhance the output signal of WNN system to be almost 38.3 bits according to the input signal and the time resolution. The SNR will be illustrated at section (5.1.3)

Table 4.1: The relation between the number of bits, number of levels, and SNR

Resolution and Signal to Noise Ratio (SNR) for signals coded as n bits			
Bits, n	**Levels, 2^n**	**Weighting of LSB, 2-n**	**SNR, *dB***
2	4	0.25	14
3	8	0.125	20
4	16	0.0625	26
5	32	0.03125	32
6	64	0.015625	38
7	128	0.0078125	44
8	256	0.00390625	50
9	512	0.001953125	56
10	1024	0.000976563	62
11	2048	0.000488281	68
12	4096	0.000244141	74
13	8,192	0.00012207	80
14	16,384	6.10352E-05	86
15	32,768	3.05176E-05	92
16	65,536	1.52588E-05	98
17	131,072	7.62939E-06	104
18	262,144	3.8147E-06	110
19	524,288	1.90735E-06	116
20	1,048,576	9.53674E-07	122
21	2,097,152	4.76837E-07	128
22	4,194,304	2.38419E-07	134
23	8,388,608	1.1921e-07	140
24	16,777,216	5.9605e-08	146
25	33,554,432	2.9802e-08	152
26	67,108,864	1.4901e-08	158
27	134,217,728	7.4506e-09	164
28	268,435,456	3.7253e-09	170
29	536,870,912	1.8626e-09	176
30	1.0737e+09	9.3132e-10	182
31	2.1475e+09	4.6566e-10	188
32	4.2950e+09	2.3283e-10	194
33	8.5899e+09	1.1642e-10	200
34	1.7180e+10	5.8208e-11	206
35	3.4360e+10	2.9104e-11	212
36	6.8719e+10	1.4552e-11	219
37	1.3744e+11	7.2760e-12	225
38	2.7488e+11	3.6380e-12	231

4.5 Control Logic

Control logic block is the main block that detects the changes of the input signal when it crosses the delta modulation quantization levels. When the $V_{i/p}$ stays within the range of one delta modulation level (V_{Low} & V_{High}) the two comparators will have Logic 0, and the circuit will be inactive until either high comparator or low comparator detects movement on $V_{i/p}$ outside this range [7], [33]. The high comparator detects a signal level change to the next delta modulation level and will send an INC signal, logic 1 to the control logic block in this case. However, if the low comparator detects the change that means the input signal has moved to the lower delta modulation level, and the low comparator will send a DEC signal. logic 1 to the control logic block.

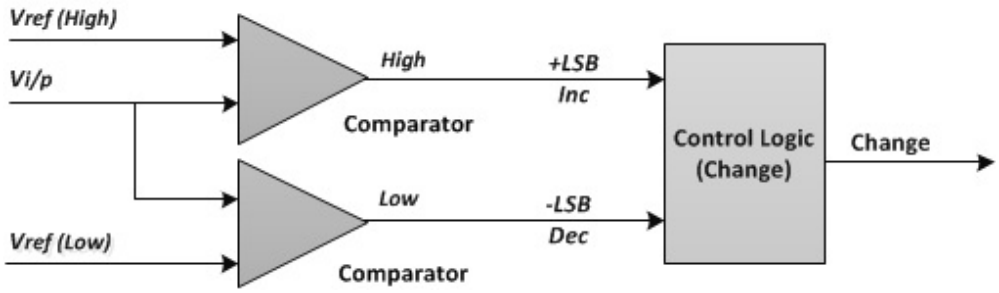

Figure 4.7: Control logic block diagram

Based on these indicator signals (INC and DEC), the change block will be able to record the changes of the input signal over the delta modulation levels. The Change signal will remain by default at logic 0, and the data emblem will have a value of $(T_{\mathrm{q}}, 0)$, but once it detects the INC or DEC signal it will change to logic 1, and thus the data emblem will have a value $(T_{\mathrm{q}}, 1)$, as shown in Figure 4.5. The control logic will ensure that the outputs of the comparators (INC and DEC) are not evaluated, while the system is still settling, and will detect and filter any glitches on these signals to deliver a sharp signal to DAC. In fact, when the DAC updates its outputs V_{HIGH} and V_{LOW}, there can be some glitches on these signals and the outputs of the comparators can oscillate between zero and one. Therefore, an architecture has been proposed in [34], and has been

implemented in the work reported in [35]-[37]. In theory, the digitized data emblem is a combination of the CHANGE & UPDN signals.

4.6 Digital Logic

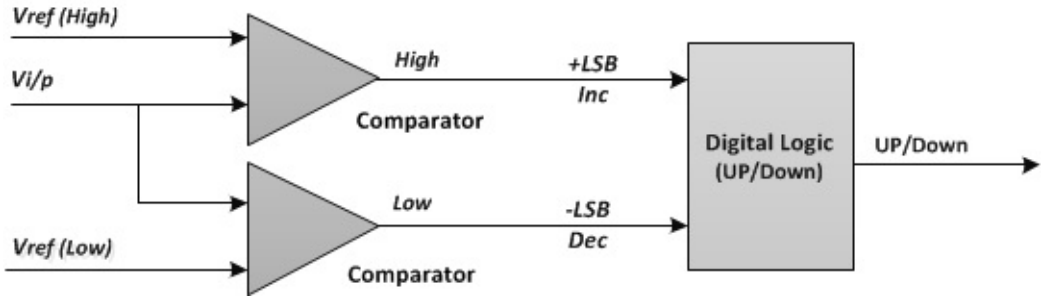

Figure 4.8: Digital logic block diagram

The digital logic block is working as counter and detector of the input signal behavior $V_{i/p}$ (up/down). The output signal of the digital logic block will be either 0 or 1 depending on the output the $V_{i/p}$. Once the $V_{i/p}$ moves out of the range (V_{Low} & V_{High}) and the input signal $V_{i/p}$ compared by the two separate comparators, the circuit will start being active and the digital logic will detect an input signal from either the high comparator or the low comparator. Both comparators have logic 0 output when the circuit is inactive. If the high comparator detects a signal change, it means that the input signal has moved to the next delta modulation level and will send an INC signal, logic 1, to the digital logic block at the time interval (T_{q}), and thus, if the low comparator detects the signal change that means that the input signal has moved to the lower delta modulation level, and will send a DEC signal, logic 1 to the digital logic block at the time interval (T_{q}).

The digital logic block will compare the received signal at the quantizing time (T_{q}) with the previous data stored at time interval ($T_{\mathrm{q}-1}$). If the data stored at the time interval ($T_{\mathrm{q}-1}$) is INC whereas the data received at time interval (T_{q}) is INC, so, the output signal of the digital logic at time interval (T_{q}) will be logic 1. If the data stored at the

time interval (T_{q-1}) is DEC and the data received at time interval (T_q) is DEC, so, the output signal of the digital logic at time interval (T_q), will be logic 0. If the data stored at time interval (T_{q-1}) is DEC and the data received at time interval (T_q) is INC, so, the output signal of the digital logic at time interval (T_q), will be logic 1. If the data stored at the time interval (T_{q-1}) is INC and the received data is DEC at time interval (T_q), the output signal of the digital logic at time interval (T_q), will be logic 0. The output signal of the digital logic block will be injected to the DAC block at the same time where the CHANGE signal is delivered. The DAC block will combine both signals to implement the digitized signal data emblem with quantized amplitude and exact time. Further details will be outlined when discussing the DAC block section at the end of this chapter.

4.7 CT-Timer

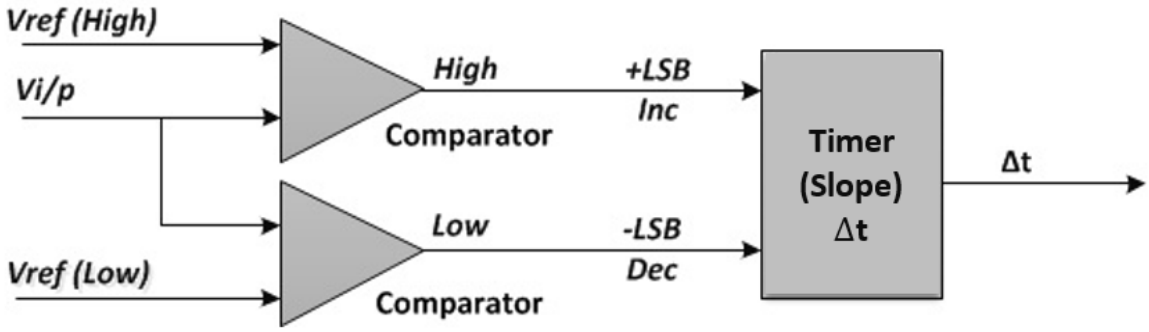

Figure 4.9: internal timer (slope of input signal)

In order to transport and store the non-uniform sampled data that was generated from the level crossing ADC, we need to add an internal sample or a complicated counter to record the time intervals among successive digitized signals (emblems). The above block diagram illustrates the block responsible for creating the internal timer, as the comparators will detect the input signal changes by comparing $V_{i/p}$ with V_{ref}, and then feed the timer block with INC or DEC signal according to the input signal behaviour. As a result, the timer block will calculate the time intervals with respect to the fixed quantization level.

4.7.1 Time Constant of CT System (ΔT)

As this system has no external clock to control the time sequence of the quantized signal at the DSP system, the internal timer (T_{q}) has to be determined for each non-uniformed sample to synchronize the data at the DSP system on the time sequence. The internal timer (T_{q}) will be calculated according to the input signal behavior for each non-uniformed sample by determining the slope of the signal. The slope equation is

$$\textbf{Slope} = \frac{\Delta V}{\Delta T} = \frac{V_2 - V_1}{X_2 - X_1} = \frac{\mathrm{V}_{\Delta}}{\textbf{Max rate of changes}} \qquad \textbf{(4.6)}$$

Where, ΔV is the difference between any two sequential delta levels, and is always a fixed value equal to $\Delta/4$. ΔT is the time interval that input signal needed to pass from one delta level to the next level for each non-uniformed digitized signal; it is a random variable uniformly distributed across $[T_{\mathrm{q}}, T_{\mathrm{q+1}}]$ depending on the slope of the input signal. The equation of the time difference is

$$\Delta \mathbf{T} = \frac{\textbf{Quantization Step}}{\textbf{Input Slope}} \qquad \textbf{(4.7)}$$

The quantization noise power becomes [38]

$$\mathbf{P(\Delta V) = P\ (Slope).\ P(\Delta T)} \qquad \textbf{(4.8)}$$

$$\mathbf{P(\Delta T)} = \frac{\mathbf{T_{sample}}^2}{3} = \frac{\mathbf{P(\Delta V)}}{\mathbf{P(Slope)}} \qquad \textbf{(4.9)}$$

$$\because\ \mathbf{SNR_{db}} = \mathbf{10\log}\left(\frac{\mathbf{P(V_{in})}}{\mathbf{P(\Delta V)}}\right) = \mathbf{10\log}\left(\frac{\mathbf{3\ P(V_{in})}}{\mathbf{P(Slope)\ T_{sample}}^2}\right) \qquad \textbf{(4.10)}$$

$$\therefore\ \mathbf{SNR_{db}} = \mathbf{10\log}\left(\frac{\mathbf{3\ P(V_{in})}}{\mathbf{P(Slope)}} + \mathbf{20\log}\left(\frac{1}{\mathbf{T_q}}\right)\right) \qquad \textbf{(4.11)}$$

The SNR, equation (4.10), consists of two terms, the first term can be determined by the properties of the input signal (V_{in}), and the second term will be calculated by the time quantizer (T_q). Therefore, in a continuous-time system the SNR depends on the time quantizer and not on the number of quantization levels as in conventional system. The main differences between conventional systems and continuous-time systems are summarized in Table (4.2) [39].

Table 4.2: Comparison between continuous-time system and conventional system

	Conventional system (Sampling)	Continuous-time system (Emblem)
Conversion trigger	Clock	No clock Aid (Level crossing)
Amplitude	Quantized	Exact value
Time	Exact value	Quantized
SNR dependency	Number of bits	Timer period (Time quantizer)
Conversion output	Amplitude	(Amplitude, Time quantizer)

Theoretically, the SNR of clock-less systems can be improved by reducing the time quantizer (T_q), which depends mainly on the quantization levels. Therefore, adding the virtual three-quantization levels as discussed at the amplitude shifting block as previously discussed will reduce the time quantizer (T_q) and will thereby increase the accuracy of the output signal.

4.8 Accumulator

As presented above, the input signal has been duplicated three times and shifted by ($\Delta/4$), ($\Delta/2$), and ($3\Delta/4$). Those four signals have been injected into the CT-ADC system at the same time without any time delay. However, as they are shifted to different amplitudes, each input signal will cross the delta quantization levels at a different time according to the input signal behaviors, which will generate four change signals, four UP/Down signals, and four timer signals (Δt).

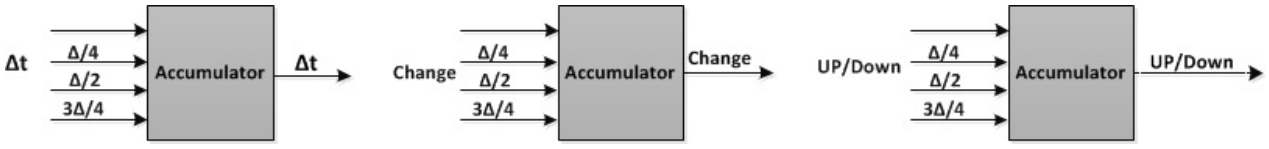

Figure 4.10: Schematic of accumulator blocks

As shown in the Figure 4.10, the block diagram of the CT-ADC has three accumulators to gather the four signals of each accumulator block to one output signal that represents the summation of the signals according to their specific amplitude (ΔV)

and interval time (T_q). As the main goal of using the shifting amplitude of input signal and accumulators is predicting the current change rapidly and achieving higher resolution. Figure 4.11 shows the difference between the Change signal generated by using the input signal solely and the Change signal by using the three amplitude shifted signals along with the input signal. We can observe that in the former case only two Change signals were generated when applying the input signal only. Whereas, in the later case, eight Change signals resulted from applying amplitude shifted input signals.

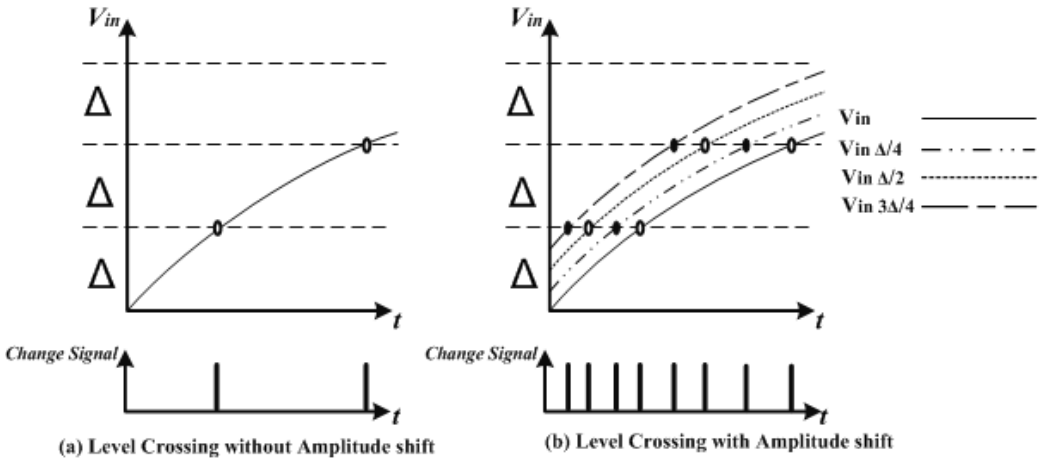

Figure 4.11: Schematic of accumulator blocks

4.9 Digital to Analog Converter (DAC)

As discussed in the last subsection, the four input signals (V_{in}) of each real input signal ($V_{i/p}$) will be injected to the accumulator, and the accumulator will gather the signals according to their internal timer to represent all these input signals as only one out signal. Then, the digital to analog converter will reconstruct the digitized output signal from the accumulators and will inject it into the wavelet neural network system to improve the signal resolution.

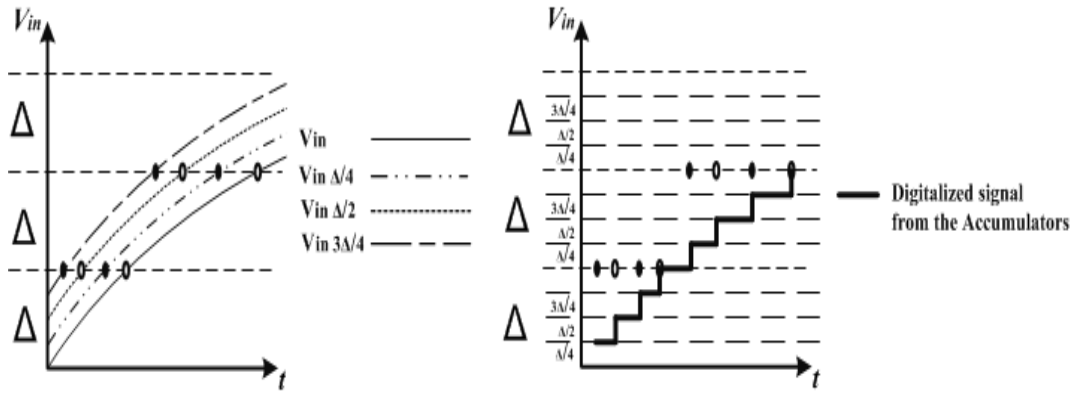

Figure 4.12: (a) The four amplitude signals (b) Digitalized output signal

4.10 Quantization Error

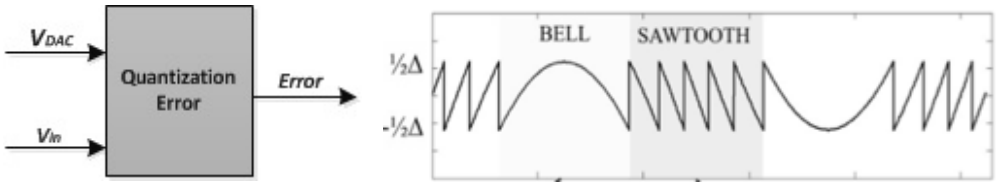

Figure 4.13: The quantization error for continuous-time analog to digital converter

The quantization error signal is the error introduced as a result of the quantization process. Therefore, the amount of this error is a function of the resolution of the quantizer. The quantization error is uniformly distributed between −1/2 LSB and +1/2 LSB, and it has been defined as the difference between the magnitude of the original input signal and the magnitude of the quantized (reconstructed) output signal.

Quantization Error **e (t) = x (t) - xq (t) =** $\frac{I}{p_{Signal}} - DAC_{Signal}$ **(4.12)**

4.10.1 Bell and Sawtooth Shape

The quantization error has a combined shape between bell-shape and sawtooth shape. The sawtooth-like error occurs during fast portions of the input when quantization levels are traversed quickly. The bell-shaped error occurs due to the shortage of quantization level to cover the highest or lowest point of the signal peak. Also, it occurs during slowly varying portions of the input signal.

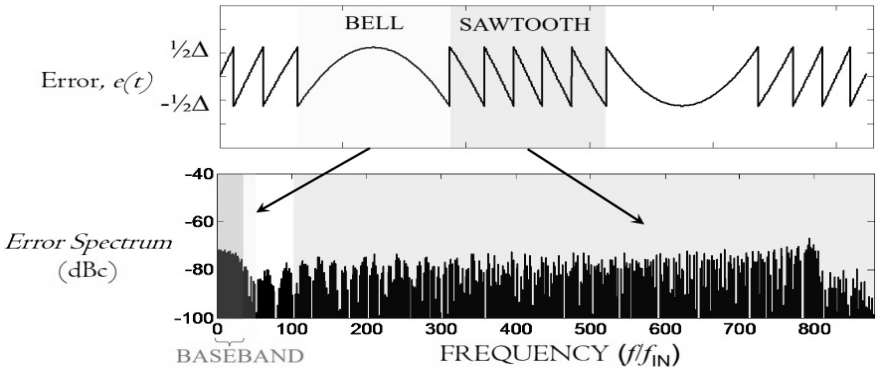

Figure 4.14: Bell- sawtooth shape [40]

4.11 Adaptive Resolution

At this stage, the digitized signal V_{DAC} will be adapted digitally to get a better signal resolution by using the quantization error of the input signal to reconstruct and modify the V_{DAC} for a better signal quality. The goal of adaptive resolution is to represent the V_{DAC} on twelve bits instead of eight bits by adding up to four binary bits using the quantization error *e(t)*. The following components will describe the method of adapting the digitized signal.

