DESIGN OF A RECONFIGURABLE POWER-ADAPTIVE HIGH-RESOLUTION NEURAL DATA COMPRESSION ALGORITHM

By

Mohammed Ashraf Hassan

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2017

DESIGN OF A RECONFIGURABLE POWER-ADAPTIVE HIGH-RESOLUTION NEURAL DATA COMPRESSION ALGORITHM

By
Mohammed Ashraf Hassan

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Under the Supervision of

Prof. Dr. Ahmed Eladawy

Professor

Electronics and Communications
Engineering Department
Faculty of Engineering, Cairo University

Dr. Hassan Mostafa

Assistant Professor

Electronics and Communications
Engineering Department
Faculty of Engineering, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2017

DESIGN OF A RECONFIGURABLE POWER-ADAPTIVE HIGH-RESOLUTION NEURAL DATA COMPRESSION ALGORITHM

By
Mohammed Ashraf Hassan

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Approved by the
Examining Committee

_____
Prof. Dr. Ahmed Eladawy, Thesis Main Advisor

_____
Prof. Dr. ----------, Internal Examiner

_____
Prof. Dr. ----------, External Examiner
(----------)

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT

2017

Engineer's Name: Mohammed Ashraf Hassan Enal
Date of Birth: 21/09/1991
Nationality: Egyptian
E-mail: mohammedenal@gmail.com
Phone: +201111353779
Address: Electronics and Communications Engineering Department,
Cairo University, Giza 12613, Egypt

Registration Date: 01/3/2014
Awarding Date: --/--/2017
Degree: Master of Science
Department: Electronics and Communication Engineering

Supervisors:

Prof. Dr. Ahmed Eladawy
Dr. Hassan Mostafa

Examiners:

Prof. Dr. Ahmed Eladawy (Thesis main advisor)
Prof. ----------(Internal examiner)
Porf. ----------(External examiner)
(----------)

Title of Thesis:

Design of a Reconfigurable Power-adaptive High-Resolution Neural Data Compression algorithm

Key Words:

Neural Signals; Multichannel Neural Recording; Data Compression; Image processing; Low-power Design; Power Harvesting; HW Implementation

Summary:

In this thesis, five different proposed low-power image compression algorithms based on discrete cosine transform (DCT) and discrete wavelet transform (DWT) are investigated and compared to provide the best trade-off between compression performance and hardware complexity. Finally, harvested power adaptive high-resolution neural data compression is introduced to control the compression algorithm according to available harvested power. Hence, maximum signal to noise and distortion ratio (SNDR) is achieved based on the available harvested power without any data loss.

Acknowledgments

First of all, I would like to thank Prof. Ahmed Eladawy and Dr. Hassan Mostafa for giving me the opportunity to work onto this subject which interests me a lot and enhances my knowledge and career. I would also like to thank Dr. Hassan for his help, suggestions and support.

Table of Contents

List of Tables

List of Figures

Nomenclature

MEMS	Micro-Electro-Mechanical Systems
EEG	Electroencephalography
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
2D-DCT	Two Dimension Discrete cosine Transform
2D-DWT	Two Dimension Discrete Wavelet Transform
ECOG	Electrocorticography
EEG	Electroencephalography
JPEG	Joint Photographic Experts Group
SNDR	Signal to Noise and Distortion Ratio
HW	Hardware
HDL	Hardware Define Language
DFT	Discrete Fourier Transform
MJPEG	Motion Joint Photographic Experts Group
FOM	Figure of Merit
ASIC	Specific Integrated Circuits
RAM	Random Access Memory
PV	Photovoltaics
RF	Radio Frequency
PZT	Lead Zirconate Titanate
IOT	Internet of Things
TX	Transmitter
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IP	Internet Protocol

Abstract

Nowadays, brain scientific research progress depends on signal compression at high spatial resolutions, for low-rate transmission through wireless connection to the outside world and efficient storage. Without data compression, these data rates would conflict the neurophysiologic restrictions in terms of low energy and low area consumption. So that neural data compression at the implant site is substantial in order to conform with the wireless rates restrictions. In this thesis, the high spatial correlation is utilized to increase the data compression ratio. Then, five different proposed low-power image compression algorithms based on discrete wavelet transform (DWT) and discrete cosine transform (DCT) are investigated and compared to provide the best trade-off between compression performance and hardware complexity. Hence, the Adaptive 2D-DWT algorithm is deduced as a promising solution for low-power implantable devices.

Furthermore, current treatment devices need the complete waveform and history for every electrode to be extracted instead of extracting the special signal features only to be able to detect and diagnose neural brain disorders. So that it must be guaranteed that the detected neural data can be transmitted continuously without any stops or data loss and also it must be guaranteed that the compressed data can be decompressed at the other side with high quality without significant distortion. In this thesis, the neural compression algorithm is adapted according to the available harvested power budget. Therefore, the maximum signal to noise and distortion ratio (SNDR) is achieved based on the available harvested power budget without any data loss.

Chapter 1 Introduction

1.1. Background

Over the last 40 years, implantable electronic devices and systems have faced a significant transformation, becoming a valuable biomedical tool for measuring, monitoring and stimulation physiological responses using wireless communication. The discovery and posterior advancement of these devices have relied heavily on the growing knowledge related to various aspects of the human neuro system, and the development of electronics technologies capable of interfacing with living tissues and organs at microscale and nanoscale. Increasing in stability, miniaturization and lower power requirement of modern electronics led to a plenty of miniature wireless electronic devices, such as sensors and intelligent gastric, implantable cardioverter defibrillators, implantable cochlear, and deep brain, nerve, and bone stimulators are implanted in patients worldwide [29,30,31,32]. Advances in semiconductor technology, particular in the area of micro fluidic lab-on-chip biomedical systems and micro-electro-mechanical systems (MEMS) have allowed for the development of units for rapid diagnostics, and precisely controlled pulsatile, sustained or rapid delivery of complex therapeutics and drugs [33].

Furthermore, these devices are used for the development of tissue engineering platforms and also have been used in regenerative medicine applications, particularly where nervous and muscular tissues are concerned. In addition to growing the survival rate and the life quality of patients globally, implantable electronic devices have contributed significantly to assessment of the biological processes taking place within the human body, including the hard mechanisms of neural control and communication, and greatly enhanced the understanding of how these are affected by various diseases and remediation. Ex MEMS and dielectric elastomer actuators have been used to explore the manner in which biological cells modulate their behavior, proliferate or differentiate in response to electrical and mechanical stimuli, knowledge which is fundamental for adequate tissue engineering design [34]. In addition to playing a deep role in the progress of biomedical sciences and regenerative medicine, communication technologies and implantable information drive memorable changes in the cultural and social attitudes of people towards technology. There, implantation is viewed beyond the medical context as a means to promote the experiences and abilities of healthy individuals. Despite of essential innovations in the application and fabrication of implantable biomedical electronic systems since the first implantable heart pacemakers, the modern implants are still faced with a number of challenges [35].

In terms of device production, there is a strong trend to produce devices with ever size and weight in order to make them compatible with normal human activities and enhance leisure for the host. Implants that weight less than 1% of the patient's body weight are typically required. When used whether single-use batteries or rechargeable batteries significantly contribute to the overall dimensional size and weight of the device. Rechargeable batteries, like those are used in cochlear implants, can be recharged transcutaneously using external signals, e.g., pizoelectricaly, radio frequency (RF), ultrasound, infrared light, low-frequency magnetic field, and so on. More recently, internal charging using the energy produced by the physiological environment

or natural body motion has been investigated. Single-use, non-rechargeable batteries, like those are used to support pulse generation in deep brain stimulators and cardiac pacemakers, have a predetermined lifetime, at the end of which they have to be surgically replaced, at high cost to the patient and the healthcare system. Further miniaturization can be earned by means of battery-less implants, where energy harvested from natural or artificial power sources surrounding the patient is used directly to power the device [36]. Inductive and electromagnetic coupling are extremely used to power remotely battery less devices. In the former case, time-harmonic magnetic field generated by the low frequency alternating current in the external coil generates an alternating current in the implanted component, whereas in the latter, electromagnetic waves are generated from the antenna in the far field region to power the implanted chip. Biomedical actuators that do not base on the harvesting, traditional wireless delivery, accumulation and storage of power in electrical form have been explored for such high-energy actuation applications as mechanical adjustment in implantable devices and drug release.

At the same time, there is a strong emphasis on increasing the functionality and reliability of these electronic devices to support complex real-time stimulation, data collection, data compression and reliable wireless data transmission to external world. This increasing complexity of signal processing electronics further increases the power budget of the device, which should remain very low if the device is remaining working for extended periods of time. For instance, a wide band technology offers high speed data transfer between the implanted devices, e.g., implantable electronic cardiovascular devices, low interference potential and the medical practitioner, yet its implementation is limited due to its high power consumption.

The ability of the implanted devices, such as glucose monitors, pacemakers, and insulin-delivery systems, smart prosthetics, and neural stimulators, to be easily interrogated by health practitioners also makes these systems susceptible to hacking [37]. In addition to having access to secret patient data, the systems can be reprogrammed, interfering with the correct device operations. Therefore, firewalls, including security check protocols, security measures, restricted network access and data encryption should be seriously considered.

Application of molecular-scale and nano-scale technologies for fabrication and design of the implantable circuits can lead to remarkable progress in power dissipation and integration density, enabling nano-biorobotics and neuroelectronic interfacing. However, current biomedical technologies are still faced with challenges, like relatively high standby power consumption, lower reliability, and electron leakage due to insufficient insulation.

Furthermore, in an effort to improve the resolution of the collected biological signals, the increasing number of electrodes demands more energy to be delivered to the electrode array, thus potentially growing the thermal energy dissipated within the implant circuitry. Given the high cost and time associated with the surgical implantation of the device and the recovery of the patient, long-term reliability of the device is fateful.

The drive towards small, light and flexible devices may reduce mechanical robustness of the implant; offensive cleaning procedures used on the devices prior to implantation may further contribute to weakening of the organic layers. The ensuing in loss of integrity and vivo degradation may be harmful to the performance of the system, leading to the system failure, e.g., subsequent surgical removal and electrical shorting. The implanted device and its degradation by-products may stimulate activation of a

range of invulnerable mechanisms, leading to inflammation, which in turn may further contribute to the implant degradation.