4.12 Amplifier

The amplifier block is put after the subtractor block to magnify the quantization error signal *e(t)* to be on the same scale as the input signal $V_{i/p}$. Amplifying the error will help in studying the error of the signal and removing the glitches and improving the resolution of the output signal. The quantization error *e(t)* will be magnified to be equivalent to the dynamic range of maximum input signal, according to the following equation.

Amplified Quantization error e (t) = [quantization error e(t) X # of Comparators] – 1 **(4.13)**

The following figure shows the amplified quantization error signal in term to the digitalized signal V_{DAC}.

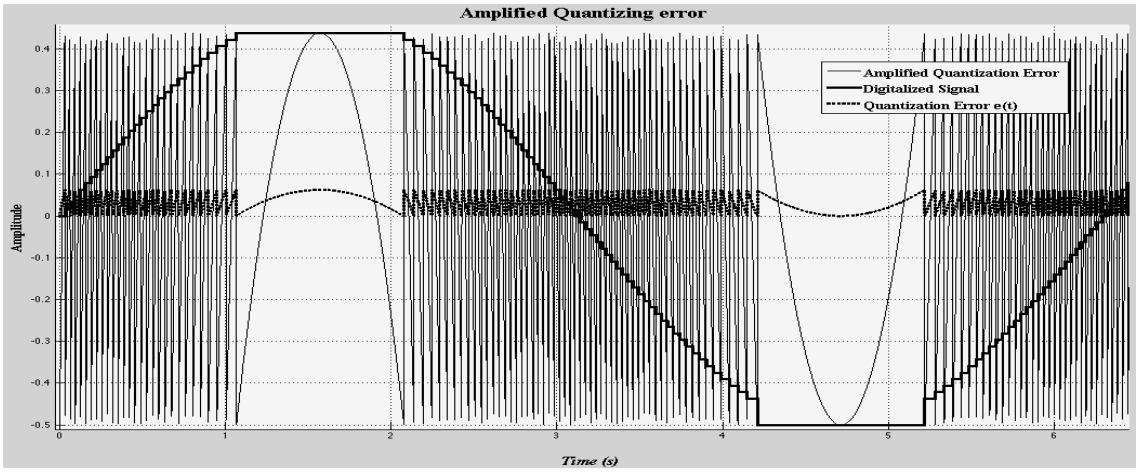

Figure 4.15: Amplified quantization error signal

The new quantization error signal after amplification has the same amplitude and time duration of the reconstructed signal V_{DAC}, and now it can be studied, compared, and combined with V_{DAC}. Therefore, the amplified quantization error signal and the digitized signal V_{DAC} will be converted to binary code to combine them according to their binary code and will be discussed in the following section.

4.13 Decimal to Binary / Binary to Decimal

For any digital hardware, values are stored in binary numbers with a fixed-length sequence of bits (1's and 0's). Binary numbers can be stored in many data formats including fixed-point and floating-point data types depending on the technique and the application for which it will be used. In this section, we will illustrate the methodology of converting the amplified quantization noise error and digitized signal V_{DAC} to binary code. Hence, the scale range of the input signal is always less than one volt that processes the digitalized signal V_{DAC} and amplified quantization noise into a fraction values with negative or positive sign. So, we will use a non-uniformed signed fixed-point to convert the fraction decimal numbers to binary code [41].

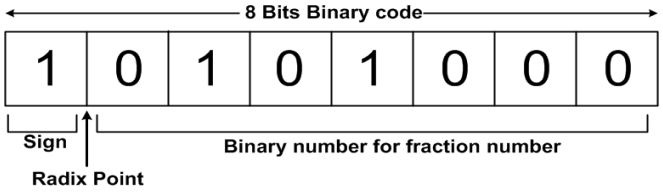

Figure 4.16: Binary word length for non-uniformed signed fixed-point

In this work, the decimal value is converted to an eight bit binary code. We use the first bit to characterize the sign (1 for negative, 0 for positive), and the remaining seven bits are used for the fraction, as the entire input signals are always fractions (between 1 and -1). Therefore, this pattern will be the best usage for the number of bits in our continuous time system. The following example will show the calculation of this non-uniformed technique of signed fixed-point to convert the fraction value to binary code.

For example, the decimal value (0.3957) will be converted to binary as follows:

Bit 1= Positive Number → **bit1 = 0 (MSB)**

Bit 2 = 0.3957 /0.5 = 0.7914 < 1 → **bit2 = 0**

Bit 3= 0.7914/0.5 = 1.5828 > 1 → **bit3 = 1**

Bit 4= [(1.5828 -1) = 0.5828] /0.5 = 1.1656 > 1 → **bit4 = 1**

Bit 5= [(1.1656 -1) = 0.1656] /0.5 = 0.3312 < 1 → **bit5 = 0**

Bit 6= 0.3312 /0.5 = 0.6624 < 1 → **bit6 = 0**

Bit 7= 0.6624 /0.5 = 1.3248 < 1 → **bit7 = 1**

Bit 8= [(1.3248 -1) = 0. 3248] /0.5 = 0.6496 < 1 → **bit8 = 0 (LSB)**

Thus, the decimal value (0.3957) will be represented in binary code as **00110010**

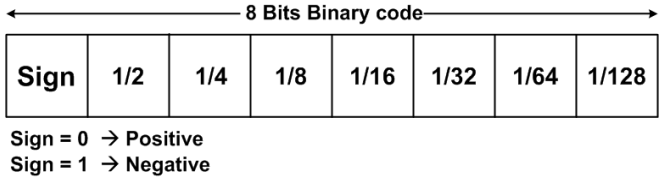

Figure 4.17: The fraction value of each binary bit

To reconstruct the decimal fraction number again, we will follow the reverting method to convert the binary code to decimal fraction number by multiplying the binary code (1's and 0's) times the fraction value of each bit. As we use 8 bit binary code, assigning the first bit to sign indication (positive or negative), only the remaining 7 bits will have the following fraction values $2^{-\frac{1}{1}}, 2^{-\frac{1}{2}}, 2^{-\frac{1}{3}}, 2^{-\frac{1}{4}}, 2^{-\frac{1}{5}}, 2^{-\frac{1}{6}}$, and $2^{-\frac{1}{7}}$.

The reconstructed decimal fraction value of the previously obtained 00110010 will be:

$00110010 = 0\,x\,2^{-\frac{1}{1}} + 1\,x\,2^{-\frac{1}{2}} + 1\,x\,2^{-\frac{1}{3}} + 0\,x\,2^{-\frac{1}{4}} + 0\,x\,2^{-\frac{1}{5}} + 1\,x\,2^{-\frac{1}{6}} + 0\,x\,2^{-\frac{1}{7}} =$ 0.3906

As shown in the prior example, the original number value was 0.3957, but the restored value of 8 bit signed fixed-point is 0.3906. In conclusion, the number of bits always affects the reassembled value, and the extra number of bits representing the decimal number; the more precise resolution will be the restored value.

4.14 Shifting, Combining, Module

Subsequently, the digitized signal and the quantization error have been converted to binary; the quantization error will be shifted 6 bits, and combined with the digitized signal to represent the combined digitized signal in 12 bits to improve the signal quality of the restored signal.

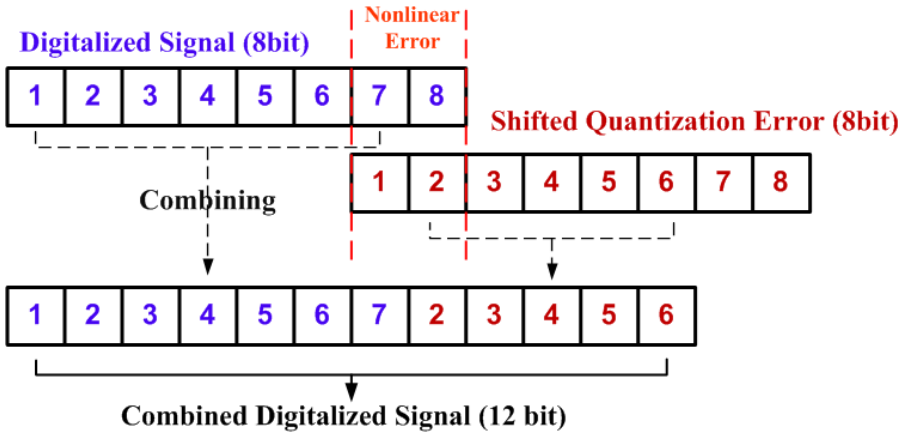

Figure 4.18: Combining digitalized signal and quantization error

Chapter 5

Simulation Results

In this chapter, the performance metrics are described then the simulation of the proposed system in MATLAB is detailed.

5.1 ADC Performance Metrics

Performance metrics are used to compare and characterize the CT-ADCs' performance. In order to achieve a reasonable and consistent comparison, much effort is being devoted to the standardization of methods to measure and characterize the CT-ADC's performance in this work. The performance metrics are often divided into static and dynamic. While static metrics are analyzed in the time domain, dynamic metrics, on the other hand, are analyzed in the frequency domain. Although there are numerous performance metrics, the amount and type of metrics used for a particular CT-ADC often depend on the application and context the ADC is employed. Therefore, metrics that are relevant to the specific application and target of continuous-time ADC are used here to compare the proposed design against previously published works. The metrics used are Signal to Noise Ratio (SNR), Signal to Noise and Distortion Noise (SINAD), Spurious-Free Dynamic Range (SFDR), Signal to Quantization Noise Ratio (SQNR), and Effective Number of Bits (ENOB). All basic performance metrics along with the specific performance metrics that have been used throughout this work are presented next [42]-[43].

5.1.1 Dynamic Range (DR)

The input dynamic range, sometimes just called dynamic range, is the range of the input signal that can be consistently measured simultaneously. For an input signal digitized into an n-bit output (resolution of n), the largest output code is 2^n-1 and the smallest output code would be greater than 0 [10], [42], [44]. Thus, the dynamic range is usually expressed (in *dB*) by

$$Dynamic\ Range\ (DR)_{db} = 20\ log\ (2^n - 1) \qquad \textbf{(5.1)}$$

DR particularly expresses the ability to measure small signals in the presence of large signals accurately. Therefore, it is an important parameter of any measurement system. It also can be defined as the ratio of the largest (maximum) input signal to the smallest possible input signal (highest harmonic or peak noise floor) that can be resolved. Moreover, dynamic range classifies the range of input signal amplitudes that can be consistently converted at a specified accuracy, expressed as the maximum ratio of the two signal levels. Considering the presence of noise floor, the peak amplitude of the noise floor restricts the minimum amplitude of a signal present at the ADC input, which allows detecting the presence of this signal in the ADC output spectrum. The following table presents the expected dynamic range of various values of resolution (bits).

Table 5.1: Table of the expected dynamic range for WNN system

Resolution (bits)	Dynamic Range (*dB*)
3	16.9020 *dB*
4	**23.5218 *dB***
5	29.8272 *dB*
6	35.9868 *dB*
7	42.0761 *dB*
8	48.1308 *dB*
9	54.1684 *dB*
10	60.1975 *dB*
11	66.2224 *dB*
12	72.2451 *dB*

5.1.2 Total Harmonic Distortion (THD)

Total harmonic distortion presents an indication of a circuit's linearity in terms of its effect on the harmonic content of a signal. In the ideal case, the pure sine wave has one frequency component, and a complex signal such as speech or music has multiple frequency components. The nonlinearities such as the converter's transfer function will produce harmonics that were not present in the original signal. In summary, THD returns the real-valued sinusoidal signal in dB.

The THD is defined as the ratio of the total root mean square (RMS) of the first given number of harmonic components (the RMS sum of the amplitudes of the harmonics) to the amplitude of the fundamental

$$\text{THD} = \frac{RMS\ \ Sum\ of\ harmonics}{RMS\ of\ fundamental} = \sqrt{\frac{V_{f2}{}^2 + V_{f3}{}^2 + \cdots V_{fn}{}^2}{V_{f1}{}^2}}$$

$$\text{THD}_{dB} = 10 \log \frac{\text{V}_{\text{f2}}{}^2 + \text{V}_{\text{f3}}{}^2 + \cdots \text{V}_{\text{fn}}{}^2}{\text{V}_{\text{f1}}{}^2} \qquad (5.2)$$

Where V_{f1} is the fundamental amplitude, V_{f2} is the second harmonic amplitude.

As a practical matter, there is no completely linear input to output transfer function. This nonlinearity leads to output distortion. As the input signal increases in amplitude, the output grows into more and more distorted. As a result, the distortion increases as the input amplitude increases. Therefore, THD performance reduces with increasing the input frequency because the effects of jitter get worse as the input circuitry becomes slew limited.

5.1.3 Signal to Noise Ratio (SNR):

Signal-to-noise ratio (SNR) is a non-standardized measure of the dynamic performance of an ADC. In this work, it is defined as the ratio of the output RMS signal amplitude (the full scale -FS) to the RMS value of the noise spectrum including all non-fundamental spectral components without the fundamental signal itself, the harmonics, and DC component. The SNR usually degrades as frequency increases because the accuracy of the comparator(s) within the ADC degrades at higher input slew rates. This loss of precision shows up as noise at the ADC output.

$$\mathbf{SNR_{db}} = \mathbf{20\log}\left(\frac{\textbf{RMS value of FS input}}{\textbf{RMS value of quntization noise}}\right) = \mathbf{10\log}\left(\frac{\sigma^2_{\textbf{signal}}}{\sigma^2_{\textbf{Noise}}}\right) = \mathbf{10\log}\left(\frac{\mathbf{P_{signal}}}{\mathbf{P_{Noise}}}\right) \quad \textbf{(5.3)}$$

Where σ is the variance, and P is the power.

In any ADC, the noise has four sources: (1) quantization noise, (2) noise generated by the converter itself, (3) jitter, and (4) application circuit noise. The application circuit noise is that noise observed by the converter because of the way the circuit is designed and laid out. The fourth source of noise; the jitter, is not applicable in a CT-ADC, as it is a clock-less system that has no jittering or aliasing. The SNR performance increases with increasing input amplitude until the input signal approaches full scale. Then, increasing the input signal amplitude by 1 *dB* would cause an equal 1 *dB* increase in SNR. As a result, the step size turns into a smaller part of the total signal amplitude as the signal amplitude increases. On the other hand, if jitter exists, the SNR performance decreases at higher input frequencies because the effects of jitter get worse.

5.1.4 Signal-to-Noise and Distortion (SINAD)

This performance metric can be called signal-to-noise and distortion (SINAD) or signal-to-noise and distortion ratio (SNDR), or signal-to-noise plus distortion (S/N+D). It is a combination of the SNR and the THD specifications. It is defined as the RMS value of the output signal to the RMS value of all the other spectral components, including harmonics but excluding dc, and can be calculated from SNR and THD according to the following formula

$$\mathbf{SINAD_{db}} = \mathbf{10\log}\left[\frac{\mathbf{1}}{\mathbf{10}^{\frac{-SNR}{10}} + \mathbf{10}^{\frac{-THD}{10}}}\right] \qquad \textbf{(5.4)}$$

Thus, it can be defined as the ratio of signal power to the total noise power plus spurious harmonics power at the output when the input is a sinusoid. These spurious harmonics are those caused by circuit non-linearity.

$$\mathbf{SINAD} = \frac{\mathbf{P_{Signal}}}{\mathbf{P_{quantization\ error}} + \mathbf{P_{distortion}} + \mathbf{P_{random\ noise}}} \qquad \textbf{(5.5)}$$

Where, P for the average power of the signal, quantization error, random noise and distortion components [45] are used in the equation. In Summary, SINAD measures the ADC dynamic performance because it compares all undesired frequency components with the input frequency, as it is the ratio of the total received power.

5.1.5 Spurious-Free Dynamic Range (SFDR)

Spurious-Free Dynamic Range (SFDR) is the difference between the magnitude of the fundamental signal and the magnitude of the largest harmonic or strongest spurious signal (in *dB*). Thus, it is defined as the ratio of the RMS value of the carrier frequency (maximum signal component) at the input signal of the ADC to the RMS value of the harmonic distortion component or the largest noise (which is referred to as "spurious" or a "spur") at its output. While the SFDR is expressed in *dB* below the fundamental signal, it is sometimes expressed in negative *dB*. Although it is a range, it should be expressed in positive *dB* [46].

$$\mathbf{SFDR_{db}} = \mathbf{20\log_{10}}\left(\frac{\mathbf{Fundamental}}{\mathbf{Highest\ Spurious}}\right) \qquad \mathbf{(5.6)}$$

5.1.6 Signal-to-Quantization-Noise Ratio (SQNR)

In the clock-less ADC, the non-uniformed samples (emblem), consist of two parameters; time quantizer (T_q) , which represents the resolution in time and the delta space level (Δ), which represents the amplitude. Both parameters create the reconstructed digitized signal. The reconstructed output signal is compared to the original input signal by calculating the signal-to-quantization-noise ratio. Therefore, The SQNR is a non-standardized measure of the maximum achievable dynamic performance of an ADC. It is given by the ratio between the RMS signal amplitude and the RMS value of the spectral components, generated by the quantization noise. The amplitude of the quantization noise decreases as resolution increases because the size of an LSB is smaller at higher resolutions, which reduces the maximum quantization error, and is expressed in *dB* as in the following formula.

$$\mathbf{SQNR_{db}} = \mathbf{10\log_{10}}\left(\frac{\mathbf{P_{signal}}}{\mathbf{P_{Noise}}}\right) = \mathbf{10\log_{10}}\left(\frac{\mathbf{RMS_{signal}}}{\mathbf{RMS_{quantization\ noise}}}\right)^{\mathbf{2}} \qquad \mathbf{(5.7)}$$

5.1.7 Effective Number of Bits (ENOB):

The effective number of bits, or ENOB, is a specification that helps to quantify the ADC dynamic performance. ENOB states that the converter performs as if it were a theoretically perfect converter with a resolution of ENOB. Thus, if the ENOB is 7.5 bits that mean the converter performs, as far as SNDR is concerned, as if it were an ideal 7.5 bit ADC. So, the number of effective bits is another method of specifying SNDR. It illustrates that the converter is equivalent to a perfect ADC of this ENOB number of bits. The following equation can calculate the ENOB

$$\textbf{ENOB} = \frac{(\mathbf{SNDR_{db}} - \mathbf{1.76\ dB})}{\mathbf{6.02\ dB}} \qquad \textbf{(5.8)}$$

ENOB specifies the dynamic performance of the clock-less ADC at a specific input frequency and resolution ratio, it degrades as frequency increases and as input level decreases for the same reasons that THD and SNR degrade with frequency increase, and it improves as input level increases. There are two main calculations for the ENOB value. One is based on the SINAD (THD and SNR), and the other one is based only on SQNR. Here, the more common method of basing the ENOB value on the SINAD is used. Compared to a calculation that uses only SQNR the figure will be worse.

5.1.8 Resolution Ratio (R)

In the conventional ADC system, the input signal is sampled at precise time periods, and the signal amplitude is estimated by the digital value, which leads to quantization noise due to the variability in the amplitude estimation. Therefore, the SNR will be expressed as a function of the number of bits (*N*). SNR=6.02 *N*+1.76. However, the continuous time ADC is sampled by a signal crossing the threshold delta modulation levels (quantization levels), which signifies that the time between two consecutive samples is indefinite and that the quantization noise is moved from amplitude to time. As the time between two consecutive crossings is equal to N T_q, where N is the integer number of non-uniformed sampled emblem. The R is variable across the input signal according to the quantization time. The equation of the resolution ratio is

$$\boldsymbol{R = \frac{1}{T_q f_{i/p}}} \quad \textbf{(5.9)}$$

Where $f_{i/p}$ is the input signal frequency.

As proven in [47], [48], the SNR of the level-crossing ADC depends only on the resolution ratio.

$$\textbf{SNR = 20 log R- 14.2} \quad \textbf{(5.10)}$$

$$\textbf{R= } \boldsymbol{10^{\frac{SNR+14.2}{20}}} \quad \textbf{(5.11)}$$

Where R is the Resolution ratio.

5.2 Simulation for CT- ADC Using MATLAB

The new clock-less system using wavelet neural network presented in this work is simulated in MATLAB to study its characteristics after each stage to demonstrate the consecutive signal improvement. In the following subsections, one sinusoidal input signal (1*Hz*) is applied with a signal resolution 0.1*μs (100 ns)* to each stage of a 4-*bit* clock-less ADC with 16 quantization noise levels (presented in Chapter 4, Figure 4.1). The input signal is 1 *volt* (± 0.5 *volt*).

5.2.1 Clock-less ADC (Delta Modulation)

Converting analog to a digital signal using delta modulation is the core stage in creating a continuous-time converter. At this stage, a sinusoidal input signal ($V_{i/p}$) is applied to the comparators to be compared to the (V_{ref}). The output of comparators is then fed into the control logic and digital logic blocks, which build the delta modulation output signal (V_{DM}). The V_{DM} is converted to analog signal again, which is called V_{ref}, using a digital to analog converter. The V_{ref} is the reference signal for the next input signal.