Achieving suitable biocompatibility is a hard matter, due to the dynamic multifaceted nature of the host biological response to synthetic and organic materials used in device fabrication. Where in vivo stimulation or sensing is required for a short period of time, resorbable implantable electronic systems can provide a solution to overcome inflammation and infections associated with long term implant utilization. The premise is that the materials used in system fabrication are biodegradable and undergo controlled dissociation over time under normal in vivo physiological conditions. The degradation by-products forbidden minimal toxic response and are removed from the peri-implantation site by means of normal metabolic activity [39]. However, fabricating a high performing electronic device from entirely biodegradable, non-toxic set of materials is a difficult undertaking, particularly at small scales. A combination of reliable and robust non-biodegradable silicon electronics with bioresorbable polymer platform offers both sufficient bulk degeneration and the flexibility of the device that the invulnerable response to the remaining material is minimal [40].

For the technology to be clinically implemented, however, the challenges associated with integration of sensitive electronics functions with the fabrication techniques used for production of biodegradable component, and the control over degradation kinetics and biocompatibility of the device should be addressed. In spite of many reports detailing the biological activity and degradation behavior of many commonly used materials in vitro and in vivo, the appreciation of these complex processes is yet to be adequate. The aim of this background is to discuss the challenges faced by modern implantable electronic systems and give a brief overview of the solutions that have been proposed, investigated and implemented in order to overcome these challenges.

When designing an implantable electronic device, several general requirements need to be addressed, namely minimal weight and size, low power consumption, high reliability, high data rate and data latency. As the case with any commercial product, the design of the implantable systems is heavily influenced by the demands and preferences of their consumers.

In addition to being less invasive to the patient body during the implantation, lighter and smaller devices are likely to result in less pain and discomfort to the host during recovery and use. The extravagant size and weight may be harmful to the recovery process by putting pressure on the adjacent tissues that have already been damaged as a result of surgery, contributing to the inflammatory processes within the peri-implant space. Light and small devices are less restrictive in terms of normal level of human activity, and thus sustain better quality of life to the patients. The power source and encapsulation components remain the major contributors to the overall size and weight of the device, whereas the electric circuitry components have decreased dramatically with the advancements in nanotechnology and MEMS. Coupling capacitors used to ensure charge balance and effectively minimize current leakage may further increase the volume of the implantable module. Lower power consumption is important in terms of both the long-term performance of the device and the safety to the patient.

Furthermore, the power use by interface electronics should be minimized to ensure longevity of the implants with single-use batteries, as the replacement of such a device would require a costly and invasive surgical procedure. Although using a rechargeable battery may require the need for battery replacement surgical interference,

the need for frequent charging may be inconvenient, resource-consuming activity and time-consuming.

Electrocorticography (ECoG) is a type of electrophysiological monitoring that uses electrodes placed directly on the naked surface of the brain to record electrical activity from the cerebral cortex. But, conventional EEG electrodes monitor this activity from the exposed surface of the cortex (outside the skull). ECoG can be performed either extra operative ECoG (outside of surgery) or intraoperative ECoG (during surgery in the operating room). Because a surgical incision into the skull (craniotomy) is required to implant the electrode grid, ECoG is an invasive procedure.

Electrocorticography (ECoG) signals are composed of local field potentials, recorded directly from outside the skill. The potentials occur primarily in cortical pyramidal cells, In addition, thus should be conducted through several layers of the cerebral cortex, arachnoid mater and cerebrospinal fluid before reaching subdural recording electrodes (placed just below outer cranial membrane). However, to reach the scalp electrodes of a conventional EEG, electrical signals should also be conducted through the skull, where potentials rapidly reduce due to the bone low conductivity. Hence, the Electrocorticography (ECoG) spatial resolution is higher than conventional EEG, a critical imaging advantage for presurgical planning [41].Electrocorticography has a spatial resolution of 1 cm and a temporal resolution of approximately 5 ms [42]. Using depth electrodes, the local field potential gives a measure of a neural population in a sphere with a radius of 0.5:3 mm around the tip of the electrode [17]. With a sufficiently high sampling rate (more than about 10 kHz), depth electrodes can also measure action potentials. In which case the spatial resolution is down to individual neurons, and the field of view of an individual electrode is approximately 0.05-0.35 mm [17].

Nowadays, electroencephalogram classification has become an important problem in several fields. In the medicine field, EEG detection could be incredibly promising for stroke or seizure detection in patients that are oversensitive to such conditions, and a great deal of research has already been put into solving this problem. Other medical applications include manufacturing transportation devices for patients with limited motor abilities to control using simply their thoughts or extremely tender facial movements. Both of these will pick up EEG and an accurate and classifier will lead to successful creation of such a system which would change the patients' lives with such a failure. Yet other neuroscience and psychology applications, Electrocorticography classification can give insight into the human brain inner workings.

In the biomedical engineering field, neural data recording has a considerable importance especially by employing neuroprosthetic devices and brain machine interfaces. Furthermore, multichannel neural recording is essentially for bio analysis and is commonly used. However, recording big amounts of data has been a challenging task; for example, a typical recording experiment in which data is obtained from a 1024-channel electrode array at the rate of 64 kHz per channel with 12-bit precision yields a data rate of around 768 Mbps, which is much beyond the capacity of state-of-art wireless links that are used in neural applications. Wireless transmission and reception are used for conducting experiments on freely behaving primates and animals. Another important requirement in a neural recording system is that it must be able to operate with a low power. All neural chips which are implanted in living human bodies must be able to operate at a very low power (less than or equal to 8–10 mW), failing would lead to temperature increasing (exceed 1°C) and cause damage of neural

tissue. Thus, a compression algorithm which prepared for neural applications must be simple, such as in brain-machine interfaces.

1.2. System Architecture

Figure 1-1 shows the full implantable neural measurement system architecture. This neural system consists of multielectrode array, analog frontend (amplifiers), analog to digital converter, data compression and signal processing, power supply, low power wireless transceiver. All these implantable blocks must be low-power, small area, safe on the human body. Neural signals have been recorded from the implantable multielectrode array will be amplified in the analog front-end (AFE) and converted to digital neural data using analog to digital (ADC) block. Subsequently, the digital data runs into the main digital module where data compression and signal processing take place. Hence, the low-rate wireless transceiver transmits the compressed neural data to the outside world (reconstruction base station), where signal reconstruction and decompression are performed. Since the system is fully implantable, energy has to be available from implantable power supply (rechargeable battery or harvesting system).

In neural implantable measurement devices the wireless link to the outside world is always the functional bottleneck in terms of a very limited data rate of a few MBit/s or in terms of limited available energy. For the transmission of neural stream data, wireless data rates in the order of 200 MBit/s (1000 electrodes with a sample rate of 20 kS/second/channel and 10 Bit of resolution) could easily occur. In addition to, in neural implantable measurement devices for raw data transmission, complete waveforms are needed instead of extracted signal features. Especially in medical diagnostics information preservation, neural data could be beneficial such as detection of epileptic disorders. In order to overcome these bandwidth and energy restrictions, data compression at the implant site is one possibility of addressing this obstacle. Hence, data compression block is a mandatory block in the neural implantable measurement system.

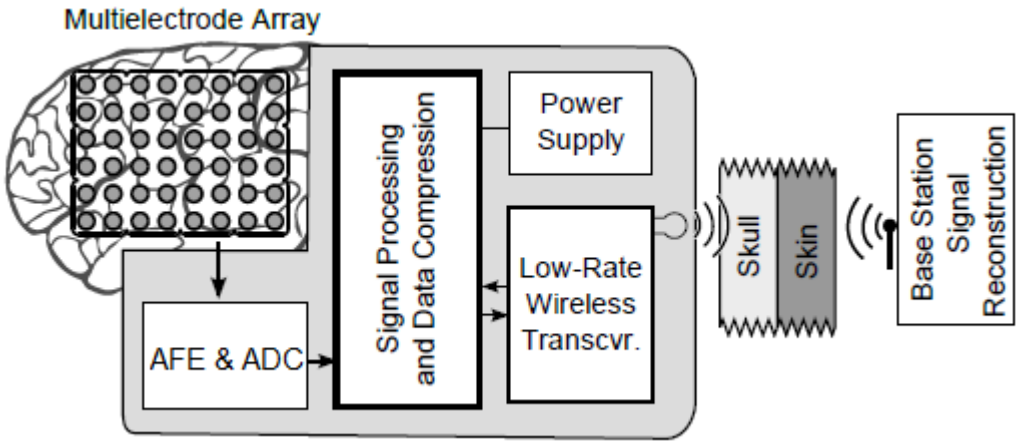


Figure 1-1: Neural System Full Architecture

1.3. Motivation

Neural implantable recording systems are widely utilized to treat neural disorders as Parkinson and Epilepsy diseases. To diagnose and detect these disorders, the complete waveform for every electrode needs to be extracted instead of extracting the special signal features only. Hence, data compression at the implant site is necessary to be able to transmit these huge sizes. To conform to the implantable subsystem requirements such as limited wireless transmission bandwidth with the outside world and low received electric power despite of huge number of recording channels, which reaches up to 1024 channels and even more to cover finer spatial resolution of the recordings [21], a low-power and efficient compression algorithm is needed.

In this thesis, high resolution neural signals is targeted whether Electrocorticography (ECOG) or Electroencephalography (EEG) or any other high resolution neural signals. Spatial space reaches up to 0.5 mm between the neighboring electrodes. Temporal sampling rate reaches up to 20 KSample/s/channel. These high resolution signals characterized high spatial domain correlation as well.

In neural measurement systems the wireless transmission node is the functional bottleneck in terms of a very limited data rate of a few MBit/s [4]. For the transmission of high resolution neural raw data for 1024 electrodes, and electrode resolution is 8-bit with previous mentioned high rate (20 KSample/s/channel), wireless data rates are in order of 200 MBit/s. Hence, Powerful and efficient compression algorithms at the implant site to comply with this huge increase in neural data size is required.

In the implantable embedded devices, powerful compression algorithms and higher compression ratios are not the unique metrics, but also the hardware efficiency (Low-Power and Area-Efficient) is considered, because power consumption is the major parameter in implantable devices. In addition, the high sampling rate places another restriction on the hardware latency of the target compression algorithm to not violate the real-time processing. Hence, all these restrictions should be combined as design guidelines to choose the most suitable compression algorithm.

Neural Data compression has an important feature which is 2D correlation, so that both spatial and temporal compression need to be considered. Hence, a better compression performance can be gained. Most of compression techniques which have been proposed for multichannel ECOG or EEG are based on the similarity between neural data compression and image compression [1,3,6].Image compression algorithms utilize the spatial correlation between adjacent electrodes only like JPEG and JPEG2000. It is obvious that these algorithms could not be used directly because they are very complex and could not achieve the low-power and real time restrictions. However, these algorithms should be modified to make them suitable for the implantable devices requirements.

Implantable devices need an efficient power source to supply it with the enough energy for the electrodes, analog interfacing and digital classification. Implantable devices are powered using couple of methods: power harvesting and implantable batteries. Implantable batteries provide the power for implantable neural devices. However, batteries have limited life time, fixed energy density, large size and chemical side effects. Thus, researchers have developed various methods to harvest energy for implantable neural devices. Devices powered by harvested energy provide more safety and comfort and have longer lifetime than conventional devices. Energies that may be scavenged include thermal energy, solar energy, infrared radiant energy, wind energy, waves energy, gravity energy, vibration energy, and body motion energy, wireless RF

radiation energy and transfer energy. Energy harvesting devices produce electric energy from their surroundings through direct energy conversion. The energy harvesting from environmental sources or human has been provided to be an effective alternative [13].

1.4. Neural Data Characteristics

To evaluate the compression algorithms performance for high-resolution neural data, virtual recorded data is used with the same signals characteristics (spatial and temporal correlation) of real data, because there is no available high resolution recorded data with these large sizes yet (1024 channels and more). Accordingly virtual data for 1024 channels with almost the same correlation of smaller sizes high resolution systems is used [1,3].
In order to measure the correlation between two signals X_1 and X_2, the Pearson Product-Moment Coefficient is used [11], as shown in Eq. (1).

$$r_{x_1x_2} = \frac{E[(x_1-\mu_{x_1})(x_2-\mu_{x_2})]}{\sigma_{x_1}\sigma_{x_2}} \quad (1)$$

The correlation coefficient $r_{x_1x_2}$ between two random variables X_1 and X_2 with Mean values μ_{x_1} and μ_{x_2} and standard deviations σ_{x_1} and σ_{x_2}.

The degree of correlation is classified in to [3]:

- $0 < |r| < 0.2$: weak correlation
- $0.2 < |r| < 0.5$: medium correlation
- $0.5 < |r| < 1$: strong correlation

The current model consists of 1024 channels organized in 32x32 grid. It has a strong average spatial correlation between adjacent channels of 0.6130 and maximum of 0.996, and goes lower when channels are spatially apart. In addition, it has a strong average temporal correlation between consecutive frames of 0.8250 and maximum of 0.9702, and goes lower when frames are not consecutive.

1.5. Contribution

This dissertation of this work includes the following contributions:

- Provide a review on different Compression designs, their architectures, simulation and test results.
- Compare between five compression designs that depend on two well-known bases and evaluate their performance, area, time and power. These bases are the two main transform coding methods which are used in image compression. They are (DCT) and (DWT). These two bases are widely used to compress images, videos and neural data.
- Introduce a new method which can control the compression algorithm according to available harvested power. Hence, maximum signal to noise and distortion ratio (SNDR) based on the available harvested power can be achieved without any data loss.

1.6. Organization of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 introduces a literature survey of the main compression algorithms. Chapter 3 provides a HW design and simulation results of the five proposed compression algorithms with their architecture, then makes a unified comparison between them with available system design code and HW design code. Chapter 4 introduces a harvested power adaptive high-resolution neural data compression algorithm as a most suitable compression algorithm to achieve the highest possible SNDR based on available harvesting power without any data loss or discontinuity in transmission to outside world. Then the thesis conclusion and future work are revealed in Chapter 5.

.

Finally, Appendix A shows a detailed description for the:

- System Level design

Chapter 2 Literature Survey of The Main Neural Compression Algorithms

2.1. Introduction

Image compression algorithms utilize the spatial correlation between adjacent electrodes. However, video compression algorithms utilize the spatial correlation and temporal correlation between consecutive frames, as well.

These bases are the two main transform coding methods which are used in image compression. They are (DCT) and (DWT). These two bases are widely used to compress images, videos and neural data.

It is obvious that these algorithms could not be used directly because they are very complex and could not achieve the low-power and real time restrictions. However, these algorithms should be modified to be suitable for the implantable devices requirements.

Finally, the comparison between all these proposed algorithms is made to provide the best trade-off between compression performance and hardware complexity.

2.2. Transform Coding

The five proposed compression algorithms that depend on two well-known bases are applied and evaluated their performance, area, time and power. These bases are the two main transform coding methods which are used in image and video compression. They are discrete wavelet transform (DWT), discrete Fourier transform (DFT) and discrete cosine transform (DCT). These two bases are widely used to compress images, videos and neural data.

2.2.1. Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is the equivalent of the continuous Fourier transform for discrete signals known. Discrete Fourier transform introduces the most popular base in digital signal processing, which transforms a discrete signal into a set of coefficients of a finite combination of complex sinusoids. The discrete Fourier transform is a base with the terms of a geometric progression in each column ‘c’ and row ‘r’, which defines the DFT matrix entries, as shown in Eq. (2).

$$f_{r,c} = (e^{-j\,2/N})^{rc} \tag{2}$$

where c/N denotes the sinusoids frequencies. If the original signal ‘x’ is evaluated for N = rT samples for all integers r = {0, ….N − 1} and the sampling period ‘T’, then the resulting infinite sequence is a periodic extension of the DFT periodic in ‘N’.

2.2.2. Discrete Cosine Transform (DCT)

The discrete cosine transform (DCT) is a mechanism for converting a signal into elementary frequency components [24,26]. DCT is similar to Discrete Fourier Transform (DFT) but with real coefficients instead of complex coefficients [7]. DCT-II is widely used for compression. The entries of the DCT matrix are given in Eq. (3).

$$C_{m,n} = \sqrt{\frac{2}{N}} * cos\left(\frac{\pi m\left(n+\frac{1}{2}\right)}{N}\right) \quad (3)$$

With m, n = 0, 1, 2 ... N−1 if m ≠{0, N} and $1/\sqrt{2}$ otherwise.

The two-dimensional DCT is used in JPEG image compression and MJPEG video compression. It computes the i, j^{th} entry of the DCT of an image, as shown in Eq. (4).

$$D(i,j) = \frac{1}{\sqrt{2N}} c(i)\, c(j) \sum_{x,y=0}^{N-1} p(x,y)\, cos\left(\frac{\pi(m+1)}{2N}\right) cos\left(\frac{\pi(n+1)}{2N}\right) \quad (4)$$

With m, n = 0, 1, 2... N−1 and c (k) = 1 if k ≠ {0, N} and $1/\sqrt{2}$ otherwise.

The common procedure, 2D-DCT of NxN block is computed and the result is quantized then entropy coded. Typically, N equals 8 and the DCT formula is applied to each row and column of the block. The result is an 8×8 frequency components matrix in which the top-left is the DC component and increase gradually horizontally and vertically to represent higher horizontal and vertical frequencies.

2.2.3. Discrete Wavelet Transform (DWT)

The two-dimensional Discrete Wavelet Transform (2D-DWT) is an effective mechanism for image compression and hence attracts much attention in recent years. It is used in JPEG2000 image compression [5,9,22,23]. DWT is any wavelet transform for which the wavelets are discretely sampled. The major difference between the Fourier Transform and the DWT is that Fourier transform decomposes the signal into cosines and sines, but the wavelet transform decomposes the signal into mutually orthogonal set of local wavelets.

There are three main types of DWT:

- **HAAR DWT**: is the simplest DWT which divides an image into four sub-bands by addition and subtraction. The procedure for two-dimensional Haar DWT is described as follows:

  (1) Vertical division: Vertical division divides an image to two separated divisions. The first part is the sums of two adjacent columns, which is stored as low frequency coefficients in left side. The other part is the differences of two adjacent columns, which is stored as high frequency coefficients in right side.

(2) Horizontal division: once the vertical division is done, the horizontal division divides the image into four parts. Sums of two adjacent rows are stored as low frequency coefficients in upper side. The other part is the differences of two adjacent rows are stored as high frequency coefficients in lower side.

- **UNIFORM DWT**: is a common DWT in the most of image processing algorithms like JPEG2000 and it's used to transform the frame to frequency domain. The DWT output is divided to 4 sub-bands, denoted as LL, LH, HL and HH. Then uses a uniform quantizer with one quantization value to compress the DWT sub-bands.

- **ADAPTIVE DWT:** is the same as uniform DWT, but it uses different quantization values to compress the DWT sub-bands. LL is the most important quarter so that it divided by the lowest quantization value. Then LH and HL are divided by intermediate quantization value. Finally, HH has the lowest importance so that it's divided by the highest quantization value.

There are two degrees of freedom to select the performance of DWT.

- Mother wavelets: like Daubechies 1-8, Symlets 4-8, Coiflets 1 and 2 and etc. Every wavelet has specific filter bank coefficients and specific performance.
- Levels: 2D-DWT generates 4 sub-bands, denoted as LL, HL, LH, and HH, 'L' stands for Low and 'H' stands for High. The low frequency sub-band LL preserves essential visional features for the original image and can be re-performed to decompose the second level of DWT if multi-resolution representation is required and so-on, as shown in Figure 2-1. Since the role of LL is more crucial than that of the other three sub-bands, finer quantization (more quantization intervals) should be applied on it. This improves the compression ratio without decreasing the image quality. Furthermore, within any single sub-band, if some values like left interval width, right interval width, and median can be computed in advance, they assist the implementation for a much more efficient quantization scheme. This is the essentially inspiration for the proposed approach and the execution steps are described in section 3 in details.

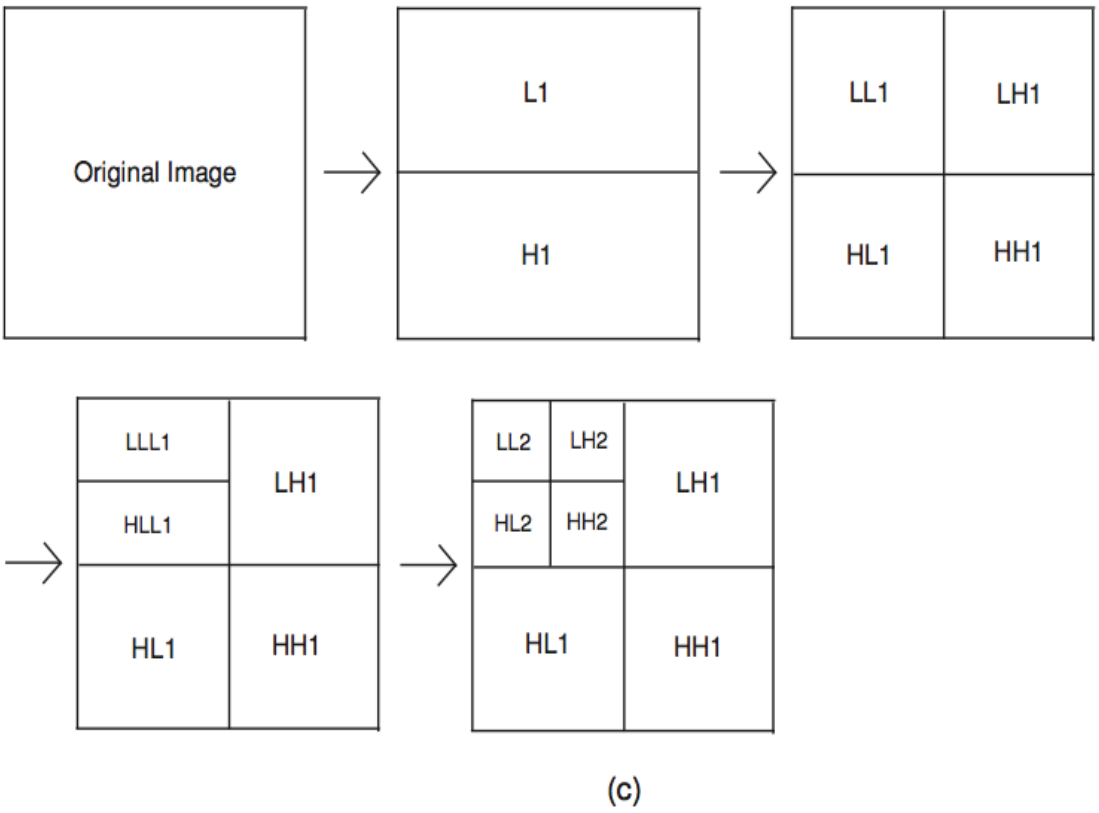


Figure 2-1: 2D-DWT decomposition into two levels [8].