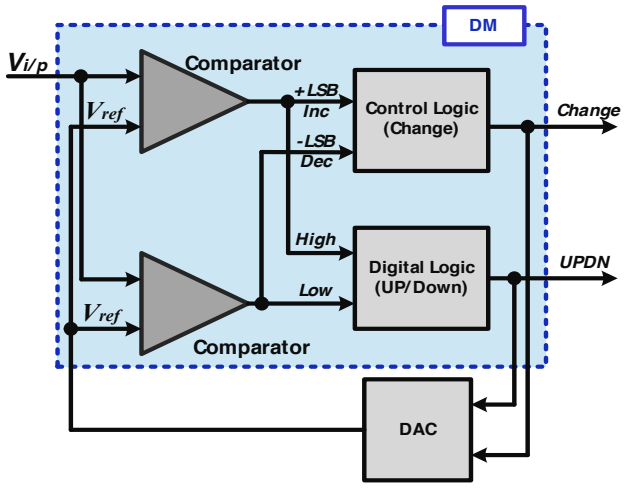

Figure 5.1: Block diagram of delta modulation ADC

The output signal of the delta modulation ADC (V_{DM}) is a digital continuous time signal that is discrete in amplitude and continuous in time. The V_{DM} is not smooth in shape and has high quantization noise error. The quantization error is calculated by subtracting the output signal from the input signal, it has a bell-sawtooth shape, and its peak-to-peak amplitude is equal to one delta modulation level (1 *LSB*) as shown in Figure 5.2. It shows the sinusoidal input signal ($V_{i/p}$) and the output signal of delta modulation (V_{DAC}) and the quantization error between the input signal and V_{DAC}.

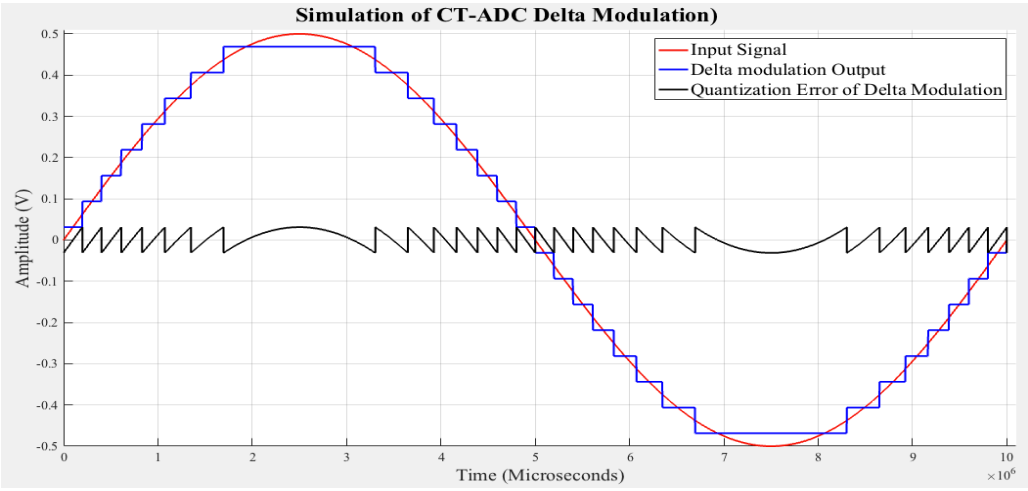

Figure 5.2: Simulation of clock-less ADC using delta modulation for 1 Hz

The total number of quantization levels of 4 bits is (2^4) = 16 levels. In many references like [5], [7], the authors used 15 levels (2^4- 1), for the 4 bits ADC, to align the mid-tread quantizer. In other words, aligning the quantization levels around the zero-input can be achieved by keeping odd symmetry quantization levels around the zero-input. However, in this proposal, all 16 levels are used for two reasons. First, the offset amplitude shifting will shift the input signal in one direction (-Δ/4, -Δ/2, and -3* Δ/4) as will be discussed in (section 5.2.2). Second, the aim of this work is to improve the signal resolution, and using this extra level increases the signal resolution. As shown in Figure 5.2, 16 quantization levels are used, seven above the zero-input and eight levels under the zero-input. The output signal is then normalized to fit the input signal.

The ENOB for delta modulation ADC output signal (V_{DM}) is 3.2 *bits*, and the SQNR is 19.68 *dB.* Thus, the SNDR is 20.90 *dB,* and SFDR is 23.67 *dB*. These results can be increased to approach high resolution by using adaptive resolution as will be discussed in the next stage.

5.2.2 Clock-less ADC Adaptive Resolution

As discussed previously, delta modulation can track the input signal only when it crosses the quantization levels, and the signal behavior cannot be detected or traced between any two consecutive quantization levels. This fact limits the resolution of the output signal. Therefore, it is decided to duplicate the input signal by shifting the amplitude three times, each time by -Δ/4. Then, all the four signals are applied to the delta modulation to predict and interpolate the signal activities between the quantization levels. The fixed time delay between the input signal and its duplication signals is assumed ideal and negligible, as the digital logic accumulator and control logic accumulator can adjust this time delay for the output signal (V_{ref}). Figure 5.3 shows the input signal and its duplication signals.

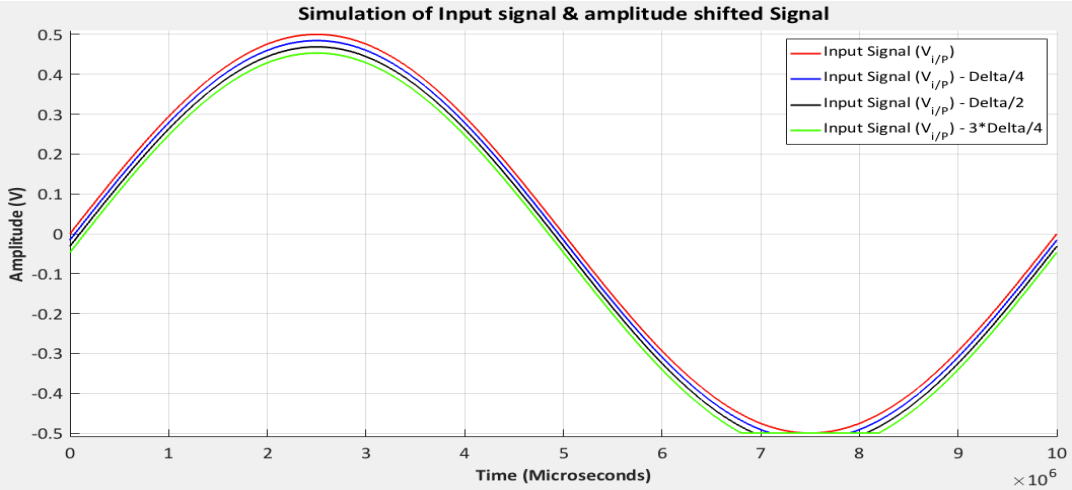

Figure 5.3: Simulation of input signal and amplitude shifted signal

At this stage, the adaptive resolution is used to enhance the signal quality of V_{DM} by duplicating the input signal three times and shifting the signal in amplitude by -Δ/4, -Δ/2, and - 3* Δ/4 consequently. The shifting is in negative amplitude from zero-input level. Thus, the duplicated signals are cutting at -0.5 volt because of the input signal range (± 0.5 volt). These amounts of signal shifting will guarantee that those four signals will never cross any quantization level at the same time, which means that each comparator will detect four signals with different T_q when the level is crossed.

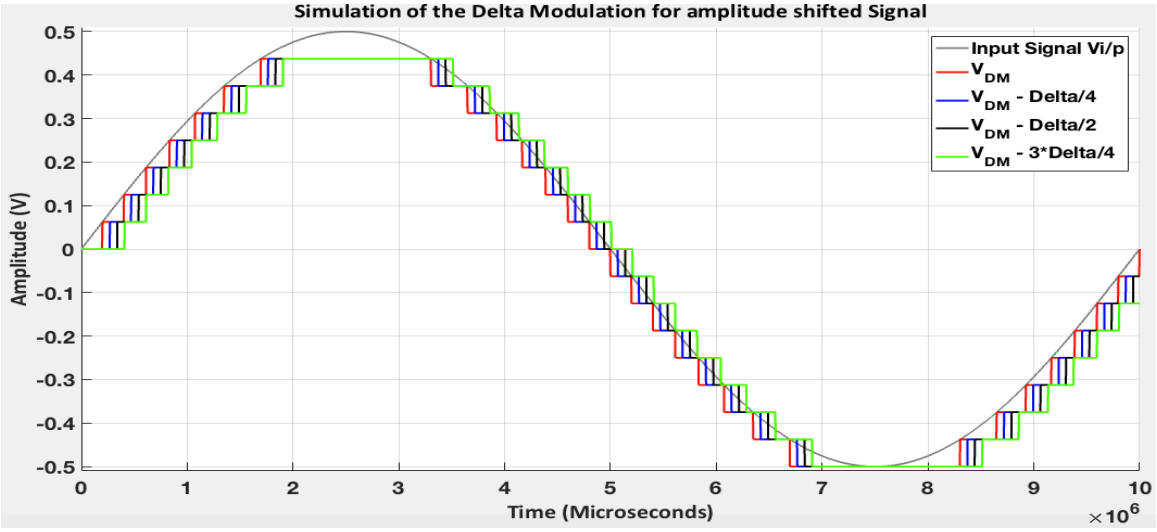

Figure 5.4: Simulation of the delta modulation for amplitude shifted signal

The four signals pass the same quantization levels at different times according to their shifting and in fact, result in four V_{DM} duplicated in shape but shifted in time. The four output V_{DM} are then encoded by the CT change detector block, which analyzes each offset signal according to their T_q The input sinusoidal signal and the four output V_{DM} are shown in Figure 5.4. The goal of this procedure is to increase the quantization levels by dividing each delta modulation level to four virtual quantization levels, which will increase the sixteen levels crossing (delta modulation levels) to sixteen physical levels crossing in addition to another forty-eight virtual crossing levels. The total sixty-four levels (physical and virtual) increase the resolution of the output signal from 4 bits to 6 bits according to Table 4.1.

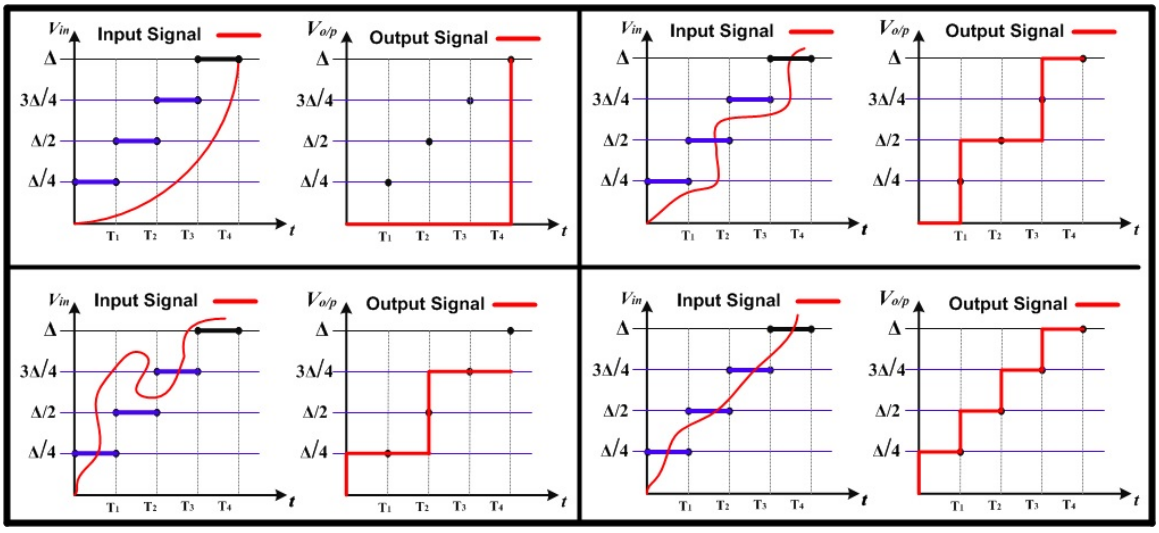

Figure 5.5: Theory of operation for accumulator block

(Blue lines): Virtual quantization levels, (Black lines): Physical quantization levels, (Red lines): input/output signals

Figures 5.5, shows the theory of operation for a CT change detector block, which acts as an intelligent accumulator, as each delta modulation step is divided into four with three new levels with different amplitudes (blue lines). In the figure, the input/output signals are plotted versus time quantizers $T_{q1}, T_{q2}, T_{q3}, \ T_{q4}$. It is clear in the figure that the

input signal must cross the virtual level at certain T_q intervals to be detected at the intelligent accumulator.

At the top left graph of Figure 5.5, the input signal does not cross any of the three virtual delta modulation steps during the corresponding time quantizer intervals. As a result, the output signal formed has no amplitude changes between T_{q1} and T_{q3} and the amplitude changed only at T_{q4} when the input signal crossed the physical quantization levels. Similarly, in the top right graph, the input signal passes neither the first virtual level $\Delta/4$ nor the third virtual level $3\Delta/4$ during the assigned time intervals. As a result, the output signal recorded amplitude changes only at T_{q2} and at T_{q4}. The bottom left figures go through similar scenarios. The best case is shown in the bottom right graph, when the input signal crosses the physical and virtual levels at the corresponding time quantizer intervals resulting in the output with the best resolution among all four outputs. In summary, these virtual levels, caused by shifting the input signal amplitude, improve the signal tracking between any two consequence physical quantization levels.

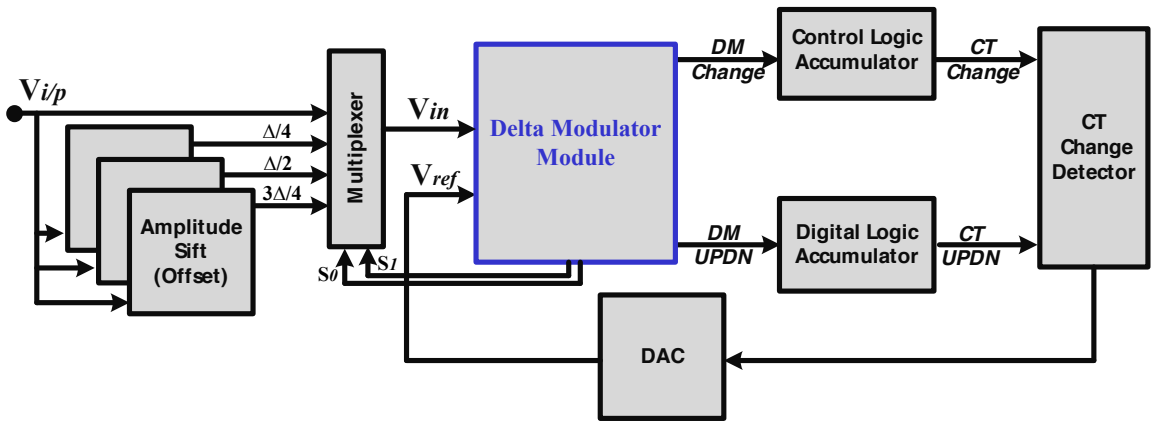

Figure 5.6: Block diagram of delta modulation ADC with amplitude shifting

The output signal of delta modulation ADC with amplitude shifting is then converted back to an analog signal, which is called V_{ref}, using a digital to analog converter. The V_{ref} generated after inducing the virtual quantization level has a lower quantization error

noise, which is variable and depends on the slope. In the best-case scenario, the quantization error noise peak-to-peak amplitude is equal to $\Delta/4$. Thus, the signal resolution of V_{ref} with virtual quantization levels is better than that of V_{ref} reproduced from the delta modulation output signal V_{DM}. Figure 5.7 shows the sinusoidal input signal $V_{i/p}$, and the output signal of amplitude shifted delta modulation V_{ref} and the quantization error between the input signal $V_{i/p}$, and V_{ref}.

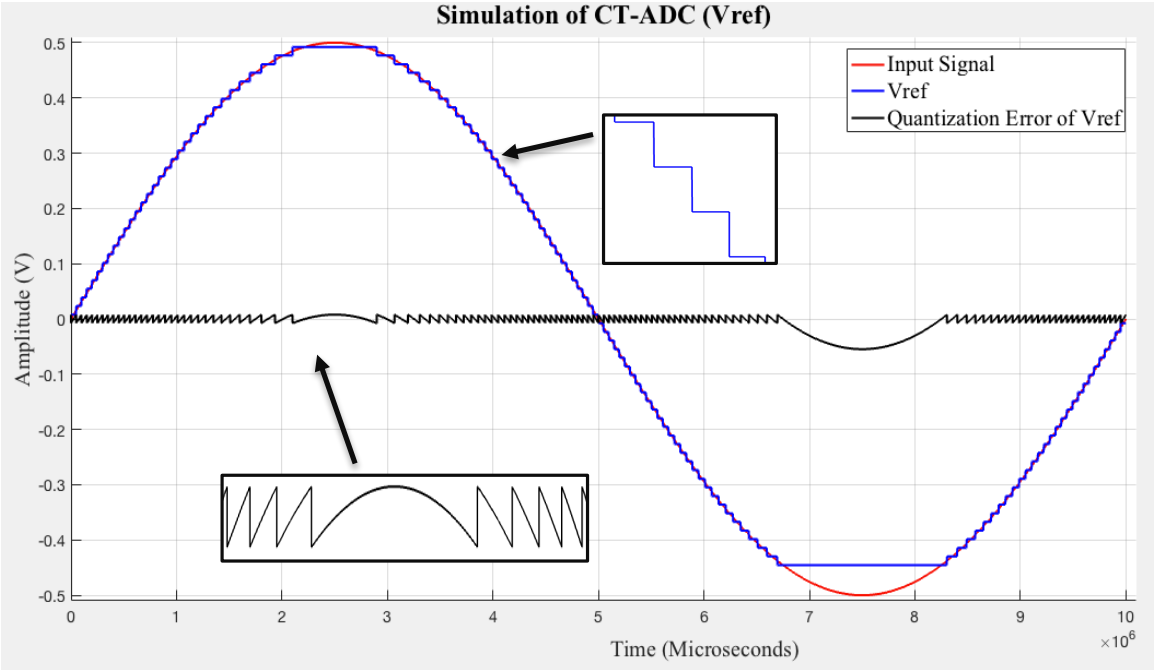

Figure 5.7: Simulation of clock-less ADC using delta modulation with amplitude shifting for 1 Hz

The results of the MATLAB simulation for the same input achieved improvements in signal resolution with SQNR, SNDR, and SFDR achieving 20.76.68 *dB,* 32.94 *dB*, and 36.99 *dB* respectively. ENOB achieved a significant progress and increased from 3.18 bits to 5.18 bits with shifted amplitude signal V_{ref}. This variable resolution is not fixed and depends on the slope of the input signal and signal behaviors between the physical quantization levels. However, the two bit improvement is the maximum achievement with the best-case behavior of input signal.

5.2.3 Combining Quantization Error with CT- Signal

In this stage, the V_{ref} which resulted from the amplitude shifted delta modulation ADC is used for a third improvement of the signal resolution. This improvement is achieved by subtracting the V_{ref} from the $V_{I/P}$ to compute the quantization noise of V_{ref}. Next, the quantization error is amplified to magnify the errors, then digitized using the same basic delta modulation block, and then encoded to binary code to combine the continuous-time signal (V_{ref}) with quantization noise. The combination occurs by replacing the LSB of V_{ref} with the most significant bits (MSB) of the quantization noise signal, plus shifting the quantization noise to increase the number of bits that represent the combined signal ($V_{Combined}$) as detailed in Section 4.14.

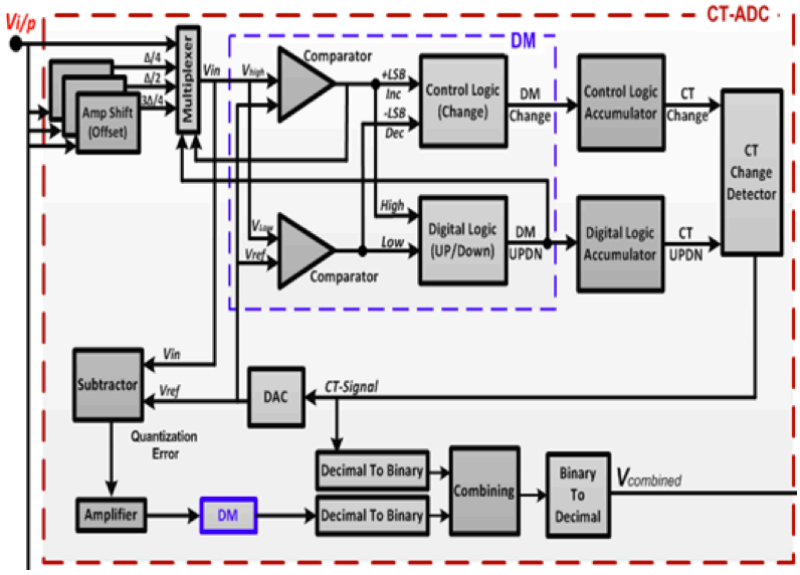

Figure 5.8: Block diagram of combined signal ($V_{Combined}$)

This stage improves the signal resolution and reduces the quantization noise of the saw-tooth shape from ($\Delta/4$) to ($\Delta/8$). This improvement is considerable because most important data, especially in continuous-time systems and burst-like signal applications, are usually reconstructed into a saw-tooth shape. Figure 5.9, shows the sinusoidal $V_{i/p}$ and the $V_{Combined}$ and the quantization error between $V_{i/p}$ and $V_{Combined}$.

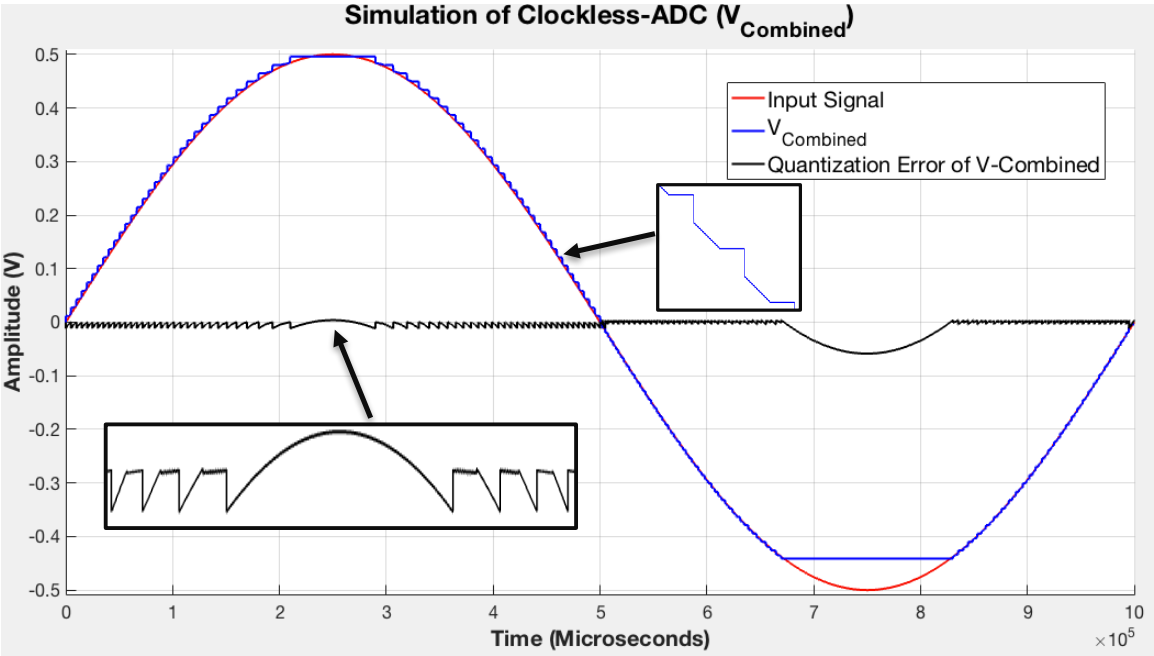

Figure (5.9): Simulation of clock-less ADC combined output signal ($V_{Combined}$) for 1 Hz

The MATLAB simulation did not record significant enhancement of the signal quality because bell-shaped quantization error is already included in all calculations. Therefore, SQNR, SNDR, and SFDR recorded 26.18 *dB,* 36.35 *dB,* and 43.35 *dB* respectively. ENOB has achieved a small increase from 5.18 bits to 5.74 bits.

5.2.4 Clock-less ADC using WNN

In the last stage of the proposed clock-less ADC, the wavelet neural network is applied to boost the resolution of $V_{Combined}$ even more. $V_{Combined}$ is the input signal *X(t)* for the WNN system and is to be processed to reach the desired output signal *d(t)* with minimal *E(t)*. The minimal *E(t)* is less than or equal to the value of the signal resolution. For example, if the signal resolution is one µsec, then the WNN will keep running until the minimal error *E(t)* <=one µsec. The *d(t)* is the $V_{i/p}$ with a time shift delay. The time shift delay is a fixed time equal to the processing time for clock-less ADC stages starting from applying the $V_{i/p}$ until observing the $V_{Combined}$.

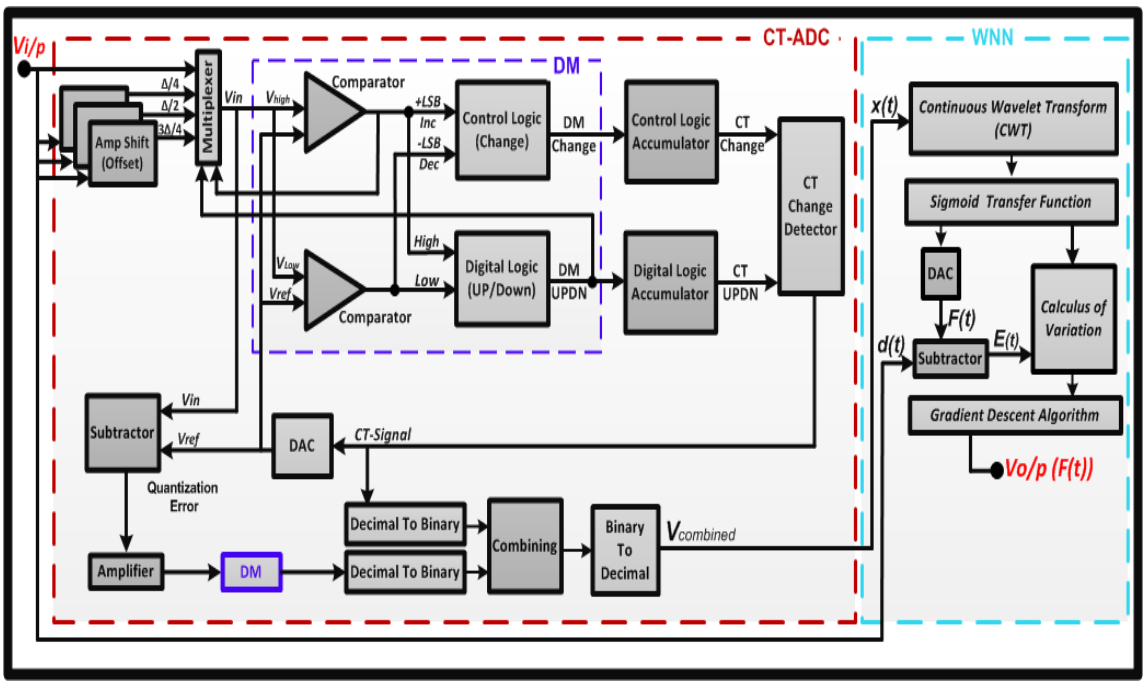

Figure 5.10: Block diagram of combined signal

The wavelet neural network procedure starts by analyzing the *X(t)* using CWT to reach the wavelet network coefficients that represent the input signal. This process is repeated for a number of times, which depends on the number of wavelons, to achieve the best wavelet coefficient values. MATLAB simulations are run with only twenty wavelons in the hidden layer. This amount of wavelons is enough to represent the signal with a high resolution in a short time. The wavelet coefficients are represented by using the Sigmoid transfer function to smooth the shape of the WN output signal *F(t)*.

Then, the WN output signal is subtracted from the *d(t)* $V_{i/p}$ to compute the wavelet neural network error *E(t)*. Then, the variation calculation is performed to adjust the wavelet coefficients (weight, scaling and shifting parameters) to reduce the quantization error to a value less than or equal to the minimal accepted error *E(t)*. In fact, the process of variation calculation to predict the right values for (weight, scaling and shifting) aims for an error equal to zero. The gradient descent algorithm is training the wavelet neural network, by repeating the process of calculating the variation with adding some

adjustment parameters, is to speed and guarantee a high output resolution. These parameters are momentum factor β and learning rate η; these are fixed and are recalculated for each circuit design depending on the input dynamic range and the input frequency range. In the MATLAB simulation a momentum factor β equal to 12.7 nm, and a learning rate η equal to 31.3 are used. The learning rate *i* is directly proportional to the delay time of the WNN. Therefore, a learning rate *i* = 100 iterations is used for the MATLAB simulation.

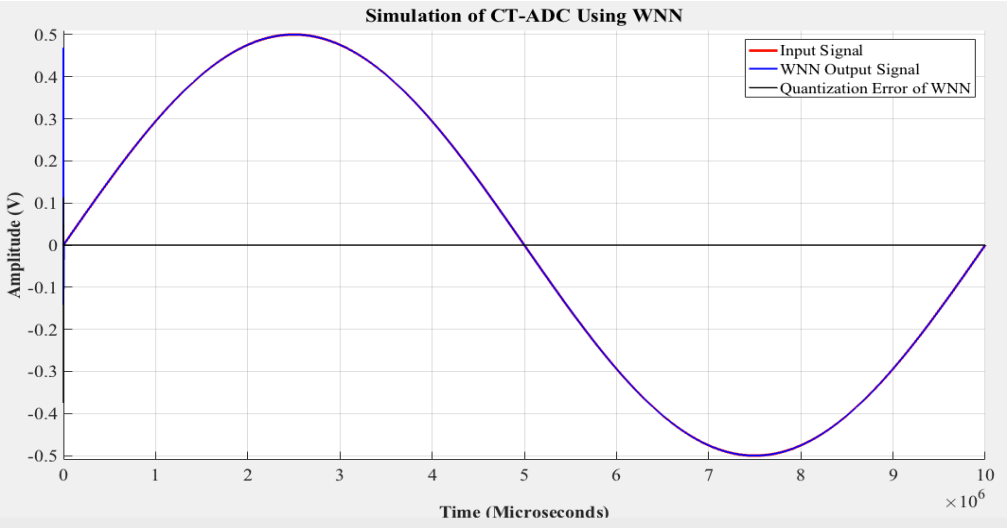

Figure 5.11: Simulation of clock-less ADC wavelet neural network (V_{WNN}) for 1 Hz

The results of the MATLAB simulation for the proposed CT-ADC (including the WNN) with an input signal of 1 *Hz* and 0.1μs time resolution, shows significant improvement in signal resolution with almost zero quantization error: SQNR is 67.29 *dB,* SNDR is 232.89 *dB,* and SFDR is 253.89 *dB.* Also, ENOB has achieved enormous enhancement, which recorded 38.39 *bits.* This signal improvement is the result after several stages of improvement clock-less ADC. The quantization error range is between – 3.5 e^{-7} to – 3.5 e^{-7} *volt*, which equal to Δ/89286.

5.2.5 Simulation for Several Input Frequencies

In this section, the same MATLAB example with the same parameters will be simulated using an input signal dynamic range of 1 *volt* (0.5 *volt* ~ -0.5 *volt*), with a 4-bit CT-ADC (16 quantization noise levels), and a signal resolution of 0.1μs (100*ns*), but at different input frequencies: 1K*Hz*, 4K*Hz*, and 20 K*Hz*. Also, the signal distortion will be studied at each input signal to characterize the specifications of ADC-WNN.

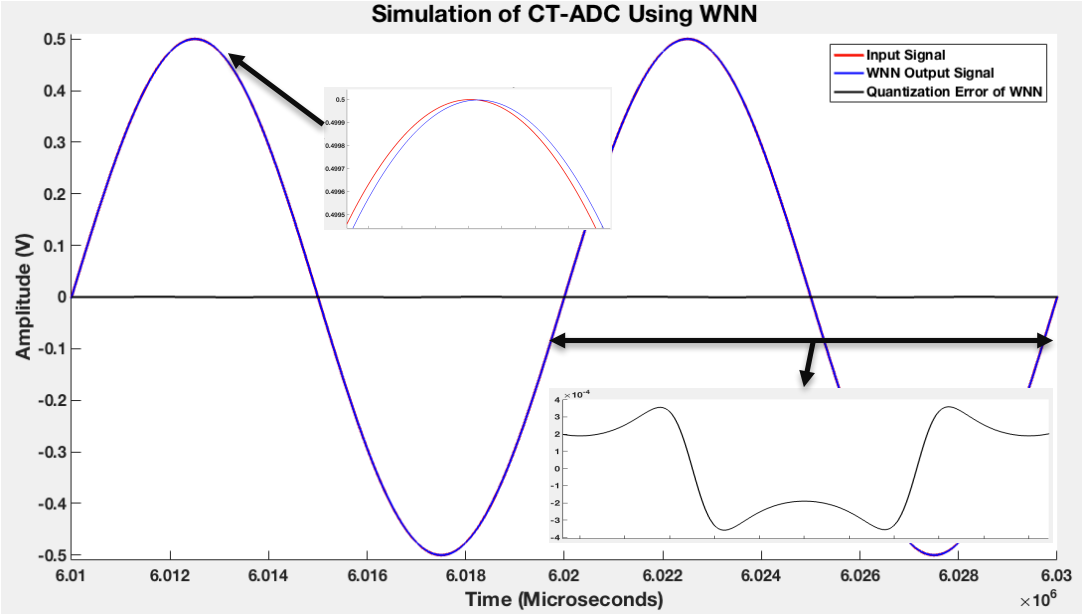

Figure 5.12: Simulation of clock-less ADC wavelet neural network (VWNN) for 1 KHz

In Figure 5.12, a regular sinusoidal input signal 1KHz is applied to the WNN system. Thus, the delta modulation stores 1000 frequencies at CT-DSP latch [7], then buffering them to the WNN system to get them processed all together. As a result, the signal resolution is a little blurry and lower than one frequency. Therefore, SQNR is reduced to 62.57 *dB*, SNR is reduced to 86.46 *dB*, SNDR is reduced to 62.60 *dB*, and the SFDR is reduced as well to 69.31 *dB*. Furthermore, the effective number of bits is reduced to 11.11 bits. The output signal of the WNN system is still achieving higher resolution than that of delta

modulation or $V_{Combined}$. As shown in Figure 5.12 the output signal of WNN has a small distortion of the bell-shape. The quantization error is in the range from -0.0004 *volts* to 0.0004 *volts,* which is equal to Δ/78.

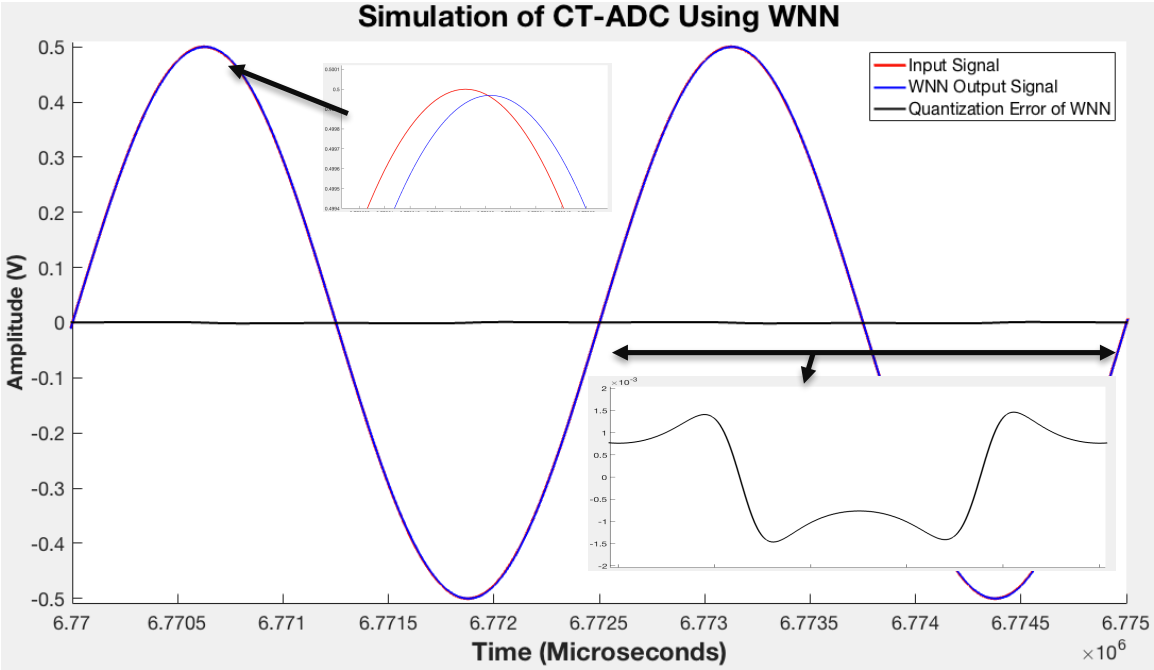

Figure 5.13: Simulation of clock-less ADC wavelet neural network (VWNN) for 4 KHz

In Figure 5.13, an input sinusoidal signal at 4 *KHz* is applied to the CT-WNN system. Hence, the delta modulation stores digitized tokens for 4000 frequencies and buffering them all together to the WNN system. Although the input signal frequency increased to 4 *KHz*, the WNN uses the same number of hidden layers and iterations. Therefore, the WNN output signal achieves high-resolution measures, but less precise than those obtained for input signals at 1 *KHz* or 1 *Hz*. At 4 *KHz*, SQNR is 52.17 *dB*, SNR is 74.42 *dB*, SNDR is 56.6 *dB*, and SFDR is 57.27 *dB*. As a result, the effective number of bits is reduced to 9.1 bits.

The output signal also has more distortion at the bell-shape, and the quantization error range increases to be in the range from -0.0015 *volts* to 0. 0.0015 *volts*, which is equal to Δ/20.

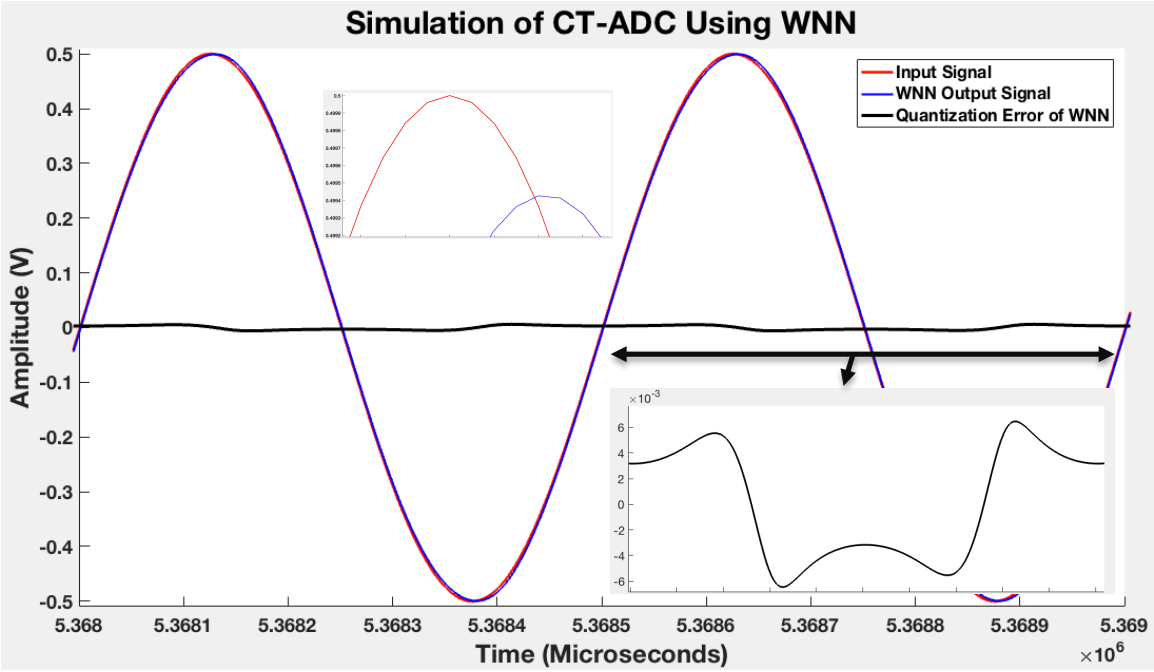

Figure 5.14: Simulation of clock-less ADC wavelet neural network (VWNN) for 20 KHz

Similarly, Figure 5.14 is showing the distortion of the output signal when the input signal is at 20 *KHz*. The distortion is higher at the bell-shape, and the quantization error range is between -0.005 and 0.005 *volts*, which is equal to $\Delta/5$. SQNR for the output signal reaches 38.34 *dB*, SNR reaches 60.47 *dB*, SNDR reaches 42.7 *dB*, and SFDR reaches 43.32 *dB*. As a result, the effective number of bits is reduced to 6.8 bits. However, the effective number of bits for CT-WNN system is still greater than that of delta modulation and $\mathbf{V_{Combined}}$, but it is getting close to their records. Therefore, the highest input signal frequency for a signal resolution of 0.1μs is 20 *KHz*.