Chapter 3 Analytical Comparison of Proposed Neural Compression Algorithms

3.1. Introduction

In this chapter, five proposed compression algorithms are introduced based on the similarity between neural data compression and image (video) compression [1,3,6,25,27] and evaluate their performance, area, time and power.
Video frames are generated such that the first frame consists of one sample from every channel with its order to combine together the first image then second frame consists of next sample from every channel with the same order to combine the second image and so on.
Image compression algorithms utilize the spatial correlation between adjacent electrodes only like JPEG and JPEG2000. However, video compression algorithms utilize the spatial correlation and temporal correlation between consecutive frames, as well, like H.264.
It is obvious that these algorithms could not be used directly because they are very complex and could not achieve the low-power and real time restrictions. However, these algorithms should be modified to be suitable for the implantable devices requirements.
Finally, the comparison between all these proposed algorithms is made to provide the best trade-off between compression performance and hardware complexity.

3.2. COMPRESSION ALGORITHMS

3.2.1. 2D-DCT8x8 Based Compression Method

The 2D-DCT8x8 algorithm is divided to 4 main steps, as shown in Figure 3-1:

- Discrete cosine transform
- Quantization
- Zigzag reorder
- Huffman

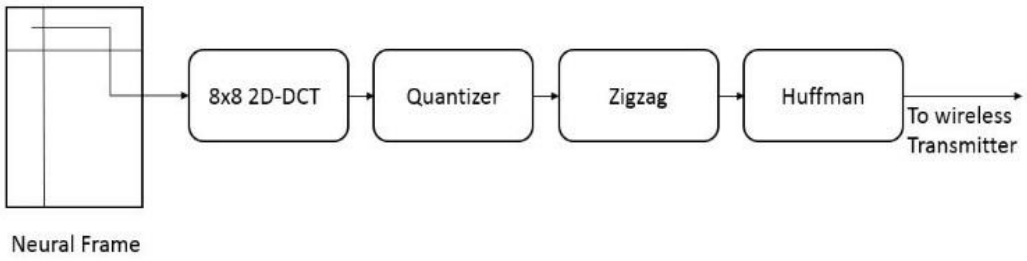


Figure 3-1: 2D-DCT8x8 Based Compression procedure

Electrodes are collected together to form the neural time instant frame. This frame of channels is divided into 8x8 blocks, working from left to right, top to bottom. Block size 8x8 has high hardware complexity but it has high compression ratio.
The 2D-DCT is applied on each block separately to be converted to frequency representation. The two-dimensional Discrete Cosine Transform is performed by coefficients matrix multiplication [6].

$$D = T\ P\ T' \tag{5}$$

As shown in Eq. (5), matrix p (the neural data 8*8 block) is left multiplied by the DCT coefficients matrix T_{8x8}, as shown in Figure 3-2, this to transform the rows. Then, the result is multiplied by the transpose of DCT coefficients matrix T'_{8x8} to transform the columns.

$$\begin{bmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .2778 & .0975 & -.0975 & -.2778 & -.4157 & .4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & .4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \\ .0975 & -.2778 & .4157 & -.4904 & .4904 & -.4157 & .2778 & -.0975 \end{bmatrix}$$

Figure 3-2:T_{8x8}Coefficients Matrix

Then, the 8x8 block of DCT frequency components is ready for quantization, as shown in Figure 3-2. This stage is the main stage to control the compression ratio and quality level. Quantization is the only lossy stage duo to rounding process, as shown in Eq. (5). That means without this stage, the data size can't be shrunk and the data can't be decompressed without any lossless. And this quantization is applied by rounding process after data scaling as shown in Eq. (6).

$$A = round\ (\frac{D * Qualit\ \ level}{Q\_matrix}) \tag{6}$$

Here a scalar constant 'Quality Level' is used as a quality controller, it changes from 1 to 10. For a highest quality and the lowest compression ratio select '10' and for a lowest quality and the highest compression select '1'. Then, it divided by the quantization value according to Q_matrix, as shown in Figure 3-3.

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 36 & 55 & 64 & 81 & 194 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Figure 3-3: Q_matrix_{8x8}

Quantizer output should have a lot of zeros components in the right bottom corner and decrease gradually to be minimum at the left top side, so that JPEG standard block ZIGZAG reorder is proposed to be used to reorder the components from lower probability to be zero to higher probability to be zero to get higher compression ratio in the Entropy stage, as shown in Figure 3-4. It needs a memory size 64B to reorder the 8x8 quantized block.

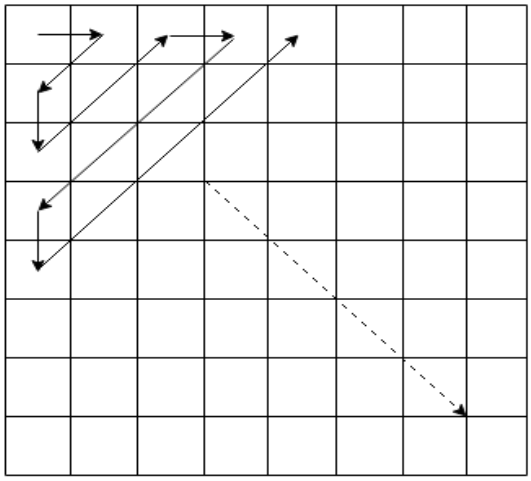


Figure 3-4: Zigzag Order 8x8

Entropy stage is a lossless stage replaces nonzero components to a stream of binary bits. Huffman coding is a common method to encode the DCT components with variable length codes according to common tables that are assigned according to statistical probabilities. A frequently used symbol will be encoded with a short code, while symbols that are rarely used are represented by a long code.
JPEG standard uses up to 4 tables divided to 2 sets luminance (DC and AC) and chrominance (DC and AC). The luminance tables are proposed to be used because Neural signals DCT components have a similar behavior of luminance DCT components.
This stage is implemented by storing these tables in Lookup Tables and accesses it sample by sample to be encoded.

3.2.2. 2D-DCT4x4 Based Compression Method

The 2D-DCT8x8 based algorithm requires high computation power and large area, and thus is not a suitable choice for implantable devices. Correspondingly, the same algorithm with smaller block 4x4 is used.

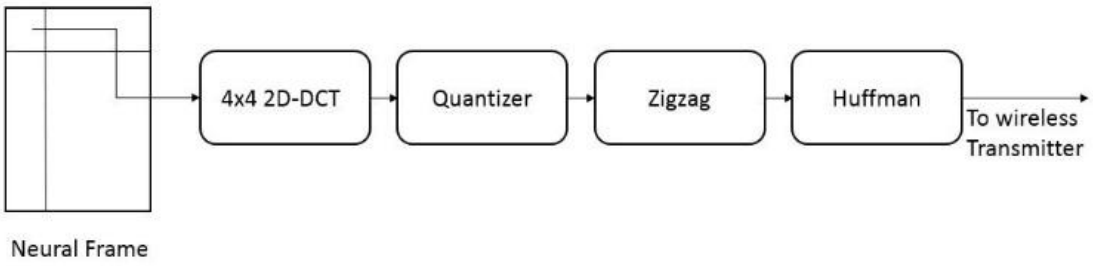


Figure 3-5: 2D-DCT8x8 Based Compression procedure.

4x4 2D-DCT is performed by matrix multiplication too with coefficients matrix T_{4x4}, as shown in Figure 3-6.

$$\begin{bmatrix} .5 & .5 & .5 & .5 \\ .6353 & .2706 & -.2706 & -.6353 \\ .5 & -.5 & -.5 & .5 \\ .2706 & -.6533 & .6533 & -.2706 \end{bmatrix}$$

Figure 3-6: T_{4x4} Coefficients Matrix

Then the DCT frequency components are quantized with the same procedure as 8x8 2D_DCT based algorithm with the proposed Q_marix_4x4, as shown in Figure 3-7.

$$\begin{bmatrix} 16 & 16 & 17 & 21 \\ 16 & 17 & 21 & 24 \\ 17 & 21 & 24 & 36 \\ 21 & 24 & 36 & 57 \end{bmatrix}$$

Figure 3-7: $\mathbf{Q\_matrix_{4x4}}$

Quantizer output should have a lot of zeros components in the right bottom corner and decrease gradually to be minimum at the left top side, ZIGZAG Reorder to reorder the components from lower probability to be zero to higher probability to be zero to get higher compression ratio in the Entropy stage, as shown in Figure 3-8.

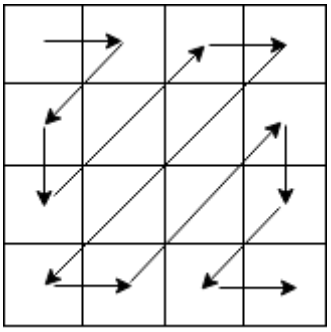


Figure 3-8: Zigzag Order 4x4

Then a memory size 16B is needed to apply ZIGZAG reorder procedure on the 4x4 quantized block. Then, the output is Huffman encoded, as shown in Figure 3-5.

In 4x4 2D-DCT, the same Entropy stage (Huffman coding) is used to replaces nonzero components to a stream of binary bits. Huffman coding is a common method to encode the DCT components with variable length codes according to common tables that are assigned according to statistical probabilities in 4x4 blocks as well. A frequently used symbol will be encoded with a short code, while symbols that are rarely used are represented by a long code.