5.3 Simulated CT-ADC Performance

The popular specifications for quantifying ADC dynamic performances are presented in Section 5.1. There are several ways to quantify the distortion and noise of ADC based on FFT analysis. The FFT output analysis is used to measure the amplitude of the various harmonics and noise components of digitized signals. The CT-ADC performance is

studied in the next section, using the same MATLAB example with the same parameters and simulated with input frequency 1 *KHz*, and signal resolution 0.1*μs (100 ns)*.

5.3.1 SINAD for CT-ADC Performance

The SINAD measurement is widely used for measuring and specifying the sensitivity of a radio receiver. The SINAD can be summarized as the ratio of the total signal power level (Signal + Distortion + Noise) to unwanted signal power (Distortion and Noise). Consequently, the higher SINAD, the better the quality of the audio signal.

Plotting the power spectrum of the SINAD for CT-ADC signals can help understand its performance. The SINAD function estimates a noise level using the median power in the regions containing only noise and distortion. Thus, the DC component (DC level and the fundamental signal) is excluded from the SINAD calculation. The noise at each point is the estimated level or the ordinate of the point, which is smaller than the fundamental signal. The noise is subtracted from the values of the fundamental signal and the harmonics. Therefore, if the fundamental is not the highest spectral component in the signal, SINAD fails.

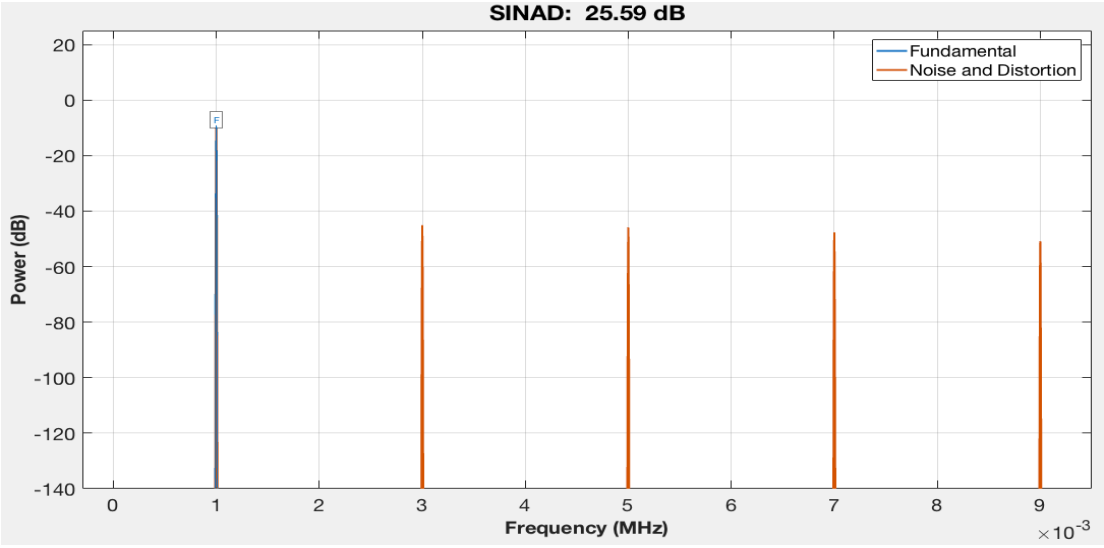

Figure 5.15: SINAD for delta modulation output signal

The spectra of the delta modulation ADC output are obtained by applying the FFT to the DM-ADC output in MATLAB. The spectrum shows the given sinusoidal input signal with a frequency of 1 *KHz*; which includes the fundamental tone plus the in-band harmonics at the output of a DM-ADC. As shown in Figure 5.15, there is neither even-order harmonics nor any noise floor present in the spectrum (2^{nd} and 4^{th} harmonics = zero), due to the symmetry characteristic of Fourier coefficients [5]. The fundamental input frequency 1 *KHz* recorded a SINAD = 25.59 *dB*. The SINAD is often converted to ENOB and achieves 3.18 bits in this case.

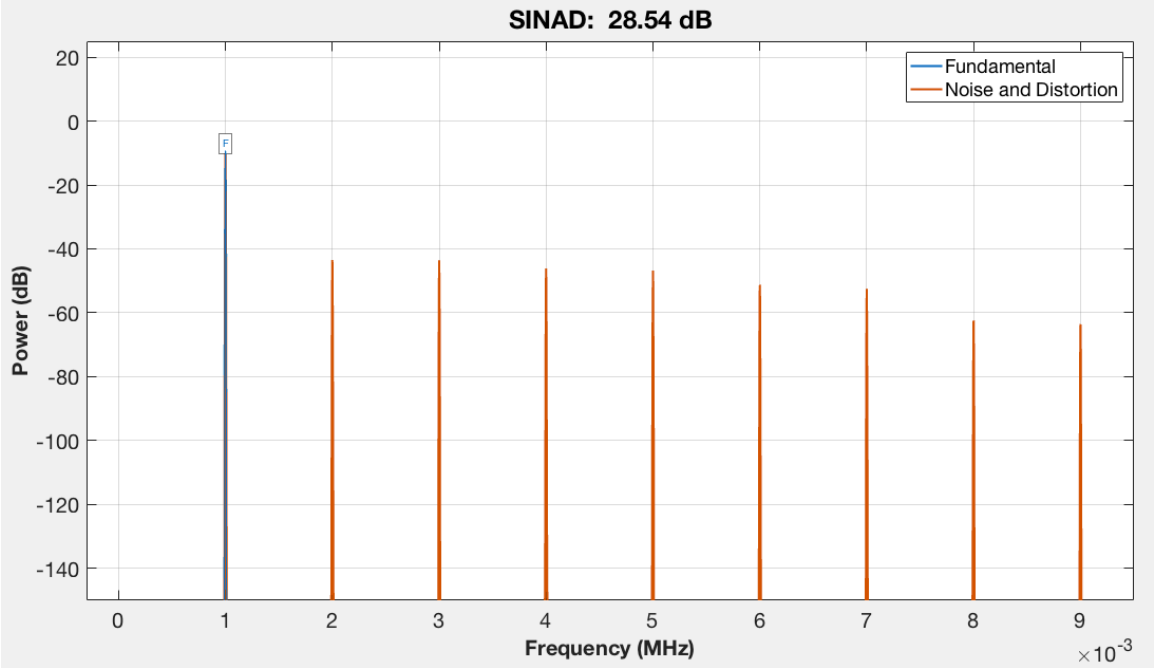

Figure 5.16: SINAD for V_{ref} output signal

Similarly, the spectra of the V*ref* output signal is generated by applying the FFT to V_{ref} output signal in MATLAB. The spectrum in Figure 5.16 shows the output signal of V_{ref} with fundamental frequency 1 KHz along with the noise and distortion harmonics at other frequencies. The even and odd harmonics are shown in the spectrum because V_{ref} is not symmetric. SINAD equals 28.54 *dB*, and ENOB equals 5.74 bits.

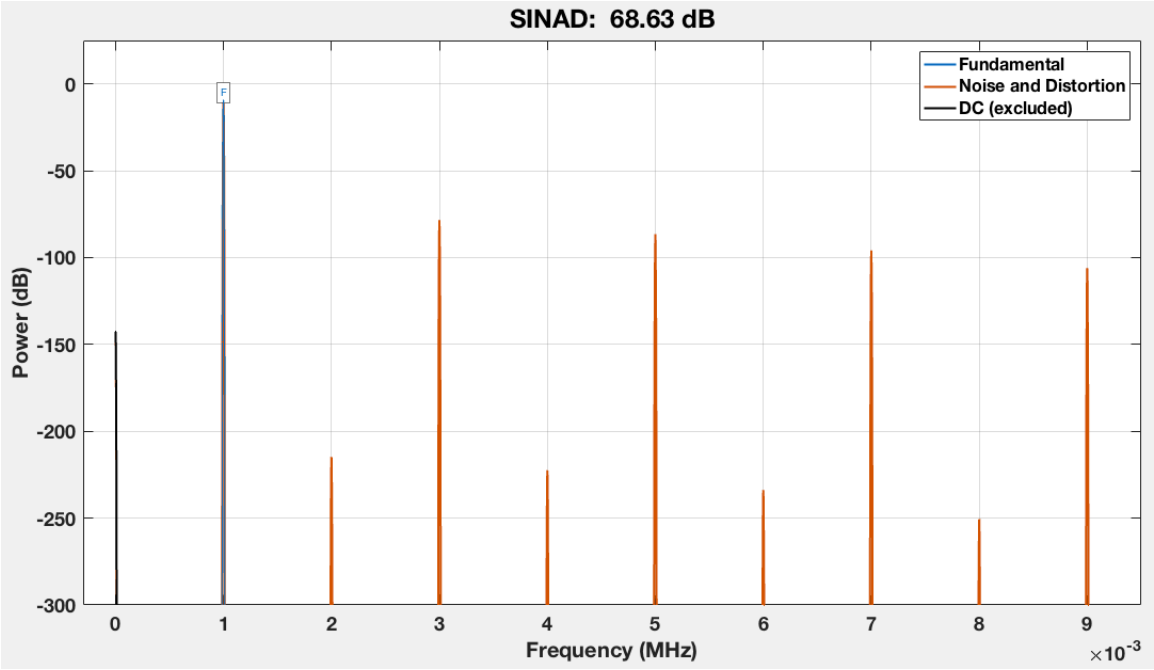

Figure 5.17: SINAD for WNN output signal

Figure 5.17 shows the spectra of the WNN output V_{WNN} with fundamental frequency 1 *KHz* and noise and distortion harmonics. The even and odd harmonics are shown. The output signal is very accurate and almost symmetric. Even harmonics have very small *dB* values in the spectrum. The recorded SINAD is 68.63 *dB,* and ENOB is 11.11 bits.

SINAD is used as a basic measurement technique for analog systems and is an important (and fundamental) performance parameter. A high SINAD indicates an efficient use of spectrum space and would improve the signal clarity of low-level receiver signal.

5.3.2 SFDR for CT-ADC Performance

SFDR computes the quantity of distortion in the system, and it is one of the most important AC performance specifications. SFDR is the ratio between the fundamental signal and the highest spurious in the spectrum. It simply defines the capability of the ADC system to detect a carrier signal (fundamental signal) in the presence of noise or any other spurious frequency. At any high-resolution ADC, the SFDR should be

considered in selecting a wideband analog ADC for a high-performance system. The SFDR typically dominated by the dynamic range between a fundamental frequency and the second or third harmonic of the fundamental frequency.

SFDR is a ratio measured in *dB*. Therefore, SFDR is dependent on bit resolution. Thus, SFDR is characteristically better at lower frequencies than higher frequencies.

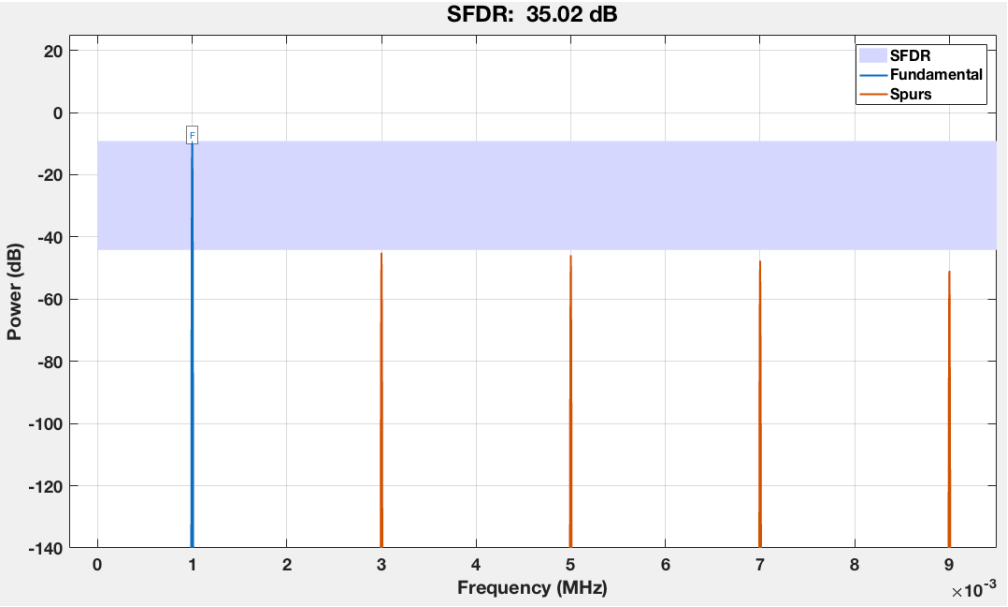

Figure 5.18: SFDR for delta modulation (V_{DM}) output signal

Figure 5.18 plots the power spectrum for the delta modulation output signal that shows a fundamental peak at 1 *KHz*. The resolution bit recorded is 3.18 bit as computed previously. Therefore, SFDR reaches 35.02 *dB.*

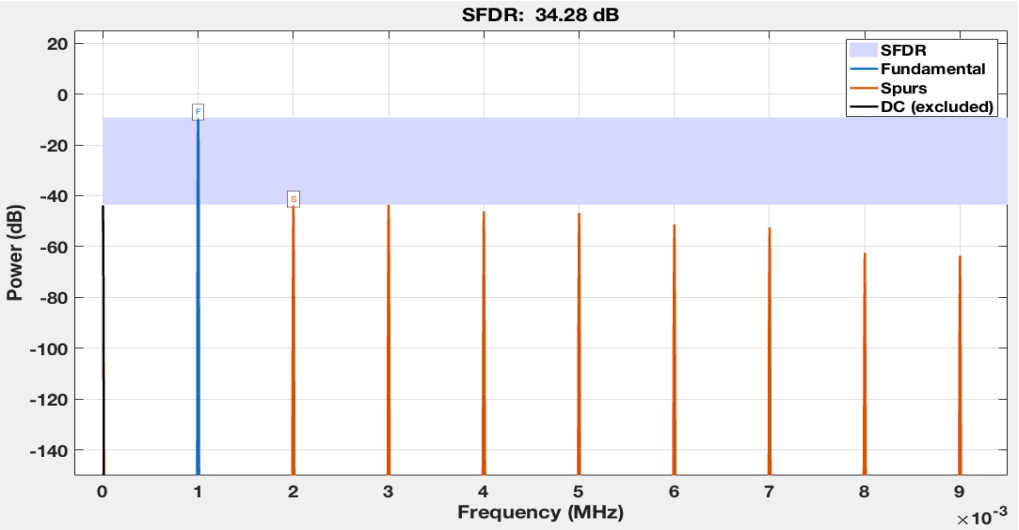

Figure 5.19: SFDR for Vref output signal

Figure 5.19 depicts the power spectrum of the V_{ref} output signal that shows a fundamental peak at 1 KHz, and achieves an ENOB of 5.74 bits; the SFDR is 34.94 *dB*.

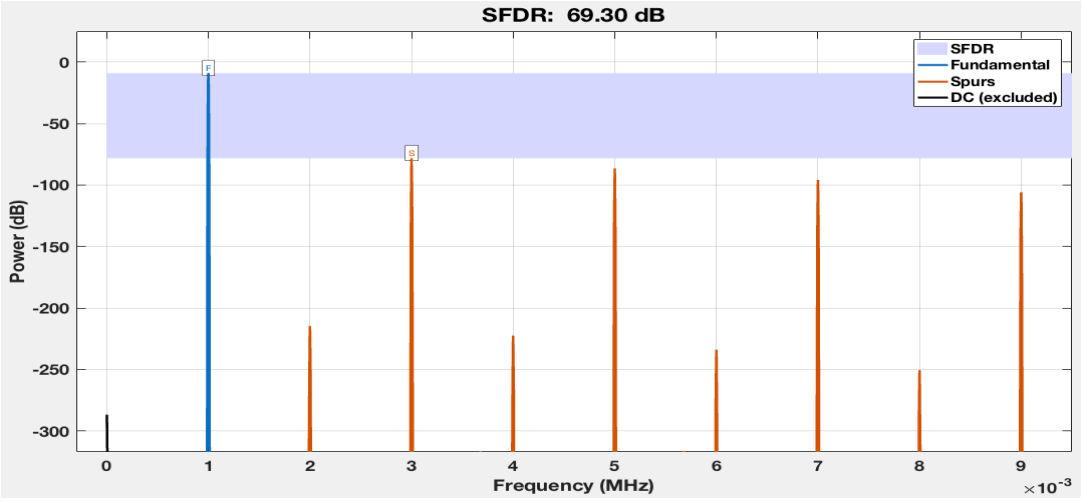

Figure 5.20: SFDR for V-WNN output signal

Finally, Figure 5.20 shows the power spectrum of the WNN output V-*WNN* with fundamental frequency 1 *KHz*. The SFDR is 69.30 *dB,* and the associated ENOB measures 11.11 bits. In general, SFDR can range from approximately 23 ~45 *dB* for ADC/DAC with 8 bits, or 55 ~ 90 *dB* for mid-range ADC/DAC with under 16 bits, to approximately 250 *dB* at 38 bits.

5.4 Monte Carlo Simulation Summary and Discussion

As a design example, a 4-bit (16-level) level-crossing ADC with an ideal comparator using 1 volt peak-peak (1 V_{pp}) and a clock resolution R=$10^{(\frac{\mathbf{SNR+14.2}}{\mathbf{20}})}$ is used and simulated on MATLAB and chosen to match the required performance criteria. The input signal used throughout the simulations is a single-tone sinusoidal run at several frequencies.

Table 5.2: Simulation results for the three systems (delta modulation system, adaptive resolution system, and WNN system) at different resolutions with applying several frequencies

Freq.	Input Resolution	Technology	SQNR	SNR	SNDR	SFDR	ENOB
1 Hz	*Tq* = 1μ Fs=1Mhz	Delta Modulation	**25.31 *dB***	**20.90 *dB***	**20.90 *dB***	**23.67 *dB***	**3.18 bit**
		V_{ref}	**26.18 *dB***	**36.35 *dB***	**36.32 *dB***	**43.35 *dB***	**5.74 bit**
		WNN O/P	**57.29 *dB***	**220.64 *dB***	**220.64 *dB***	**233.91 *dB***	**36.36 bit**
	Tq = 100n Fs=10Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.51 *dB***	**4.34 bit**
		WNN O/P	**67.29 *dB***	**231.17 *dB***	**231.17 *dB***	**253.90 *dB***	**38.11 bit**
	Tq = 12.5n Fs=80Mhz	Delta Modulation	**25.31 *dB***	**20.90 *dB***	**20.90 *dB***	**23.67 *dB***	**3.18 bit**
		V_{ref}	**26.18 *dB***	**36.35 *dB***	**36.32 *dB***	**43.35 *dB***	**5.74 bit**
		WNN O/P	**67.29 *dB***	**232.9 *dB***	**232.9 *dB***	**253.89 *dB***	**38.4 bit**
100 Hz	*Tq* = 1μ Fs=1Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.49 *dB***	**4.34 bit**
		WNN O/P	**56.51 *dB***	**86.45 *dB***	**68.64 *dB***	**69.31 *dB***	**11.11 bit**
	Tq = 100n Fs=10Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.49 *dB***	**4.34 bit**
		WNN O/P	**67.21 *dB***	**106.46 *dB***	**88.63 *dB***	**89.31 *dB***	**14.43 bit**
	Tq = 12.5n Fs=80Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.49 *dB***	**4.34 bit**
		WNN O/P	**76.31 *dB***	**124.52 *dB***	**106.7 *dB***	**107.37 *dB***	**17.43 bit**
1 KHz	*Tq* = 1μ Fs=1Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.49 *dB***	**4.34 bit**
		WNN O/P	**44.14 *dB***	**66.47 *dB***	**48.65 *dB***	**49.31 *dB***	**7.78 bit**
	Tq = 100n Fs=10Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.49 *dB***	**4.34 bit**
		WNN O/P	**62.57 *dB***	**86.46 *dB***	**68.63 *dB***	**69.31 *dB***	**11.11 bit**
	Tq = 12.5n Fs=80Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.49 *dB***	**4.34 bit**
		WNN O/P	**75.37 *dB***	**104.52 *dB***	**86.69 *dB***	**87.37 *dB***	**14.11 bit**

Freq.	Input Resolution	Technology	SQNR	SNR	SNDR	SFDR	ENOB
4 KHz	*Tq* = 1μ Fs=1Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.49 *dB***	**4.34 bit**
		WNN O/P	**32.35 *dB***	**54.47 *dB***	**36.74 *dB***	**37.39 *dB***	**5.81 bit**
	Tq = 100n Fs=10Mhz	Delta Modulation	**25.31 *dB***	**26.39 *dB***	**25.58 *dB***	**35.13 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.88 *dB***	**32.48 *dB***	**4.34 bit**
		WNN O/P	**52.17 *dB***	**74.42 *dB***	**56.60 *dB***	**57.27 *dB***	**9.11 bit**
	Tq = 12.5n Fs=80Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.49 *dB***	**4.34 bit**
		WNN O/P	**69.39 *dB***	**92.48 *dB***	**74.66 *dB***	**75.33 *dB***	**12.11 bit**
20 KHz	*Tq* = 1μ Fs=1Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.49 *dB***	**4.34 bit**
		WNN O/P	**18.80 *dB***	**39.70 *dB***	**24.06 *dB***	**24.65 *dB***	**3.70 bit**
	Tq = 100n Fs=10Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.49 *dB***	**4.34 bit**
		WNN O/P	**38.33 *dB***	**60.47 *dB***	**42.66 *dB***	**43.32 *dB***	**6.79bit**
	Tq = 12.5n Fs=80Mhz	Delta Modulation	**25.31 *dB***	**26.40 *dB***	**25.60 *dB***	**35.03 *dB***	**3.96 bit**
		V_{ref}	**26.17 *dB***	**38.27 *dB***	**27.89 *dB***	**32.49 *dB***	**4.34 bit**
		WNN O/P	**56.34 *dB***	**78.5 *dB***	**60.67 *dB***	**61.35 *dB***	**9.79 bit**

Monte Carlo analysis is run to simulate the performance specifications for the three stages (V_{DM}, $V_{ref,}$ and V_{WNN}) of the clock-less-ADC system. The simulations are run over three different time resolutions, (*Tq* = 1μsec, 100 nSec, and 12.5 nSec) at variable frequencies (1 *Hz*, 100 *Hz*, 1 *KHz*, 4 *KHz*, and 20 *KHz*). All simulation results are listed in Table 5.2

The following is concluded from this analysis:

1) The delta modulation V_{DM} and Adaptive resolution V_{ref} systems have the same and exact performance, and achieve the same static resolution, ENOB, over different input frequencies except at very low input frequency (1 *Hz* to 3 *Hz*), as the input signal rises and falls very slowly causing a very low slope at the very low input frequencies, which creates few digitized samples, or tokens.

2) The WNN system has different performance and dynamic resolution over varying input frequencies. The achieved ENOB is inversely correlated with input frequency. Therefore, the lower frequencies achieve higher resolution. However, the higher frequencies achieve lower resolutions.

3) The time resolution is inversely correlated with the performance of WNN systems because a WNN system processes the wavelet coefficients (*C*) and the neural wavelons several times to achieve higher resolution. Therefore, the output signal of WNN system has a better resolution with reduced time resolution, as the WNN system interpolates the gap between the quantization levels

4) The time resolution has an almost constant correlation with the performance of delta modulation V_{DM} and adaptive resolution V_{ref} systems because they both depend on the static slope for input frequency.

5) A WNN system has a very high ENOB (38.4 bits) at 1 *Hz* input frequency because the WNN processes are running and analyzing only one sinusoidal signal. However, when the input frequency is 20 *KHz*, the ENOB achieved is 9.79 bits, because the WNN processes analyze and run throughout the 20 *KHz* frequencies. Thus, the error factor for *C* at 20 *KHz* is inflated, which impacts and reduces the number of resolutions bits.

In summary, The WNN system can enhance the signal resolution and improve its purity very well. The simulation tests prove that the WNN system has the ability to improve the clarity of the input signal running at a frequency range of 1 *Hz* to 20 *KHz* with a time resolution ranging from $0.125\ \mu s\ to\ 1\mu s$.

5.5 Performance Summary

In this thesis, we investigated the design of a clock-less-ADC using WNN and simulated their achieved high performance and high resolution. The following table presents the specifications and characteristics for the proposed design.

Table (5.1): Clock-less ADC performance summary

Parameter	This Proposition
Reconstruction	WNN in MATLAB
Amplitude Resolution	4 bits
Full-Scale Input (V_{pp})	1 volt
Adaptive Resolution	Yes
Automatic Calibration	Yes
Input Bandwidth	0.001 – 20 KHz
Frequency band (MHz)	1 - 80
Timer Resolution	$.0125\ \mu s - 1\mu s$
SQNR (*dB*)	56.34 *dB* @20KHz 67.29 *dB* @1Hz
SNR (*dB*)	78.5 *dB* @20KHz 232.89 *dB* @1Hz
SNDR (*dB*)	60.67 *dB* @20KHz 232.89 *dB* @1Hz
SFDR (*dB*)	61.35 *dB* @20KHz 253.89 *dB* @1Hz
ENOB	9.79 bit @20KHz 38.4 bit @1Hz
Clock Resolution	$10^{4.635}$ @20KHz $10^{12.355}$ @1Hz

Chapter 6

Conclusions and Suggestions for Future works

6.1 Summary and Contributions

In this thesis, we presented a new design for a clock-less CT-ADC with adaptive resolution and WNN as an interpolation technique to improve the precision of the clock-less ADC even further. The wavelet neural networks are a combination of wavelet analysis and neural network theory, which have the benefit of time-frequency localization of wavelet transform and the powerful learning function of neural networks. Therefore, WNN has optimized coverage of functions based on wavelet frame theory and the time-frequency localization of wavelet transform through suitably choosing the dilation and translation parameters and adjusting wavelet coefficients. Thus, wavelet analysis has proven to be a valuable tool for analyzing a wide range of time series and representing nonlinearities.

The new clock-less ADC is designed with 4 bit component circuits, a peak-to-peak voltage of 1 *volt*, 16 threshed levels, and a variable range input frequency 1 *Hz* ~ 20 *KHz* bandwidth simulated in MATLAB with different characteristics. The adaptive resolution is added as the first stage of signal resolution improvement, then WNN reconstructs the output signal to the final high-resolution output signal. This design achieves an ENOB of 9.79 *bits* to 38.4 *bits* for a sinusoidal input frequency ranging from 1 *Hz* to 20 *KHz*.

Resolution and performance for the new CT clock-less ADC using wavelet neural network are investigated. A 4 bit, 1 *Hz* ~ 20 *KHz* bandwidth design is simulated in MATLAB with various characteristics. In the following section, it will be outlined how this design compares to other level-crossing / CT clock-less previously published ACD designs.

6.2 Performance Comparison

We investigated the clock-less ADC using wavelet neural network and compared its performance with other previously published designs. The following table shows the performance results on several aspects.

Table 6.1: Comparison between several published researchers

Parameter	[7] 2008	[47] 2009	[10] 2011	[13] 2012	[49] 2013	[50] This Work
Amplitude Resolution	8 *bits*	4 *bits*	4 - 8 *bits*	4 - 8 *bits*	6 *bits*	4 *bits*
Timer Resolution	No timer	10*ns*	1*μs*	No timer	0.2 to 100*μs*	0.1 to 100*μs*
Adaptive Resolution	No	Yes	Yes	Yes	No	yes
Automatic Calibration	No	-	No	Yes	No	yes
Reconstruction and Test	DAC	Interpolation in MATLAB	6th order interpolation in MATLAB	Test DAC	3rd to 6th order interpolation in MATLAB	WNN in MATLAB
SNDR *(dB)*	*47 - 62*	*75*	*43.2 - 52.2*	*47 – 54*	49.4	60.67 – 232.9
Input Bandwidth *(KHz)*	*0.02 - 4*	*1000- 10,000*	*0.001 - 1*	*0.02 – 20*	0.005 - 5.1	0.001 - 20
Full-Scale Input V*pp*	*0.5*	-	*1.4*	*0.72*	2.25	1
Desired SNR (dB)	–	67	–	54-58	–	78.5- 232.9
ENOB	7.5 -10 *bits*	12.2 *bits*	6.9-8.4 *bits*	7.5-8.7 *bits*	7.9 *bits*	9.79 - 38.4 *bits*

As shown in the table, the design for a CT clock-less ADC published in 2008 used 8 bit comparators and achieved a resolution of 7.5-10 bits. The designs which were proposed after that focused on reducing the number of amplitude resolution (comparators) to reduce the hardware without significant efforts to enhance the signal resolution. Also, the authors used adaptive resolutions to track the input signal behavior and adjust the output resolution. Furthermore, they used interpolation to predicate the signal behavior between the crossing levels.

As in [47], the authors presented a new procedure by using variable signal resolution and reducing the captured non-uniformed data tokens to accelerate the encoding process and reduce the digitized data tokens stored in DSP. In this procedure, according to the behavior slope of the input signal, some recorded data tokens are eliminated if the signal slope has the same direction as that during the previous level crossing. As a result, the input signal is presented by fewer tokens. Thus, this technique is well suited for high-frequency applications. The main advantage of this technique is that decreasing the number of digitized data reduces the dynamic power dissipation by using lower delay cells. On the other hand, this system has several drawbacks as it employs some hardware components with the low-quality output signal. Although the signal changes the amplitude according to the slope of the input signal, the comparators (op-amps circuits), who create the delta level crossing, are still fabricated at the circuit level using the circuit space. As known, each comparator causes mismatch error, and leakage power dissipated even when the circuit is idle, Thus, reducing the number of fabricated comparators is the main goal for any clock-less ADC designs. Moreover, there is no control on the variable resolution of the output signal, which limits the application usage of this system due to low signal resolution.

In [10], the authors presented a system with an adaptive signal resolution to achieve better signal resolution. It increases the number of non-uniformed data space, according to the input signal behavior slope. This technique increases and changes the quantization resolution by using either 4 bit or 8-bit resolution depending on signal behavior. This system fixes some of the drawbacks of the previous design [47] by achieving better signal resolution tuned to the signal slope. This is approached using reliable detectors of the signal slope to approach better output signal. This will in return increase the applications that can deploy this design. However, this procedure was not able to solve some problem of the previous technique [47]. These include the fact that the hardware component is physically fabricated with all circuit components required for 8 bit resolution, although the signal behaviors don't require some of the cross levels

(Delta modulation). This causes unnecessary power dissipation and extra area dissipation, and still without a perfect output signal resolution.

In [13], the authors presented a second-generation circuit design that using an 8 bit circuit design to achieve clock-less ADC with the aliasing-free operation and higher SNDR and ENOB at a wide range of input frequencies. The design steps forward to make the input-activity-dependent clock-less ADC system more practical as such systems are still at the research stage. Although the proposed circuit achieved high SNDR and ENOB the system is still not competitive with any conventional system.

In [49], the authors innovated a new system and circuit design using a 6 bit circuit to use the clock-less ADC with low power dissipation and low frequency for biomedical applications. The design achieved decent improvement in ENOB for low input frequencies, but the system specifications are very limited to low frequency applications only with high input frequency amplitude.

In this work, a CT clock-less ADC using wavelet neural network to reconstruct the output signal and interpolate the signal behaviors between the level crossing is presented. A high signal resolution with more than 38 bits, by using 4 bit physical circuit components is achieved. The system can operate at very high speeds because of using Wavelet neural network and Gradient Descent Algorithm.

6.3 Conclusions

This thesis presents a clock-less ADC based on WNN to improve the output precision. It is primarily targeted for low power applications. Using a WNN technique for asynchronous ADC has resulted in higher SNDR, as there is no noise floor present in the output spectrum. The high-level simulation results indicate that an ADC employing WNN using a 4 bit system has achieved an ENOB for more than 38 bits. The clock-less ADC using WNN 4-bit system can be fabricated by 16 comparators. However, any conventional ADC with the same SNDR and the 38-bit system can be manufactured in

2.7488 E^{11} comparators, as computed in Table 4.1. Therefore, the presented system can be a prospective research area for energy and silicon area savings for high-quality signals in low power applications.

The prediction model of the proposed clock-less ADC based on WNN combines the advantages of both the wavelet and the traditional neural network and through the calculation function such as flexible and shifting to multi-scale analysis of the data. The results show that the wavelet neural network prediction model has the following advantages: Much better convergence and higher prediction precision, and stronger capability of study and popularization. So, it can provide the on-line prediction and has broad applications.

The wavelet neural networks in the proposed ADC succeeds in improving the signal quantization for clock-less ADC without fabricating any extra nonlinear hardware components, saving silicon area and power dissipation, and reducing the comparators' mismatch errors. Also, it achieves a higher signal resolution output with a few training iterations by using Gradient-Descent Algorithm; the wavelet neural network was interpolating the unknown signal behavior between the delta level crossing and presented the highest accurate output signal on a continuous time system. The characteristics of the proposed design in this work make the proposed clock-less ADC especially suitable for biomedical applications, the low-frequency application as well as high-frequency applications, and multiple voice applications. The achieved variable resolution depends on the input signal behavior, wavelet coefficients, and training process for neural networks. Therefore, the signal resolutions improvement is variable not fixed.

6.4 Suggestions for Future Work

In this section, we discuss some topics related to this work and which are worth further detailed studies. The clock-less ADC using WNN design is presented in this work with its full analysis and MATLAB simulation. However, for future work, it is anticipated to fabricate this design, compute the saved silicon area and measure performance metrics and power dissipation at various input frequencies.

Bibliography

[1] E. Allier, G. Sicard, L. Fesquet, and M. Renaudin, “Asynchronous level crossing analog to digital converters,” Measurement Journal, Elsevier Publisher, Vol. 37, pp.296-309, Apr. 2005.

[2] Yongjia Li, Wouter A. Serdijn “A Continuous-Time Level-Crossing ADC with 1-BitDAC and 3-Input Comparator” in Proc. IEEE Int. Symp. Circuits and Systems, Seoul, South Korea, 2012, pp. 1311–1314.

[3] E. Allier, G. Sicard, L. Fesquet, M. Renaudin “A New Class of Asynchronous A/D Converters Based on Time Quantization” in Proc. IEEE Int. Symp. Asynchronous Circuits and Systems, Vancouver, BC, Canada, May 2003, pp. 196–205.

[4] Y. W. Li, K. L. Shepard, and Yannis Tsividis, “A continuous-time programmable digital FIR filter,” IEEE J. Solid-State Circuits, vol. 41, no.11, pp. 2512–2520, Nov. 2006.

[5] Bob Schell “Continuous-Time Digital Signal Processors: Analysis and Implementation” degree of Doctor of Philosophy, Columbia University, 2008.

[6] Y. Tsividis, “Event-driven data acquisition and digital signal processing—A tutorial,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 8, pp. 577–581, Aug. 2010.

[7] Bob Schell and Yannis Tsividis, “A Continuous-Time ADC/DSP/DAC System With No Clock and With Activity-Dependent Power Dissipation.” IEEE J. Solid-State Circuits, vol. 43, no. 11, pp. 2472–2481, Nov.2008.

[8] M. Kurchuk and Y. Tsividis, “Signal-dependent variable-resolution clockless A/D conversion with application to continuous-time digital signal processing” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 5, pp. 982–991, May 2010.

[9] Xiaoyang Zhang, Yong Lian, “A 300-mV 220-nW Event-Driven ADC With Real-Time QRS Detection for Wearable ECG Sensors”, Dept. of Electr. & Comput. Eng. (ECE), Nat. Univ. of Singapore (NUS), Singapore, Singapore, March 2014.

[10] Michael Trakimas, Sameer R. Sonkusale, “An Adaptive Resolution Asynchronous ADC Architecture for Data Compression in Energy Constrained Sensing Applications” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 5, pp. 921–934, May 2011.

[11] A. Ratiu ; D. Morche ; Y. Tsividis ; S. Patil “A 3–10 fJ/conv-step Error-Shaping Alias-Free Continuous-Time ADC” Department of Electrical Engineering, Columbia University, New York, NY, USA. IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 51, NO. 4, APRIL 2016.

[12] M. Kurchuk, C. Weltin-Wu, D. Morche, and Y. Tsividis, “Event-driven GHz-range continuous-time digital signal processor with activity dependent power dissipation,” IEEE J. Solid-State Circuits, vol. 47, no. 9,pp. 2164–2173, Sep. 2012.

[13] C. Weltin-Wu and Y. Tsividis, " An event-driven, alias-free ADC with signal-dependent resolution," in Proc. IEEE Symp. VLSI Circuits, 2012, pp. 28-29.

[14] Z. Zhao and A. Prodic, “Continuous-time digital controller for high- frequency DC-DC converters,” IEEE Trans. Power Electron., vol. 23, no. 2, pp. 564–573, Mar. 2008.

[15] D. Bruckmann et al., “Optimization and implementation of continuous-time DSP systems by using granularity reduction,” in Proc. IEEE Int. Symp. Circuits. Syst. (ISCAS'11), pp. 410–413, May 2011.

[16] R.Kokozinski N.Tavangaran, D.Bruckmann and K.Konrad, "Continuous-time digital systems with asynchronous sigma delta modulation," in Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European, 2012, pp. 225–229.

[17] Y. Tsividis, "Event-driven, continuous-time ADCs and DSPs for adapting power dissipation to signal activity," in Proc. IEEE Int. Symp. Circuits Syst., Paris, France, May/Jun. 2010, pp. 3581–3584.

[18] M. Kurchuk, C. Weltin-Wu, D. Morche, and Y. Tsividis, "GHz-range programmable continuous time digital FIR with power dissipation that automatically adapts to signal activity," in Proc. IEEE Int. Solid-State Circuits Conf., Feb. 2011, pp. 232–233.

[19] A. N. Longhitano, F. del Cesta, P. Bruschi, R. Simmarano "A continuous time switched capacitor DAC with offset and flicker noise cancellation" 9 th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), Villach, Austria. June 2013, pp.197–200.

[20] Lokenath Debnath, Firdous Ahmad Shah "Wavelet Transforms and Their Applications" Springer, Second Edition, 2015.

[21] Antonios K. Alexandridis, Achilleas D. Zapranis. "Wavelet Neural Networks with applications in Financial Engineering, Chaos, and Classification" Wiley Publishing; 2014 edition.

[22] MATLAB Wavelet Toolbox.

[23] Alexander E. Hramov, Alexey A. Koronovskii, Valeri A. Makarov, Alexey N. Pavlov, Evgenia Sitnikova "Wavelets in Neuroscience" Springer-Verlag, New York (2015).

[24] Robert X Gao, Ruqiang Yan "Wavelets: Theory and Applications for Manufacturing" Springer, December 2010.

[25] Alexander E.Hramov, Alexey A.Koronovskii, Valeri A.Makarov, Alexey N.Pavlov, Evgenia Sitnikova “Wavelets in Neuroscience” ISSN 0172-7389, ISBN 978-3-662-43849-7, DOI 10.1007/978-3-662-43850-3, Springer August 5, 2014.

[26] Zhigang Zeng, Jun Wang "Advances in Neural Network Research and Applications." Springer; 2010 edition.

[27] Zhong Zhang; Dept. of Syst. Eng., Ind. Technol. Center of Okayama, Japan ; “Unsteady signal analysis by using wavelet transform” Intelligent Processing Systems, 1997. ICIPS '97. 1997 IEEE International Conference on (Volume: 2).

[28] Q. Zhang and A. Benveniste, “Approximation by Nonlinear Wavelet Networks” IEEE Trans. Neural Networks, vol. 3, no. 6, pp. 889-898, Nov. 1992.

[29] Xu Yang, Hiroyuki. Kumehara, Wei Zhang “Back Propagation Wavelet Neural Network Based Prediction of Drill Wear from Thrust Force and Cutting Torque Signals” Vol. 2, No. 3 Computer and Information Science, August 2009..

[30] Da-Ke Chen, Jiu-Qiang Han, “Approaches To Realize High Precision Analog-To-Digital Converter Based On Wavelet Neural Network” International Conference on Wavelet Analysis and Pattern Recognition, Beijing, China, 2-4 Nov. 2007.

[31] Dake Chen, Jiuqiang Han, “Application of wavelet neural network in signal processing of MEMS accelerometers.” Microsystem Technologies (Impact Factor: 0.95). 01/2011; 17(1):1-5. DOI: 10.1007/s00542-010-1169-7, November 2010..

[32] Harold H.Szu, B. Telfer, and S. Kadambe, “Neural network Adaptive Wavelets for Signal Representation and Classification,” Optical Engineering. Vol.32, No.9, pp. 1907-1916,1992.

[33] Venkata Narasimha Manyam, Dhurv Chhetri, and J Jacob Wikner “Clockless Asynchronous Delta Modulator Based ADC for Smart Dust Applications” 2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip.

[34] E. Allier, G. Sicard, L. Fesquet, and M. Renaudin, "A new class of asynchronous A/D converters based on time quantization," in Proc. Ninth IEEE Int. Symp. on Asynchronous Circuits and Systems, Vancouver, May 2003, pp. 196-205.