In 4x4 2D-DCT, the same 4 tables which are used in JPEG standard are used, these tables are divided to 2 sets luminance (DC and AC) and chrominance (DC and AC). The luminance tables are proposed to be used because Neural signals DCT components have a similar behavior of luminance DCT components.
This stage is implemented by storing these tables in Lookup Tables and accesses it sample by sample to be encoded.

3.2.3. ADAPTIVE 2D-DWT Based Compression Method

The main differences between DWT-based and DCT-based methods are the two-dimension transformation block and the quantizer block but the entropy stage (Huffman Coding) is the same in 2 algorithms, as shown in Figure 3-9. Input data are directly 2D-DWT transformed as a one block without division to smaller blocks. That means the 2D-DWT block size is the same as the neural frame size.

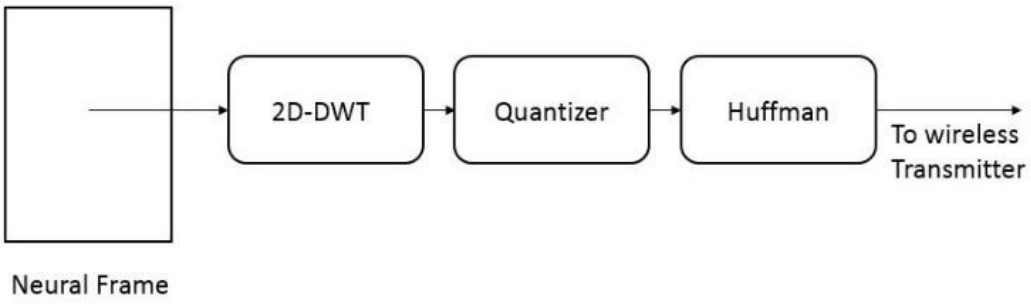


Figure 3-9: Adaptive DWT based Compression procedure.

In this work the Adaptive quantization DWT compression algorithm is proposed to be used instead of regular DWT (uniform quantization), because it achieves higher performance than regular DWT [2], with a small hardware overhead. The complexity of DWT depends on the length of the filter coefficients and the number of levels. Hence, 1-level DWT with the biorthogonal spline 5/3 filter is a suitable solution for this case [1]. 2D-DWT is used to transform the frame to frequency domain. The 2D-DWT output is divided to 4 sub-bands, denoted as LL, LH, HL and HH as shown in Figure 3-10.

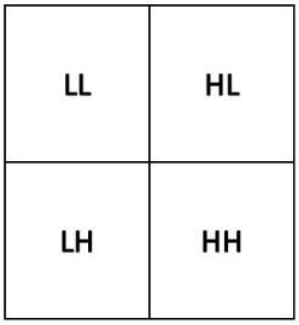


Figure 3-10: 1-level DWT Decomposition

The Adaptive quantizer uses different quantization values to compress the DWT sub-bands. LL is the most important quarter so that it divided by the lowest quantization value. Then LH and HL are divided by intermediate quantization value. Finally, HH has the lowest importance so that it's divided by the highest quantization value. To perform the adaptive quantization algorithm, left interval width, the right interval width and median for each sub-band have to be computed in advance [2]

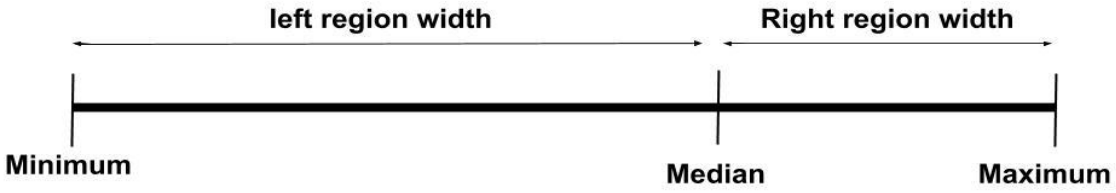


Figure 3-11: Adaptive DWT Procedure

$$Q_{i,j} = \text{sign}(C_{i,j}) \left[\frac{|C_{i,j} - \text{Median}|}{\text{Quant .Interval}} \right] \quad (7)$$

Quantized value of 2D-DWT output is calculated, as shown in Eq. (7). Quantization Interval will equal the right interval width divided by the quantization step if the DWT component is larger than the median, or will equal the left interval width divided by the quantization step if the DWT component is smaller than the median, as shown in Figure 3-11. Quantization step is used to control in the compression ratio. If it's large the compression ratio will be small, and vice versa.

After quantization, a lot of zeros should be in HH sub-band, and less in HL and LH sub-bands, then LL has a lowest number of zeros. So that the Quantization outputs LL,

HL, LH and HH respectively are reordered to be ready for Huffman encoder, as shown in Figure 3-12.

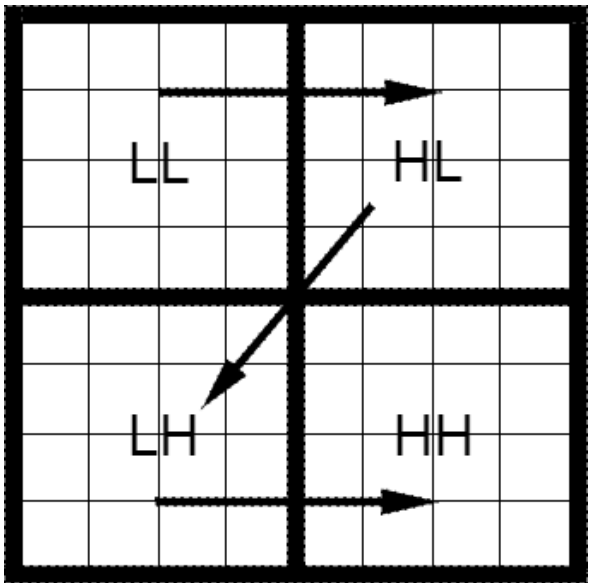


Figure 3-12: DWT Zigzag Order

In Adaptive 2D-DWT algorithm, the same 4 tables which are used in JPEG standard are used, these tables are divided to 2 sets luminance (DC and AC) and chrominance (DC and AC). The luminance tables are proposed to be used because Neural signals DWT components have a similar behavior of luminance DCT components.
This stage is implemented by storing these tables in Lookup Tables and accesses it sample by sample to be encoded.

3.2.4. Diff-2D-DCT8x8 Based Compression Method

To utilize the temporal correlation between consecutive samples at the same electrode (channel) as the spatial correlation between channels is utilized, a three dimensions algorithm needs to be applied like video compression algorithms to compress in three-dimensions, but these algorithms require high computation power and large storage.
To utilize the correlation between 8 frames together, 8KB memory size is needed for 1024 channels to store 8 consecutive neural frames to be processed simultaneous. And thus are not suitable for implantable devices. So that difference between consecutive frames algorithm is proposed to be used. In this algorithm, the 2D-DCT is applied on the frames difference to utilize the correlation between the consecutive time instants [1]. It needs 1KB only to store the last frame to be subtracted from the next frame.

Then, complete the remaining path as 2D-DCT8x8 based algorithm, as shown in Figure 3-13.

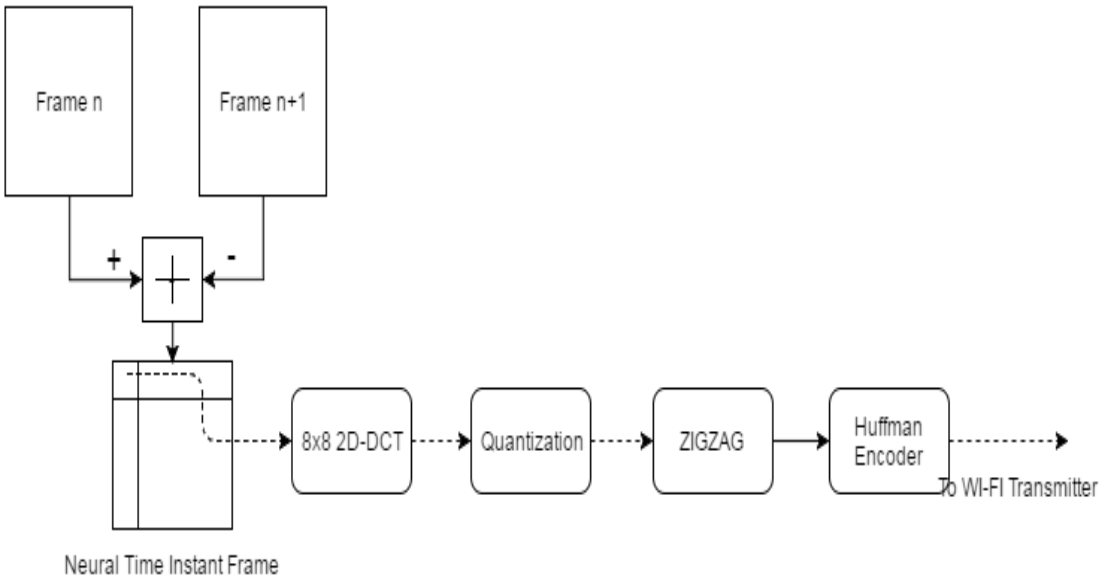


Figure 3-13: DF-2D-DCT8x8 Based Compression procedure

First frame is stored in the frame buffer waiting the second frame. After sampling the second frame, it will be subtracted from the first one and it will be stored in the buffer at the same time instead of the first frame to be processed with the next frame. The output from the subtraction operation will be divided into 8x8 blocks, working from left to right, top to bottom. Block size 8x8 has high hardware complexity but it has high compression ratio.

The Diff-2D-DCT is applied on each block separately to be converted to frequency representation. The two-dimensional Discrete Cosine Transform is performed by coefficients matrix multiplication [6].
The frequency components will be quantized with the same 8x8 matrix like 2D-DCT compression algorithm and the output will be zigzag reordered before Huffman encoding stage. Huffman coding uses the same 4 tables which are used in 2D-DCT compression algorithm.

3.2.5. Diff-2D-DCT4x4 Based Compression Method

In Diff-2D-DCT4x4compression algorithm, the goal is to utilize the spatial correlation on smaller area 4x4 instead of 8x8 with the temporal correlation at the same time to reduce the power and save the area and storage, as shown in Figure 3-14.

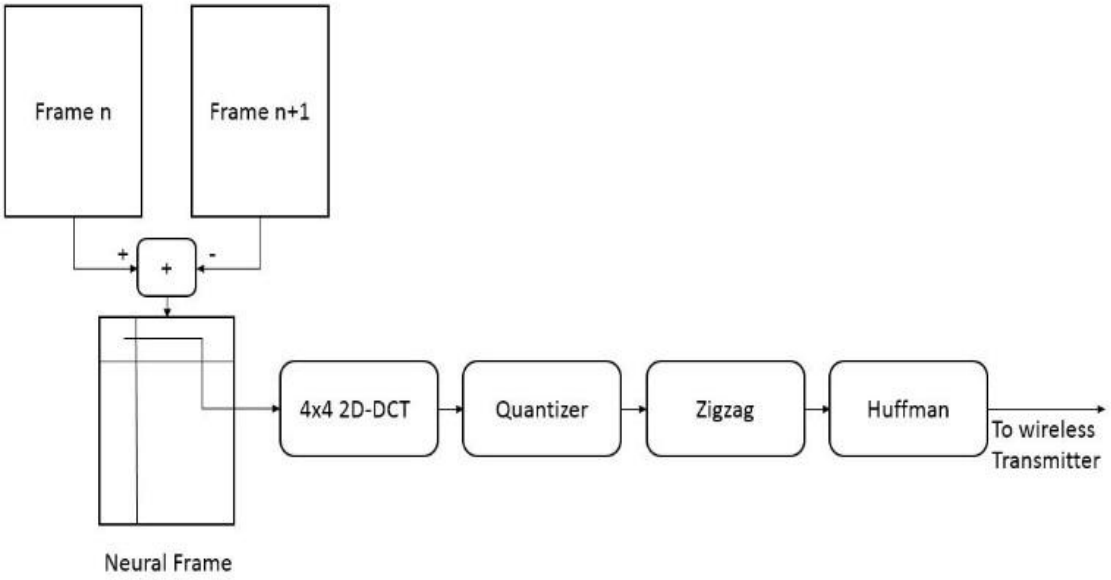


Figure 3-14: DF-2D-DCT4x4 Based Compression procedure

First frame is stored in the frame buffer waiting the second frame. After sampling the second frame, it will be subtracted from the first one and it will be stored in the buffer at the same time instead of the first frame to be processed with the next frame. The output from the subtraction operation will be divided into 4x4 blocks, working from left to right, top to bottom. Block size 4x4 has lower hardware complexity but it has lower compression ratio.

3.3. Result Comparison And Discussion

In order to evaluate the proposed compression algorithms three performance metrics are used:

- Compression ratio (R): to measure the ratio between compressed data size and original data size, as shown in Eq. (8).

$$R = \frac{\text{Compressed data size}}{\text{Original data size}} \tag{8}$$

- Signal to Noise and Distortion Ratio (SNDR): to measure the quality of reconstruction data $\widehat{D}$ after compression and decompression again compared to original data D [3], as shown in Eq. (9).

$$\text{SNDR} = 10\text{dB} \cdot \log\left(\frac{\|D\|_2^2}{\|D-\widehat{D}\|_2^2}\right) \tag{9}$$

- Figure of Merit (FOM): The SNDR measures the quality only. But there is another performance measurement to be compared and it's not less importance than SNDR especially in implantable devices. This parameter is the hardware complexity (area or power). Both can be used to express the hardware complexity. In this thesis, area is used as an indicator on the hardware complexity. Area (Power) cannot be used alone, because it can be decreased at the expense of output Latency and vice versa by changing the level of pipelining. So that the hardware block latency should be included, as well, in the consideration to be fair comparison, as shown in Eq. (10).

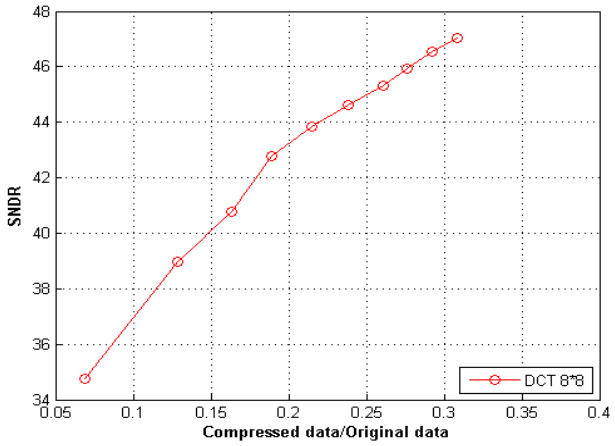
$$\mathrm{FOM} = 10\mathrm{dB}\,.\,\frac{\log\left(\frac{\|D\|_2^2}{\|D-\hat{D}\|_2^2}\right)}{\mathrm{Latency}\ .\ \mathrm{Area}} \tag{10}$$

Figure 3-15 shows the SNDR for all previous compression algorithms for different compression ratios.

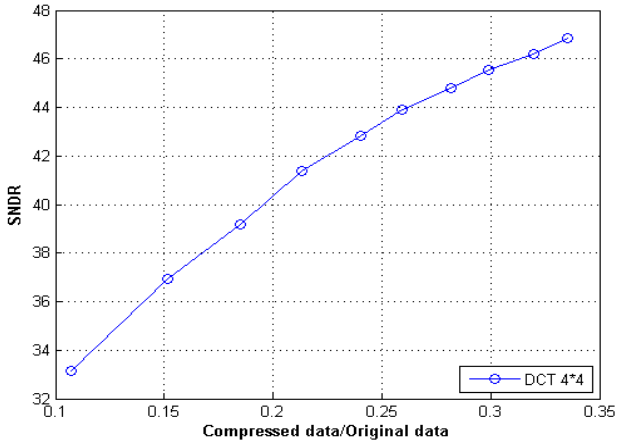
a) 8x8 2D-DCT
b) 4x4 2D-DCT
c) 8x8 DF-2D-DCT
d) 4x4 DF-2D-DCT
e) 2D-DWT

Figure 3-16 shows the FOM for all previous compression algorithms for different compression ratios.

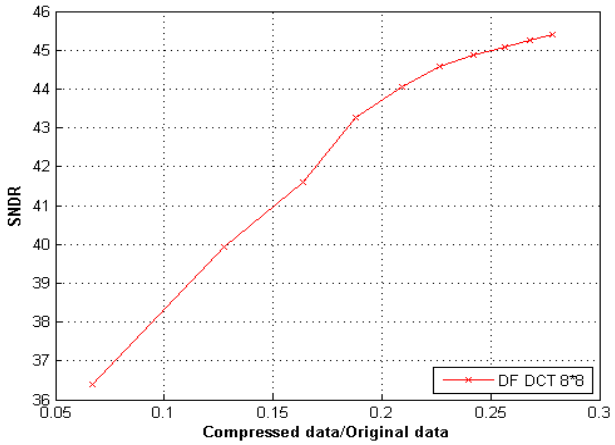
a) 8x8 2D-DCT
b) 4x4 2D-DCT
c) 8x8 DF-2D-DCT
d) 4x4 DF-2D-DCT
e) 2D-DWT


SNDR
Compressed data/Original data
DCT 8*8

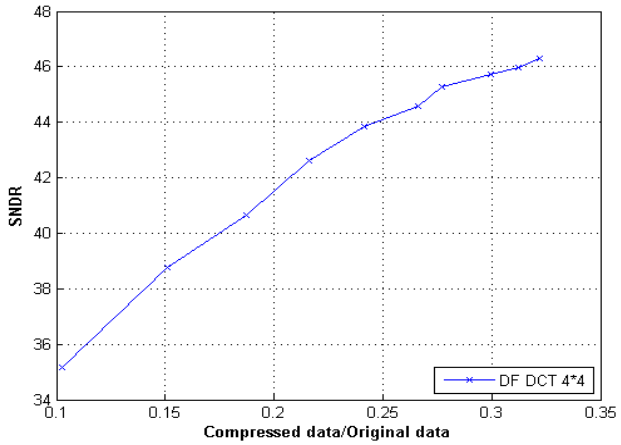
(a)


SNDR
Compressed data/Original data
DCT 4*4

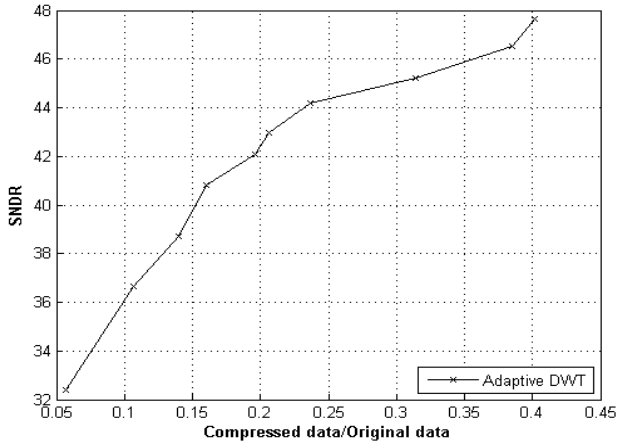
(b)


SNDR
Compressed data/Original data
DF DCT 8*8

(c)

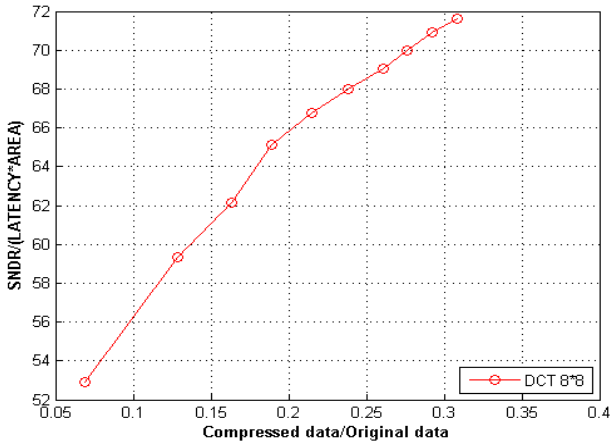


(d)

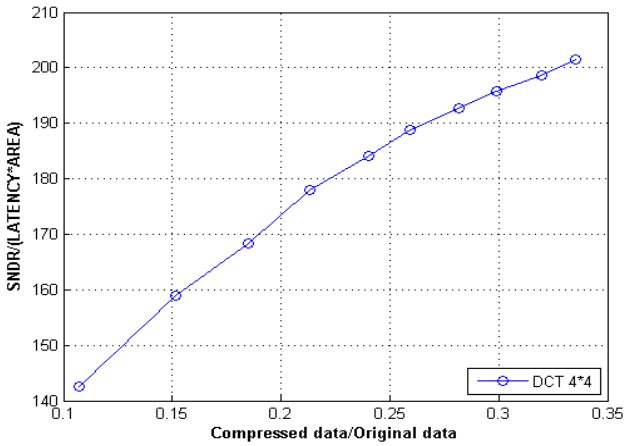


(e)