[35] Y. Li, K. Shepard, and Y. Tsividis, "A continuous-time programmable digital FIR filter," in Proc. 11th IEEE Int. Symp. On Asynchronous Circuits and Systems, Vancouver, Mar. 2005, pp. 138-143.

[36] E. Allier, J. Goulier, G. Sicard, A. Dezzani, and M. Renaudin, "A 120 nm low power asynchronous ADC," in Proceedings oflSLPED, Aug. 2005, pp. 60-65.

[37] Y. Li, K. Shepard, and Y. Tsividis, "A continuous-time programmable digital FIR filter," IEEE J. Solid-State Circuits, vol. 41, no. 11, pp. 2512-2520, Nov. 2006.

[38] E. Allier, G. Sicard, L. Fesquet, M. Renaudin “A New Class of Asynchronous A/D Converters Based on Time Quantization” Proc. Intl. Symposium on Asynchronous Circuits and Systems, pp. 196- 205, May 2003.

[39] L. Fesquet, G. Sicard, and B. Bidegaray-Fesquet, “Targeting ultra-low power consumption with non-uniform sampling and filtering” in IEEE International Symposium on Circuits and Systems, ISCAS2010. Paris, France: IEEE, May 2010, pp. 3585–3588.

[40] 2010 CMOS Emerging Technologies Conference Presentation Slides Volume 4, Circuits Track, May 19-21, 2010, Whistler, BC, Canada.

[41] David Money Harris, Sarah L. Harris “Digital Design and Computer Architecture” 2007: Morgan Kaufmann.. [42] Silva, Paulo, Huijsing, Johan “High-Resolution IF-to-Baseband Sigma Delta ADC for Car Radios,” Springer Netherlands, 2008.

[43] David R. Martinez, Robert A. Bond, M. Michael Vai “High-Performance Embedded Computing Han*dB*ook: A Systems Perspective,” CRC Press; 2008 edition.

[44] Dominique Dallet “Dynamic Characterisation Of Analogue-To-Digital Converters,” Springer-Verlag New York Inc.; 2010 edition.

[45] Lavrador, Pedro Miguel Telecommun. Inst., Univ. of Aveiro, Portugal de Carvalho, N.B.; Pedro, Jose Carlos (March 2004). "Evaluation of signal-to-noise and distortion ratio degradation in nonlinear systems." IEEE Microwave Theory and Techniques Society 52 (3): 813 – 822. Doi: 10.1109/TMTT.2004.823543. ISSN 0018-9480.

[46] Walt Kester, MT-003 “Understand SINAD, ENOB, SNR, THD, THD + N, and SFDR, so You Don't Get Lost in the Noise Floor,” Tutorial, Analog Devices, MT-003, January 2009.

[47] K. Kozmin, J. Johansson, and J. Delsing, “Level-crossing ADC performance evaluation toward ultrasound application” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 56, no. 8, pp. 1708-1719, Aug. 2009.

[48] Weidong Kuang “Correction and Comment on (An Adaptive Resolution Asynchronous ADC Architecture for Data Compression in Energy Constrained Sensing Applications) ” IEEE Transactions On Circuits And Systems—I: Regular Papers, Vol. 60, No. 4, April 2013.

[49] Yongjia Li, Duan Zhao and Wouter A. Serdijn “A 0.8V 8-Bit Low-Power Asynchronous Level- Crossing ADC with Programmable Comparison Windows”, in Proc. IEEE Biomed. Circuits and Systems Conf., 2013, pp. 138–141.

[50] Tamer Elsalahati, Ezz El-Masry and Dalia A. El-Dib “High Precision Clockless ADC using wavelet neural network” IEEE, Microelectronics (ICM), 2016 28th International Conference, December 2016.

Appendix A: MATLAB Code

```
%  |-------------------------------------|   %#codegen
%  |    Continous Time ADC using         |
%  |    Wavelet Neural Network (WNN)     |
%  |  Copyright by Tamer Elsalahati (R)  |
%  |-------------------------------------|
close all;clear; clc;

% Creat I/P Signal "SinWave"
%------------------------------
Frequency=1;                    % Input Frequency
resolution=1e-06;               % Sample time (1 us) / Signal Resolution
Fs=1/resolution;                % Sampling frequency
L = Fs;                         % Length of signal
t =(0:1:L)* resolution;         % Time vector
amp=0.5;                        % Input Signal Amplitude
x=amp*sin(2*pi*Frequency*t);    % Sinusoid input singal (Hz)
km= length(x);                  % Length of signal

% Delta Modulation ADC (CT-ADC)
% -------------------------------
comparator=16;                   % Numbers of Comparators - 4 bit design
required (16) Comparators
Delta=1/comparator;              % Quantization Step
bit=8;                           % Numbers of encoding Binrary bits
shift=4;                         % Combination Binary Shifting binary

%  Create  (Amplitude Shifting (offset) Section(4.2)
%  ------------------------------------------------------
x1=x-Delta/4;                   % Shifted Input signal Vi/p (Delta/4)
x2=x-Delta/2;                   % Shifted Input signal Vi/p (Delta/2)
x3=x-((Delta*3)/4);             % Shifted Input signal Vi/p (3/4 Delta)

for k=1:1:km
    if x1(k)<=-amp
       x1(k)=-amp;
    end
    if x2(k)<=-amp
       x2(k)=-amp;
    end
    if x3(k)<=-amp
       x3(k)=-amp;
    end
end
```

Create Delta Moduation (Quantization Levels) ----------

```
Quantization_Levels=double(-(amp):Delta:(amp));  % Quantization Levels
Number_of_Levels=length(Quantization_Levels);    % Compute Number of
Quantization Levels

% Create the Continuous Time system for I/P Signal (x)
% ----------------------------------------------------

%  UP / Down Signal / Signal Behaviour (X)
% ----------------------------------------
beh=zeros(1,km,'double');
UPDN=zeros(1,km,'double');
UPDN(km-1:km)=1;

for k=1:1:km-2
     if x(k+2)>=x(k)
         UPDN(k)=1;
     end
     if (x(k) >= 0)
        beh(k)=1;
     else
        beh(k)=0;
     end
 end

% Create Change Signal for input Signal (X)
% ----------------------------------------
Change= zeros(1,km,'double');
jo=(comparator/2)+1;  j=jo;

for k=2:1:km-1
    if (x(k)>=Quantization_Levels(j) && UPDN(k)==1)
        Change(k)=1;
        j=j+1;
    elseif (x(k)< Quantization_Levels(j) && UPDN(k)==0)
        Change(k)=1;
        j=j-1;
    end
end

Change(1)=0; Change_DM=Change;

%  Create Change Signal for input Signal (X1)
%-------------------------------------------
Change1= zeros(1,km,'double');
j=jo;
for k=2:1:km-1
    if (x1(k)>=Quantization_Levels(j) && UPDN(k)==1)
        Change1(k)=1;
```

```
        j=j+1;
    elseif (x1(k)< Quantization_Levels(j) && UPDN(k)==0)
        Change1(k)=1;
        j=j-1;
    end
end

 for k=1:1:km
     if Change1(k)==1
         Change(k)=1;
     end
 end

%  Create Change Signal for input Signal (X2)
%-----------------------------------------------
Change1= zeros(1,km,'double');
j=jo;

for k=2:1:km-1
    if (x2(k)>=Quantization_Levels(j) && UPDN(k)==1)
        Change1(k)=1;
        j=j+1;
    elseif (x2(k)< Quantization_Levels(j) && UPDN(k)==0)
        Change1(k)=1;
        j=j-1;
    end
end

 for k=1:1:km
     if Change1(k)==1
         Change(k)=1;
     end
 end

%  Create Change Signal for input Signal (X3)
%-----------------------------------------------
Change1= zeros(1,km,'double');
j=jo;

for k=2:1:km-1
    if (x3(k)>=Quantization_Levels(j) && UPDN(k)==1)
        Change1(k)=1;
        j=j+1;
    elseif (x3(k)< Quantization_Levels(j) && UPDN(k)==0)
        Change1(k)=1;
        j=j-1;
    end
end

 for k=1:1:km
```

```
    if Change1(k)==1
        Change(k)=1;
    end
end

for k=2:1:km-1
    if (UPDN(k-1)==1 && UPDN(k)==0)
        Change(k)=0;
        Change(k+1)=0;
    end
    if (UPDN(k-1)==0 && UPDN(k)==1)
        Change(k)=0;
        Change(k+1)=0;
    end
 end
```

Create the DAC (CT-DAC) % (Section4.9) ----------

```
DAC= zeros(1,km,'double');
for k=2:1:km-1
    if UPDN(k)==1
        if beh(k)==1
            if Change(k)==1
                DAC(k)=DAC(k-1)+Delta/4;
            else
                DAC(k)=DAC(k-1);
            end
        elseif beh(k)==0
            if Change(k)==1
                if Change(k-1)==1
                    DAC(k)=DAC(k-1);
                else
                    DAC(k)=DAC(k-1)+Delta/4;
                end
            else
                DAC(k)=DAC(k-1);
            end
            if ( Change(k)==1 &&  Change(k+1)==0 &&  Change(k-1)==1 ) %--------
------
                     DAC(k)=DAC(k-1)+Delta/4;
            end
        end
    elseif UPDN(k)==0
        if beh(k)==1
            if Change(k)==1
                if ( Change(k+1)==1 && Change(k-1)==1 )
                    DAC(k)=DAC(k-1);
                else
                    DAC(k)=DAC(k-1)-Delta/4;
                end
```

```
            else
                DAC(k)=DAC(k-1);
            end
        elseif beh(k)==0
            if (Change(k)==1 && Change(k-1)==0)
                DAC(k)=DAC(k-1)-Delta/4;
            else
                DAC(k)=DAC(k-1);
            end
            if (Change(k-1)==0 && Change(k)==1 && Change(k+1)==1 )
                DAC(k)=DAC(k-1)-Delta/4;
            end
        end
    end
    if DAC(k)>=Quantization_Levels(Number_of_Levels)
        DAC(k)=Quantization_Levels(Number_of_Levels);
    end
    if DAC(k)<=-amp
        DAC(k)=-amp;
    end
end
DAC(1)=0;

% Bell Shape Filtering
% --------------------
for k=1:1:km
    if DAC(k)>=Quantization_Levels(Number_of_Levels)
        DAC(k)=Quantization_Levels(Number_of_Levels);
    end
    if DAC(k)<=Quantization_Levels(1)+Delta/2
        DAC(k)=-amp;
    end
end
%  Compute the Quantization Error of (x)                  % Section(4.10)
%  -------------------------------------
Vref_error=x-DAC;        % Quantization Error
Vref=DAC;
```

Amplified the quantizing error (e) % Section(4.12) ----------

```
% Filter the Amplified quantizied error
% -------------------------------------
error_amp=Vref_error;
 for k=1:1:km
    if DAC(1,k)<=-amp+Delta
        error_amp(1,k)=-Delta/4;
    end
 end
error_amp=4.*error_amp.*(comparator-1)+amp;
```

Convert the Amplified quantizing error to Digital Signal Using Continuous system ----------
---

```
jo=1;  Error_Digitlization=error_amp;

beh_Error=zeros(1,km);



% beh_Error Signal
%------------------
for p = 1:1:km % for all the signal Value
    if (Error_Digitlization(p) >= 0)
        beh_Error(p)=1;
    else
        beh_Error(p)=0;
    end
 end

% UP Signal
%-----------
UP=zeros(1,km);

for k=1:1:km-1
    if Error_Digitlization(k)>=Error_Digitlization(k+1)
        UP(k)=1;
   else
        UP(k)=0;
    end
end
UP(km)=UP(km-1);

% Down Signal
%-------------
DN_Error=zeros(1,km);
for k=2:km
    if Error_Digitlization(k)>=Error_Digitlization(k-1)
        DN_Error(k)=1;
   else
        DN_Error(k)=0;
    end
end
DN_Error(1)=DN_Error(2);

 %  |---------------------------------------------|
 %  |  Create the Error_Change from  Error Signal  |
 %  |---------------------------------------------|

% Create Error_Change Signal (+ve Cycle)
```

```
%---------------------------------------
Error_Change=zeros(1,km);
  for k=1:1:km
          Error_Change(k)=0;
  end

for k=1:1:km
    if (Error_Digitlization(k)>=Quantization_Levels(j) && UPDN(k)==1 &&
DN_Error(k)==1)
        Error_Change(k)=1;
        j=j+1;
        if j>=Number_of_Levels
            j=Number_of_Levels;
        end
    end
    if (DN_Error(k)==0 && UPDN(k)==1)
        j=jo;
    end
end

% Create Error_Change Signal (-ve Cycle)
%-----------------------------------
ju=Number_of_Levels;
 for ju=Number_of_Levels:-1:jo
     for k=2:1:km-1
         k=2;
         if (Error_Digitlization(k)<=Quantization_Levels(ju) && UPDN(k)==0 &&
DN_Error(k)==0)
             Error_Change(k)=1;
             k=k+1;
             ju=ju-1;
             if ju<jo
                 ju=jo;
             end
         end
         if (DN_Error(k)==0 && DN_Error(k+1)==1 && ju==jo && UPDN(k)==0)
             ju=Number_of_Levels;
         end
         if (DN_Error(k)==0 && DN_Error(k-1)==1 && UPDN(k)==0)
             ju=Number_of_Levels;
         end
     end
     ju=ju-1;
 end

 for k=1:1:km-1
     if (Error_Digitlization(k)<(Quantization_Levels(jo)-Delta/bit) &&
UPDN(k)==0 && DN_Error(k)==0)
       Error_Change(k)=0;
     end
```

```
 end

km= length(Error_Digitlization);
for k=1:1:km
    if (Error_Digitlization(k)>=(Quantization_Levels(Number_of_Levels)-
(Delta/2)) && DN_Error(k)==0 && UPDN(k)==0);
        Error_Change(k)=0;
    end
end

 %      |-------------------------------------------------|
 %      |       Recover the DAC Signal of Error Signal      |
 %      |-------------------------------------------------|

DAC_Error=zeros(1,km);
for k=2:1:km
        % (+ve) Cycle recovery
        %-------------------------------------------------
    if (Error_Change(k)==1 && DN_Error(k)==1 && UPDN(k)==1)
        DAC_Error(k)=DAC_Error(k-1)+Delta;
    end
    if (DN_Error(k)==0 && UPDN(k)==1)
        DAC_Error(k)=Quantization_Levels(jo);
    end
    if (Error_Change(k)==0 && DN_Error(k)==1 && UPDN(k)==1)
        DAC_Error(k)=DAC_Error(k-1);
    end
%         % (-ve) Cycle recovery
%         %-------------------------------------------------
    if (UPDN(k-1)==1 && UPDN(k)==0)
        DAC_Error(k)= Quantization_Levels(Number_of_Levels);
    end
    if (Error_Change(k)==1 && DN_Error(k)==0 && UPDN(k)==0)
        DAC_Error(k)=DAC_Error(k-1)-Delta;
    end
    if (DN_Error(k)==1 && UPDN(k)==0)
        DAC_Error(k)=Quantization_Levels(Number_of_Levels);
    end
    if (Error_Change(k)==0 && DN_Error(k)==0 && UPDN(k)==0)
        DAC_Error(k)=DAC_Error(k-1);
    end
    if DAC_Error(k)<=Quantization_Levels(jo)
        DAC_Error(k)=Quantization_Levels(jo);
    end
end
```

```
          // ----------------- Convert Decimal - to _ Binary-------------------
--- //

 |-------------------------------|
```

```
|   Convert Vref to Binary      |                    % Section(4.13)
|-------------------------------|
```

```
% Prelocation Of Array
% ---------------------
b2=zeros(1,km,'double');
b3=zeros(1,km,'double');    b3_x=zeros(1,km,'double');
b4=zeros(1,km,'double');    b4_x=zeros(1,km,'double');
b5=zeros(1,km,'double');    b5_x=zeros(1,km,'double');
b6=zeros(1,km,'double');    b6_x=zeros(1,km,'double');
b7=zeros(1,km,'double');    b7_x=zeros(1,km,'double');
bit1=zeros(1,km,'double');  bit1_e=zeros(1,km,'double');
bit2=zeros(1,km,'double');  bit2_e=zeros(1,km,'double');
bit3=zeros(1,km,'double');  bit3_e=zeros(1,km,'double');
bit4=zeros(1,km,'double');  bit4_e=zeros(1,km,'double');
bit5=zeros(1,km,'double');  bit5_e=zeros(1,km,'double');
bit6=zeros(1,km,'double');  bit6_e=zeros(1,km,'double');
bit7=zeros(1,km,'double');
in_1=zeros(1,km,'double');  in_2=zeros(1,km,'double');
Sign=zeros(1,km,'double');
Sign_signal=zeros(1,km,'double');
stepp=zeros(1,km,'double');

in=Vref;

for k=1:1:km
    in_1(k)=abs_func(in(1,k));
end

% Convert DAC to Binary
% ---------------------
for k=1:1:km
    Sign(k)=sign(in(k));
end
for k=1:1:km
      if sign(in(k))==1
        bit1(k)=0;
      elseif sign(in(k))==-1
        bit1(k)=1;
      elseif sign(in(k))==0
        bit1(k)=0;
      end
end

for k=1:1:km
        b2(k)=double(in_1(k))/0.5;     % bit 2
        if b2(k)>=1
            bit2(k)=1;
        else bit2(k)=0;
```

```
        end
        if b2(k)>=1
        b3_x(k)= b2(k)-1;       % bit 3
        else b3_x(k)=b2(k);
        end
        b3(k)=b3_x(k)/0.5;
         if b3(k)>=1
            bit3(k)=1;
         else bit3(k)=0;
         end

        if b3(k)>=1        % bit 4
        b4_x(k)= b3(k)-1;
        else b4_x(k)=b3(k);
        end
        b4(k)=b4_x(k)/0.5;
         if b4(k)>=1
            bit4(k)=1;
         else bit4(k)=0;
         end

        if b4(k)>=1        % bit 5
        b5_x(k)= b4(k)-1;
        else b5_x(k)=b4(k);
        end
        b5(k)=b5_x(k)/0.5;
         if b5(k)>=1
            bit5(k)=1;
         else bit5(k)=0;
         end

         if b5(k)>=1        % bit 6
        b6_x(k)= b5(k)-1;
        else b6_x(k)=b5(k);
        end
        b6(k)=b6_x(k)/0.5;
         if b6(k)>=1
            bit6(k)=1;
         else bit6(k)=0;
         end

        if b6(k)>=1               % bit 7
        b7_x(k)= b6(k)-1;
        else b7_x(k)=b6(k);
        end
        b7(k)=b7_x(k)/0.5;
         if b7(k)>=1
            bit7(k)=1;
         else bit7(k)=0;
         end
```

```
        if b7(k)>=1              % bit 8
        b8_x(k)= b7(k)-1;
        else b8_x(k)=b7(k);
        end
        b8(k)=b8_x(k)/0.5;
         if b8(k)>=1
            bit8(k)=1;
         else bit8(k)=0;
         end
end

% Convert Quantization error to and Binary
% ----------------------------------------
inn=transpose(Error_Digitlization);
for k=1:1:km
     in_2(k)=abs_func(inn(k));
end

for k=1:1:km
        b2(k)=double(in_2(k))/0.5;     % bit 2
        if b2(k)>=1;
            bit2_e(k)=1;
        else bit2_e(k)=0;
        end
        if b2(k)>=1;
        b3_x(k)= b2(k)-1;       % bit 3
        else b3_x(k)=b2(k);
        end
        b3(k)=b3_x(k)/0.5;
         if b3(k)>=1;
            bit3_e(k)=1;
         else bit3_e(k)=0;
         end
        if b3(k)>=1;        % bit 4
        b4_x(k)= b3(k)-1;
        else b4_x(k)=b3(k);
        end
        b4(k)=b4_x(k)/0.5;
         if b4(k)>=1;
            bit4_e(k)=1;
         else bit4_e(k)=0;
         end
        if b4(k)>=1;        % bit 5
        b5_x(k)= b4(k)-1;
        else b5_x(k)=b4(k);
        end
        b5(k)=b5_x(k)/0.5;
         if b5(k)>=1;
            bit5_e(k)=1;
         else bit5_e(k)=0;
```

```
         end

         if b5(k)>=1;        % bit 6
        b6_x(k)= b5(k)-1;
        else b6_x(k)=b5(k);
        end
        b6(k)=b6_x(k)/0.5;
         if b6(k)>=1;
            bit6_e(k)=1;
         else bit6_e(k)=0;
         end

        if b6(k)>=1              % bit 7
        b7_x(k)= b6(k)-1;
        else b7_x(k)=b6(k);
        end
        b7(k)=b7_x(k)/0.5;
         if b7(k)>=1
            bit7_e(k)=1;
         else bit7_e(k)=0;
         end
        if b7(k)>=1              % bit 8
        b8_x(k)= b7(k)-1;
        else b8_x(k)=b7(k);
        end
        b8(k)=b8_x(k)/0.5;
         if b8(k)>=1
            bit8_e(k)=1;
         else bit8_e(k)=0;
         end
end
```