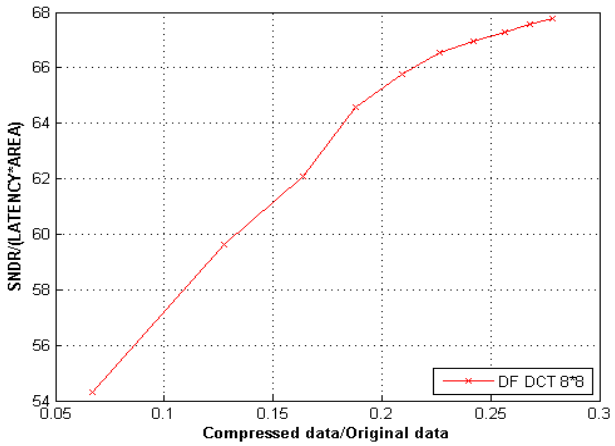
Figure 3-15: SNDR vs compression ratios for all algorithms


SNDR/(LATENCY*AREA)
Compressed data/Original data
DCT 8*8

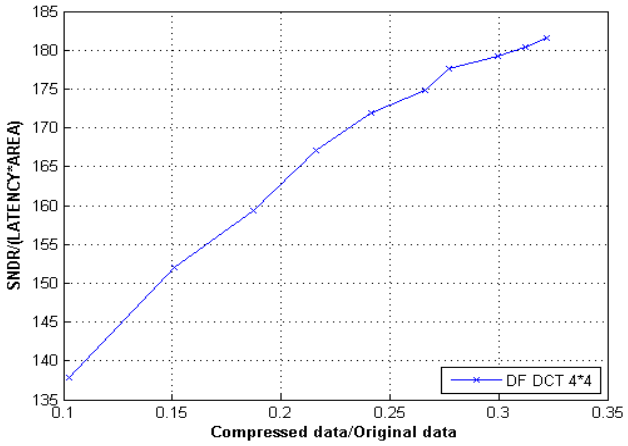
(a)


SNDR/(LATENCY*AREA)
Compressed data/Original data
DCT 4*4

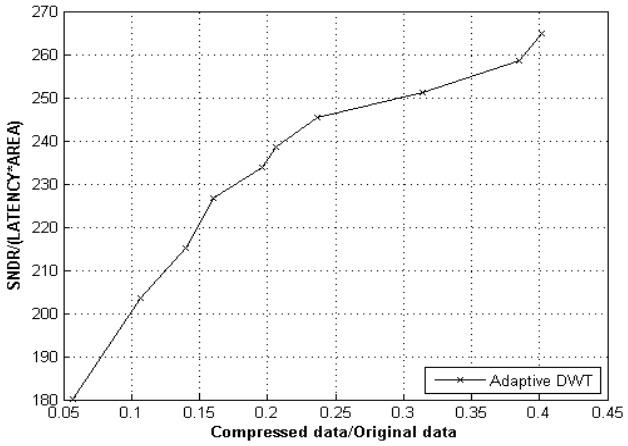
(b)


SNDR/(LATENCY*AREA)
Compressed data/Original data
DF DCT 8*8

(c)



(d)



(e)

Figure 3-16: SNDR/(LATENCY*AREA) vs compression ratios for allalgorithms

Figure 3-17.a shows the signal-to-noise and distortion ratio (SNDR) for all proposed compression algorithms for different compression ratios. Spatiotemporal compression algorithms whether DF_DCT_8*8 or DF_DCT_4*4 consistently achieve higher SNDR than spatial only algorithms for a wide range of compression ratios. Because spatiotemporal algorithms utilize the temporal correlation between consistence frames, but spatial 2D algorithms whether DCT_8*8 or DCT_4*4 utilize the spatial correlation between adjacent channels only.
In addition, it's obvious that DCT_8*8 achieve better performance than DCT_4*4 duo to larger block size. Block size 8*8 utilizes the correlation between 64 channels but 4*4 block utilize the correlation between 16 channels only. But DCT_8*8 algorithm it's not a perfect compression algorithm to be used in implantable devices duo to its hardware complexity as will be discussed later. Adaptive quantization DWT algorithm achieves better compression performance than DCT_4*4 compression algorithms whether spatial or spatiotemporal for a wide range of compression ratios till SNDR value at or above 45 dB. But it's still lower than DCT_8*8 compression algorithms whether spatial or spatiotemporal.

The above comparison among the compression algorithms is based on the performance (SNDR). Now the hardware implementation area and hardware latency should be included into the comparison to give real design insights. In order to get them, the following steps are conducted:

- The area of the hardware design is measured on 130 nm technology for ASIC implementation. And calculate the needed SRAM memory for every compression algorithm then multiply it by 6-transistor SRAM area for the same technology.
- Hardware latency per frame (1024 channels) is used as a reference in the comparison.
- Area and latency is used normalized to the maximum value.

Table 3-1 lists the performance metrics for the algorithms and the following design insights are extracted from the table:

- DWT algorithm has the smallest area and power, but it has the largest latency per frame.
- DF_DCT_8*8 has the largest area and power (triple of DF_DCT_4*4), but it has the smallest latency per frame.
- The major difference between spatiotemporal algorithms and spatial algorithms is the required memory size.

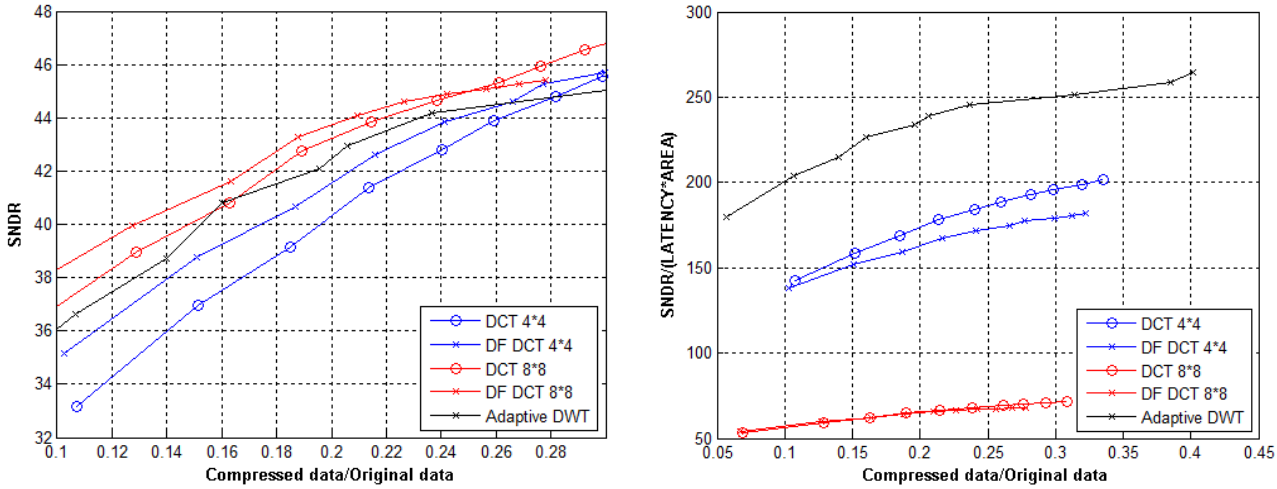
Figure 3-17.b shows the FOM for all previous compression algorithms for different compression ratios. Adaptive DWT compression algorithm is deduced as the best algorithm for all compression ratios duo to its hardware simplicity and all DCT based algorithms is worse for all compression ratios duo to its hardware complexity. In addition, spatial only compression algorithms achieve higher FOM than spatiotemporal

algorithms because they add extra area overhead more than their effect on the SNDR performance.

Finally, the Adaptive Quantization DWT algorithm is recommended as the most suitable compression algorithm for low-power implantable devices for neural data compressing. To reconstruct the data without performance degradation SNDR more than 42 dB needs to be achieved. Then, the size of compressed data will be 19% from the original data size. For the seizure detection, SNDR around 30 dB is adequate. Then, the size of reconstructed data will be 2.5% from the original data size.

	Area (um)	Memory (KB)	Memory (µm)	Total Area (µm)	Power (mw)	Latency Per Frame (clock cycle)
DCT 8*8	726923	0.086	1710	728633	96.94	3280
DF DCT 8*8	721770	1.08	21617	743387	95.62	3280
DWT	110639	1	19906	130545	11.36	5888
DCT 4*4	232939	0.02	427	233366	32.06	3648
DF DCT 4*4	235855	1.02	20334	256189	31.06	3648

Table 3-1: Hardware Performance Summary of the compression algorithm



(a) **(b)**

Figure 3-17: Comparison of the performance of the compression algorithms

Chapter 4 : Harvested Power Adaptive High-Resolution Neural Data Compression(PANDCA)

4.1. Introduction

The main goal of any implantable compression device is to get the smallest data size to be transmitted to the outside world with lowest distortion and data loss at receiver side. In this work, the neural compression algorithm is adapted according to the available harvested power budget. Therefore, the maximum signal to noise and distortion ratio (SNDR) is achieved based on the available harvested power budget without any data loss.

Implantable devices need an efficient power source to supply them with enough energy to power the electrodes, analog interfacing and digital classification as will be described later. Implantable devices are usually powered by a rechargeable battery that is charged by using a micro-scale energy harvesting system. These implantable rechargeable batteries provide the energy for implantable biomedical devices. However, batteries have limited lifetime, fixed energy density, large size and chemical side effects. Thus, researchers have developed various methods to harvest energy for implantable neural devices.

Devices powered by harvested energy provide more safety and comfort and have longer lifetime than conventional devices. Energies that may be scavenged include:

- Thermal energy

  The body temperature changes when it receives or transmits energy. In this situation, the molecules are in constant motion, and this excitation is measured by temperature. Only by temperature difference can extract energy from a thermal reservoir (human body) be guaranteed. The conversion possibility between heat and work has been restricted to thermal machines.

- Solar energy

  Solar energy is the conversion of energy from light into electricity, either using concentrated solar power(indirectly) or using photovoltaics(directly). Concentrated solar power systems use mirrors or lenses and tracking systems to be able to focus a large area of sunlight into a small beam. Photovoltaic (PV) cells convert light into an electric current using the PV effect.

- Body motion energy (Piezoelectric energy)

  Brothers Pierre and Jacques Curie discovered the piezoelectric effect in quartz crystals in 1880. In general, can be defined as the conversion of mechanical energy to electrical energy (direct effect) or conversion of electrical energy to mechanical energy (inverse effect) [21]. The direct piezoelectric effect provides that an electrical charge is generated when it subjected to a mechanical energy,

whether delivered from traction, compression or just vibration. In turn, the inverse piezoelectric effect is the piezoelectric material ability to generate mechanical energy when subjected to an electrical charge in opposite sides [21].

- Gravity energy

- Infrared radiant energy

Energy harvesting devices from their surroundings produce electric energy through direct energy conversion. The energy harvesting from environmental sources or human body has been provided to be an effective alternative [13].