Combine DAC and Amplified Quantization Error % Section(4.14) ---------- ----------| Solution 8 (first 6 bits of DAC + first 2-6 bits of Error) | | No Sign bit of Error | ----------

```
Decimal_Combined=zeros(1,km,'double');
for k=1:1:km
    if bit1(1,k)==1
         Sign_signal(1,k)=-1;
    else Sign_signal(1,k)=1;
    end
Decimal_Combined(1,k)= bit2(1,k)*power_func(2,-1);
Decimal_Combined(1,k)= Decimal_Combined(1,k)+bit3(1,k)*power_func(2,-2);
Decimal_Combined(1,k)= Decimal_Combined(1,k)+bit4(1,k)*power_func(2,-3);
Decimal_Combined(1,k)= Decimal_Combined(1,k)+bit5(1,k)*power_func(2,-4);
Decimal_Combined(1,k)= Decimal_Combined(1,k)+bit6(1,k)*power_func(2,-5);
Decimal_Combined(1,k)= Decimal_Combined(1,k)+bit7(1,k)*power_func(2,-6);
Decimal_Combined(1,k)= Decimal_Combined(1,k)+bit8(1,k)*power_func(2,-7);
```

```
Decimal_Combined(1,k)= Decimal_Combined(1,k)+bit2_e(1,k)*power_func(2,-8);
Decimal_Combined(1,k)= Decimal_Combined(1,k)+bit3_e(1,k)*power_func(2,-9);
Decimal_Combined(1,k)= Decimal_Combined(1,k)+bit4_e(1,k)*power_func(2,-10);
Decimal_Combined(1,k)= Decimal_Combined(1,k)+bit5_e(1,k)*power_func(2,-11);

Decimal_Combined(1,k)=Decimal_Combined(1,k)*Sign_signal(1,k);
    if Decimal_Combined(1,k)>=amp
    Decimal_Combined(1,k)=amp;
    end

    if Decimal_Combined(1,k)<=0
    Decimal_Combined(1,k)=Decimal_Combined(1,k)- Delta/8;
    end
end

%  Filtering the Combined signal (Vcombined)
%  -----------------------------------------

% Get the Delta (t) from DAC
%--------------------------------
for k=2:1:km
    if DAC(k)==DAC(k-1)
        stepp(k)=0;
    else
        stepp(k)=1;
    end
end

% Divide the Tq to two periods for Filtering
%--------------------------------------------
tol=find(stepp);  tok=length(tol);
toll=zeros(1,(2*tok)+1,'double');   fd=zeros(1,km,'double');

for ux=1:1:tok
    [toll(2*ux)]=tol(ux);
end

toll(1)=toll(2)/2;

for ux=3:2:2*tok-1
    [toll(ux)]=(toll(ux+1)+toll(ux-1))/2;
    %ux=ux+1;
end

toll2=round_func(toll);
Decimal_Combined_tam=DAC;
toll2(2*tok+1)=0;

for nj=1:2:2*tok
```

```
    if (UPDN(toll2(nj))==1 && beh(toll2(nj))==1); % (1)  if the Signal Ramp up
and Positive Cycle
        for tp=toll2(nj):1:toll2(nj+1);
            Decimal_Combined_tam(tp)=Decimal_Combined(tp);
        end
        for tp=toll2(nj+1):1:toll2(nj+2);
            Decimal_Combined_tam(tp)=DAC(tp);
        end
    end
end

for nj=1:2:2*tok
    if (UPDN(toll2(nj))==0 && beh(toll2(nj))==1); % (2)  if the Signal Ramp
down and Positive Cycle
        for tp=toll2(nj):1:toll2(nj+1);
            Decimal_Combined_tam(tp)=DAC(tp);
        end
        for tp=toll2(nj+1):1:toll2(nj+2);
        Decimal_Combined_tam(tp)=Decimal_Combined(tp);
        end
    end
 end

 for nj=1:2:2*tok
     if (UPDN(toll2(nj))==0 && beh(toll2(nj))==0); % (3)  if the Signal Ramp
Down and Negative Cycle
         for tp=toll2(nj):1:toll2(nj+1);
             Decimal_Combined_tam(tp)=Decimal_Combined(tp);
         end
         current_Decimal_Combined=DAC(toll2(nj-1))-Delta/4;
         for tp=toll2(nj+1):1:toll2(nj+2)
             Decimal_Combined_tam(tp)=current_Decimal_Combined;  % ana 3awaz el
DAC ya7'od el cycle ely 2ablaaha
             fd(tp)=Decimal_Combined_tam(tp);
         end
     end
 end

for nj=1:2:2*tok
    if (UPDN(toll2(nj))==1 && beh(toll2(nj))==0)            % (4)  if the
Signal Ramp up and Negative Cycle
        Current_Decimal_Combined_tam=DAC(toll2(nj-1));
        for tp=toll2(nj):1:toll2(nj+1);
            Decimal_Combined_tam(tp)=Current_Decimal_Combined_tam;
            fd(tp)=Decimal_Combined_tam(tp);
        end
        for tp=toll2(nj+1):1:toll2(nj+2);
            Decimal_Combined_tam(tp)=Decimal_Combined(tp);
        end
    end
```

```
end

% Filter Bell shap at Negative Cycle
% ------------------------------------
for k=2:1:km;
    if (Decimal_Combined_tam(k)<=(-amp+Delta))
        Decimal_Combined_tam(k)=DAC(k);
    end
end

% Filter Bell shap at Positive Cycle
% ---------------------------------------
for nj=2:2*tok
    if (Decimal_Combined_tam(toll2(nj))>=(amp-Delta) && UPDN(toll2(nj))==0 &&
UPDN(toll2(nj-1))==1 && beh(toll2(nj))==1); % (1)  if the Signal Ramp up and
Positive Cycle
        for tp=round_func((toll2(nj)+toll2(nj-
1))/2):1:round_func((toll2(nj)+toll2(nj+1))/2);
            Decimal_Combined_tam(tp)=Decimal_Combined(tp);
        end
    end
end

for nj=2:2*tok         % This is only for the Negative Cycle
    if (Decimal_Combined_tam(toll2(nj))<=(-amp+3*Delta/4) && UPDN(toll2(nj))==1
&& UPDN(toll2(nj-1))==0 ) % (1)  if the Signal Ramp down and Negtive Cycle
        for tp=round_func((toll2(nj)+toll2(nj-
1))/2):1:round_func((toll2(nj)+toll2(nj+1))/2);
            Decimal_Combined_tam(tp)=Decimal_Combined(tp);
        end
    end
end

%  Filter Positive-to-Negative or Negative-to-Positive Cycle
%  ---------------------------------------------------------
for nj=2:2*tok    % for Zero negative cycle
    if (UPDN(toll2(nj))==0 && beh(toll2(nj-1))==1 && beh(toll2(nj))==0);  % fow
Zero
        for tp=toll2(nj+1):1:toll2(nj+2);
            Decimal_Combined_tam(tp)=DAC(tp);
            ttt=DAC(tp);
        end
    end
end

for nj=2:2*tok    % for Zero Positive cycle
    if (UPDN(toll2(nj))==1 && beh(toll2(nj-1))==0 && beh(toll2(nj))==1);  % fow
Zero
        for tp=toll2(nj-2):1:toll2(nj-1);
            Decimal_Combined_tam(tp)=DAC(tp);
```

```
        end
        for tp=toll2(nj-1):1:toll2(nj);
            Decimal_Combined_tam(tp)=Decimal_Combined(tp);
        end
        for tp=toll2(nj):1:toll2(nj+1);
            Decimal_Combined_tam(tp)=DAC(tp);
        end
    end
end

V_combined=Decimal_Combined_tam-Delta/16;
EEE=x-V_combined;
```

Wavelet Neural Network % Chapter(3) ----------

```
d=0.8*x;

d=d+amp;

Gx0=0.8*V_combined;

Gx0=Gx0+amp;



% WNN Initial Values
% --------------------
N=20;                                % Number of hidden layers
i=10;                                % learning iterations
Ny=31.3;                             % learning rate.
B= 1.2700e-08;                       % Momentum Factor to the weights.
ai=zeros(N,km+1,i);                  % dilation
bi=zeros(N,km+1,i);                  % translation
wi=zeros(N,km+1,i);                  % Weights

% Prelocation Of Array
% ----------------------
taw=zeros(1,km);    Gx1=zeros(1,km);            Gx2=zeros(1,km);
Gx=zeros(1,km);     error_WNN=zeros(1,km);      Gx_dash=zeros(1,km);
De_Dw=zeros(1,km);  De_Db=zeros(1,km);          De_Da=zeros(1,km);
Delta_wi=zeros(1,km+1,i); Delta_bi=zeros(1,km+1,i); Delta_ai=zeros(1,km+1,i);
SNDR_Gx= zeros(1,i) ;  ENOB_X_Gx_new= zeros(1,i);  Signal_corr= zeros(1,i);
Delta_wi=zeros(1,km+1,i); Delta_bi=zeros(1,km+1,i); Delta_ai=zeros(1,km+1,i);

%       ------------------| Calculate the WNN again|-------------
for kf=1:1:i
    for kr=1:1:N
        for k=1:1:km
            ai(kr,k,kf)=40458*kr*0.1;                   % dilation
```

```
            bi(kr,k,kf)=53.4600*kr*0.1;                 % translation
            wi(kr,k,kf)=1.9440*kr*0.1;                  % Weights
        end
    end
end

% Prelocation Of Array
% ---------------------

% Wavelet Morlet Equation
%----------------------------
for kf=1:1:i
    for kr=1:1:N
        for k=1:1:km
            taw(kr,k)= (Gx0(k)-bi(kr,k,kf))/ai(kr,k,kf);                    %
Equation (3.22)
            Gx1_N(kr,k)=cos(1.75*taw(kr,k)).* exp(-(t(k).*t(k))/2);         %
Equation (3.17)
            Gx2_N(kr,k)=wi(kr,k,kf)*Gx1_N(kr,k);
            Gx_N(kr,k,kf)=1/(1+exp(-Gx2_N(kr,k)));                          %
Equation (3.18)
            error_WNN_N(kr,k)= 0.5.*((d(k)-Gx_N(kr,k))^2);                  %
Equation (3.21)

%Gradient-Descent Algorithm
%----------------------------
            Gx_dash(kr,k)= Gx_N(kr,k).*(1-Gx_N(kr,k));
            De_Dw(kr,k)=-(d(k)-Gx_N(kr,k)).*
Gx_dash(kr,k).*cos(1.75.*taw(kr,k)).*exp(-0.5.*(taw(kr,k).*taw(kr,k)));    %
Equation (3.23)
            De_Db(kr,k)=-(d(k)-Gx_N(kr,k)).*
Gx_dash(kr,k).*(wi(kr,k,kf)/ai(kr,k,kf)).*exp(-
0.5.*(taw(kr,k).*taw(kr,k))).*(1.75.*(sin(1.75.*taw(kr,k)))+
taw(kr,k).*(cos(1.75.*taw(kr,k))));   % Equation (3.24)
            De_Da(kr,k)=(taw(kr,k).*De_Db(k));                              %
Equation (3.25)
            Delta_wi(kr,k+1,kf)=-kf.*Ny.*De_Dw(kr,k)+ B.*wi(kr,k,kf);       %
Equation (3.30)
            Delta_bi(kr,k+1,kf)=-kf.*Ny.*De_Db(kr,k)+ B.*bi(kr,k,kf);       %
Equation (3.31)
            Delta_ai(kr,k+1,kf)=-kf.*Ny.*De_Da(kr,k)+ B.*ai(kr,k,kf);       %
Equation (3.32)
            wi(kr,k+1,kf)=wi(kr,k,kf)+Delta_wi(kr,k+1,kf);                  %
Equation (3.33)
            bi(kr,k+1,kf)=bi(kr,k,kf)+Delta_bi(kr,k+1,kf);                  %
Equation (3.34)
            ai(kr,k+1,kf)=ai(kr,k,kf)+Delta_ai(kr,k+1,kf);                  %
Equation (3.35)
        end
    end
```

```
end

% The Highest ENOB
% -----------------
for kf=1:1:i
for kr=1:1:N
     ENOB_WNN_best(kr,kf)=(sinad(1.25*(Gx_N(kr,1:km,kf)-amp))-1.76)/6.02;
end
end

Max_ENOB=max(ENOB_WNN_best(kr,kf));
[Best_N,Best_iteration]=find(ENOB_WNN_best(1:kr,kf)==Max_ENOB);
Gx=Gx_N(Best_N,1:km,Best_iteration);

EE=d-Gx; % to plot the Error_WNN
Gx=Gx-amp;
Gx=1.25*Gx;
 %-------------------------------------------------------------%
 %                     Performance Metrics                     %
 %-------------------------------------------------------------%

% Calculate the Delta Modulation (ADC/DAC)
% ----------------------------------------

% Change Signal for Delta Modulation
% ----------------------------------
  for k=2:1:km-1
        if (UPDN(k-1)==1 && UPDN(k)==0)
            Change_DM(k)=0;
            Change_DM(k+1)=0;
        end
        if (UPDN(k-1)==0 && UPDN(k)==1)
            Change_DM(k)=0;
            Change_DM(k+1)=0;
        end
  end

% Create the DAC (CT-DAC)
% -----------------------
DAC_DM= zeros(1,km,'double');
for k=3:1:km-1
    if UPDN(k)==1;
        if beh(k)==1;
            if Change_DM(k)==1;
                DAC_DM(k)=DAC_DM(k-1)+Delta;
            else
                DAC_DM(k)=DAC_DM(k-1);
            end
        elseif beh(k)==0;
            if Change_DM(k)==1;
```

```
                if Change_DM(k-1)==1
                    DAC_DM(k)=DAC_DM(k-1);
                else
                    DAC_DM(k)=DAC_DM(k-1)+Delta;
                end
            else
                DAC_DM(k)=DAC_DM(k-1);
            end
            if ( Change_DM(k)==1 &&  Change_DM(k+1)==0 &&  Change_DM(k-1)==1 );
%--------------
                    DAC_DM(k)=DAC_DM(k-1)+Delta;
            end
        end
    elseif UPDN(k)==0;
        if beh(k)==1;
            if Change_DM(k)==1;
                if ( Change_DM(k+1)==1 && Change_DM(k-1)==1 )
                    DAC_DM(k)=DAC_DM(k-1);
                else
                    DAC_DM(k)=DAC_DM(k-1)-Delta;
                end
            else
                DAC_DM(k)=DAC_DM(k-1);
            end
        elseif beh(k)==0;
            if Change_DM(k)==1;
                DAC_DM(k)=DAC_DM(k-1)-Delta;
            else
                DAC_DM(k)=DAC_DM(k-1);
            end
            if ( Change_DM(k+1)==1 && Change_DM(k-1)==0 && Change_DM(k)==1); %-
-----------
                DAC_DM(k)=DAC_DM(k-1)-Delta;
            end
        end
    end
    if DAC_DM(k)>=Quantization_Levels(Number_of_Levels)
        DAC_DM(k)=Quantization_Levels(Number_of_Levels-1);
    end
    if DAC_DM(k)<=-amp
        DAC_DM(k)=-amp;
    end
end
DAC_DM(1)=0;

%  Calculate the Quantization Error Of (x)
%  ---------------------------------------
error_DM=x-DAC_DM;        % Quantization Error

% Normalizing Signals
```

```
% ------------------------
DAC_normalized=DAC+Delta/2;
Error_normalized=Vref_error-Delta/2;

DAC_DM_normalized=DAC_DM+Delta/2;
error_DM_normalized=error_DM-Delta/2;
```

```
Simulation_Results

-------------------
```

```
% Root Mean Square (RMS)
%-------------------------
x_RMS=rms(x);
DAC_RMS=rms(DAC);
Gx_RMS=rms(Gx);
error_RMS=rms(Vref_error);
EE_RMS=rms(EE);
Error_DM_rms=rms(x-DAC_DM-Delta/2);
Error_Vshift_rms=rms(x-DAC-Delta/4);
Error_Vref_rms=rms(EEE);

% SQNR
% ------
SQNR_DM=10*log10((x_RMS/Error_DM_rms)^2);
SQNR_Vref=10*log10((x_RMS/Error_Vref_rms)^2);
SQNR_WNN=10*log10((x_RMS/EE_RMS)^2);

% SNR
% ---
SNR_DM= snr(DAC_DM+Delta/2);
SNR_Vref= snr(V_combined);
SNR_WNN= snr(Gx);

% SNDR
% -----
SNDR_DM= sinad(DAC_DM+Delta/2);
SNDR_Vref= sinad(V_combined);
SNDR_WNN= sinad(Gx);

%  SFDR
% ------
SFDR_DM= sfdr(DAC_DM+Delta/2);
SFDR_Vref= sfdr(V_combined);
SFDR_Gx= sfdr(Gx);

% ENOB
% -----
ENOB_DM=((SNDR_DM)-1.76)/6.02;
```

```
ENOB_Vref=((SNDR_Vref)-1.76)/6.02;
ENOB_WNN=((SNDR_WNN)-1.76)/6.02;

% Summary Table of ADC Performance
% ---------------------------------
Table = {'V_DM';'V_Ref';'V_WNN'};
SQNR = [SQNR_DM;SQNR_Vref;SQNR_WNN];
SNR = [SNR_DM;SNR_Vref;SNR_WNN];
SNDR = [SNDR_DM;SNDR_Vref;SNDR_WNN];
SFDR = [SFDR_DM; SFDR_Vref; SFDR_Gx];
ENOB = [ENOB_DM; ENOB_Vref; ENOB_WNN];
Freq=[Frequency;Fs;resolution];

clc
Simulation_Results=table(SQNR,SNR,SNDR,SFDR,ENOB,Freq,...
    'RowNames',Table)

Best_iteration_resukts=ENOB_WNN_best;
```

```
Simulation_Results =

              SQNR      SNR       SNDR      SFDR      ENOB      Freq

              ______    ______    ______    ______    ______    _____

    V_DM      25.314    20.903    20.902    23.672    3.1798        1

    V_Ref     26.175    36.353    36.318    43.346    5.7406    1e+06

    V_WNN     50.715    226.42    226.42    233.91    37.319    1e-06

|-----------------------------------------------|

|                     Ploting                   |

|-----------------------------------------------|
```

```
% Plot Delta Modulation (V_DM)
% ------------------------------
figure(1); title ('Simulation of CT-ADC Delta Modulation (V_D_M)');
xlabel('Time (Microseconds)'); ylabel('Amplitude (V)'); grid on;
hold all; plot(x,'r');
hold all; plot(DAC_DM+Delta/2,'b');
hold all; plot(x-DAC_DM-Delta/2,'k');

% Plot Digitalized Output Signal (Vref)
% --------------------------------------
```

```
figure(2); title ('Simulation of CT-ADC Delta Modulation (V_R_E_F)');
xlabel('Time (Microseconds)'); ylabel('Amplitude (V)'); grid on;
hold all; plot(x,'r');
hold all; plot(Vref-Delta/8,'b');                    % Plot Normalized (Vref)
hold all; plot(Vref_error-Delta/8,'black');          % Plot Normalized (Vref
error)

% Plot V_combined Output
% ----------------------
figure(3); title ('SQNR for the Simulated CT- ADC ( V_C_o_m_b_i_n_e_d)');
xlabel('Time (Microseconds)'); ylabel('Amplitude (V)'); grid on;
hold on; plot (x ,'r');
hold on; plot (V_combined,'k');
hold on; plot (EEE,'b');

% Plot V_WNN Output Singal
% --------------------
figure (4); title ('SQNR for the Simulated CT- ADC ( V_W_N_N)'); xlabel('Time
(Microseconds)'); ylabel('Amplitude (V)'); grid on;
hold on; plot (x ,'r');
hold on; plot (Gx,'k');
hold on; plot (x-Gx,'b');

% Plot SINAD
% -------------
figure (5); sinad((DAC_DM+Delta/2),Fs);
figure (6); sinad(V_combined,Fs);
figure (7); sinad(Gx,Fs);

% Plot SFDR
% -------------
figure (8); sfdr((DAC_DM+Delta/2),Fs);
figure (9); sfdr(V_combined,Fs);
figure (10); sfdr(Gx,Fs);

% Plot SFDR
% -------------
figure (11); snr((DAC_DM+Delta/2),Fs);
figure (12); snr(V_combined,Fs);
figure (13); snr(Gx,Fs);
```