In this work, piezoelectric energy harvesting is proposed, which uses a direct energy conversion from vibrations and mechanical deformation to electrical energy. This is a promising technique to supply power sources in implantable biomedical devices, since it has higher energy conversion efficiency and a simple structure.
Recently, various technologies, such as micro- and macro-mechanics, advanced materials, and electric circuit design, have been emerged and investigated to improve the performance and the conversion efficiency of the piezoelectric energy harvesters. In this work, the focus is on recent progress of piezoelectric energy harvesting technologies based on PbZrTi (PZT) materials, which have the most outstanding piezoelectric properties. The higher output energy density of the (PZT) piezoelectric energy harvester is 231 mW/cm2 [14,15].

In all the transmission rates and power calculations the TI chip CC3100MOD is used as a reference in this work [16]. This chip is a low-power Wi-Fi for Internet of Things (IoT) applications and operates in two modes:

- Standby mode: with current up to 140 μA and average power up to 504 μW.

- Low Power Tx mode: with current up to 223 mA and average power up to 802.8 mW.

The main transmission protocols:

- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)

The TCP is one of the major the Internet protocol suite protocols. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Hence, the entire suite is commonly referred to as TCP/IP. TCP provides error-checked, ordered, and reliable delivery of a stream of bits between applications running on hosts communicating by an IP network. But TCP is a relatively complex protocol, with a lot of features that come at a cost. It imposes some overhead in terms of packet size. The TCP handshake also takes some extra latency compared to simpler protocols. So User Datagram Protocol (UDP) transmission mode is used in this work. This protocol was designed by David Reed in 1980 and formally defined as a main network protocol. With UDP, computer applications can send messages, in this case referred to as neural

compressed data, to other hosts on an Internet Protocol (IP) network. Prior communications are not needed in order to set updata paths or transmission channels. This protocol is a simpler message-based connectionless protocol. Connectionless protocols do not set up a dedicated end-to-end connection. Communication is held by information transmission in one direction from source to destination without waiting to acknowledge from the receiver. The UDP mode is preferred to be used in independent packets transmission such as sound packets and neural data packets. The proposed reference chip achieves a UDP actual throughput up to 16 Mbps.

In this work, a feedback from the power harvesting device is needed to be able to know the input current level (power level), because PANDCA utilizes it as an input, as will be described later. Hence, a current sensor is adopted to detect the input current level. This input is quantized to suitable number of levels according to the power harvesting device. Then, used as an indicator for the available power level to the compression block.

4.2. Motivation

Ordinary implantable biomedical devices have 3 scenarios to utilize the available harvested power budget to transmit the compressed neural data to the outside world:

- First scenario is to transmit the neural compressed data with fixed low compression ratio continuously with low rate to guarantee that the minimum available power is enough to transmit the compressed data over all the time.

- Second scenario is to transmit a defined period of neural data when a predefined triggering event occurs such as seizure spikes if there is enough available harvested power budget to transmit this period to the outside world. Otherwise, this triggering event is discarded if there is not enough harvested power budget to transmit this period.

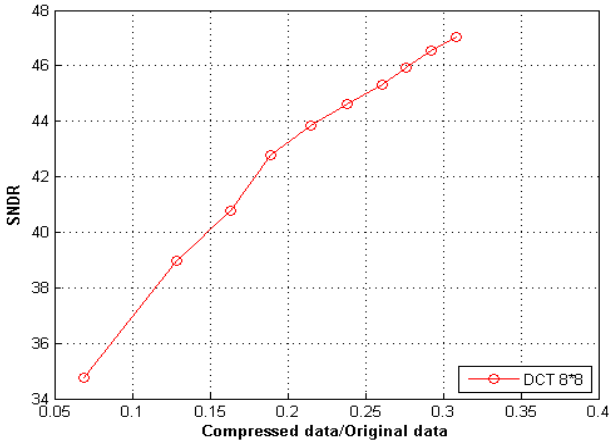
- Third scenario is to send neural data continuously without dependence on any special event with suitable rate as long as there is enough harvested power budget to send it continuously. Otherwise, if there is not enough harvested power budget to continue the transmission, it stops the transmission till producing enough power from harvesting power source then starts the transmission again.

However, all these scenarios are not efficient enough for the current biomedical implantable devices constrains. Current treatment devices need the complete waveform and history for every electrode to be extracted instead of extracting the special signal features only to be able to detect and diagnose neural brain disorders. Accordingly, it should be guaranteed that the detected neural data can be transmitted continuously without any pauses or data loss. In addition, the compressed data can be decompressed at the other side with high quality without significant distortion.

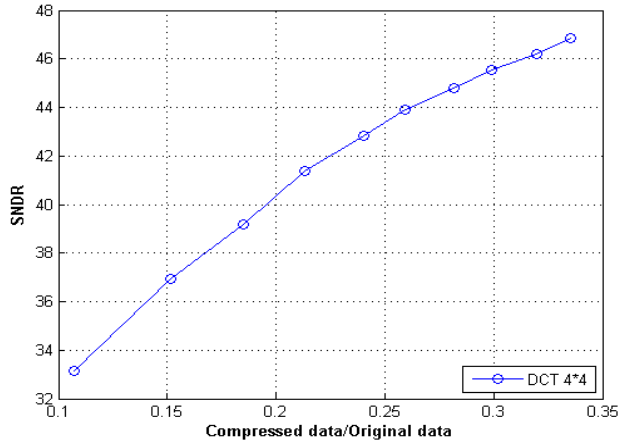
In this work, PANDCA is proposed to use the available harvested power budget to adapt the compression algorithm ratio. Therefore, the proposed technique allows transmitting the compressed neural data continuously. Hence, it achieves maximum signal noise and distortion ratio (SNDR) according to available harvested power budget without any data loss.

4.3. Selected Compression Algorithm

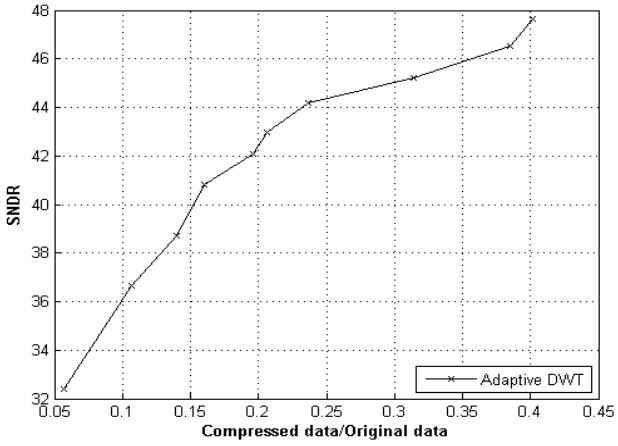
Despite of Adaptive 2D-DWT algorithm is a most suitable solution for low-power implantable devices. But 2D-DCT compression algorithm is selected to be used in PANDCA due to the linearity characteristic in the target region as shown in Figure 4-1. This characteristic is very important in PANDCA to be able to divide it to equal steps, as will be described later.



(a)



(b)



(c)

Figure 4-1: SNDR vs compression ratios for all algorithms

4.4. Power Oriented Algorithm

4.4.1. Neural System Power Components

The main power hungry blocks in neural implantable devices are the wireless transmitters [18] and the data compressors [19]. Thus, the large percentage of the harvested power budget is dedicated to the wireless transmitter and compression block as shown in Figure 4-2.

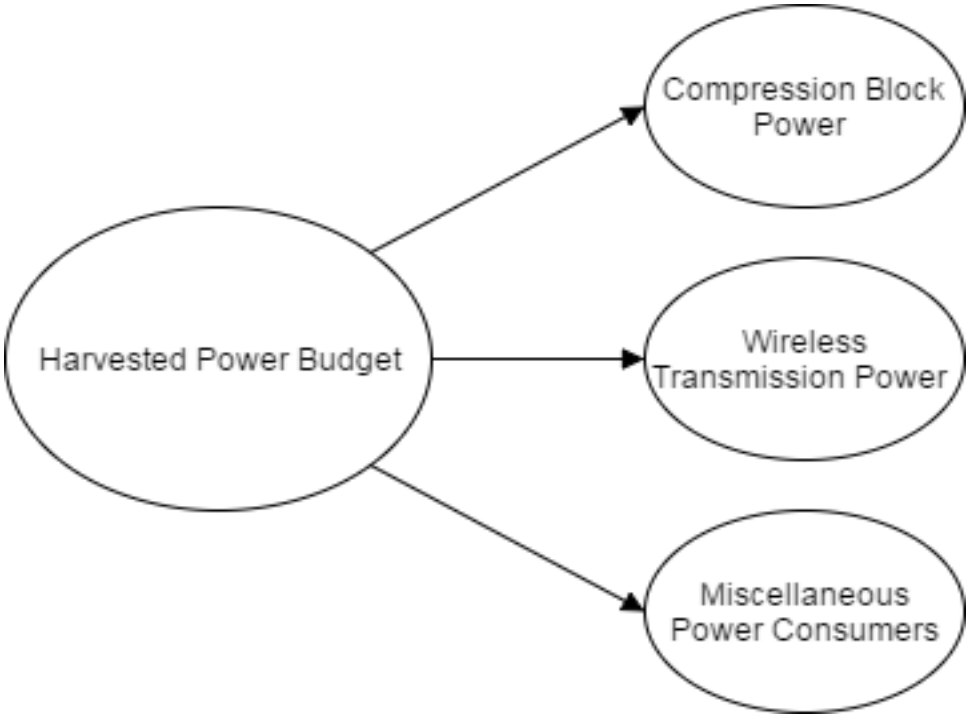


Figure 4-2: Neural system Power Tree

Compression Block Power:

If the number of channels (electrodes) is constant, compression block consumes constant power regardless of the compression ratio. After RTL Hardware Implementation on 130nm technology for ASIC implementation, PANDCA consumes 32.06 mW. This power consumption is drawn from the power budget as a constant value and the remaining power budget is used by the wireless transmitter.

Miscellaneous Power Consumers:

Neural Implantable devices have a lot of blocks which consume power such as electrodes, analog interfacing, and digital controllers. However, the main characteristic which combines them is that they consume fixed power regardless any change in

quality factor. So this power consumption is drawn from the power budget too as a constant value and the remaining power budget is used by the wireless transmitter.

Wireless Transmission Power:

Low power WIFI chips support data rates up to 16 Mbps. The TI chip CC3100MOD in transmission only mode is always sleep except when there is available data to be transmitted. In sleep mode, the absorbing current consumption is very small compared to transmission mode so that it can be negligible.

Therefore, the power consumption duration is in the TX mode duration only. Hence, the size of data to be transmitted is the main parameter in power consumption. When this data size increases, the duration of Tx mode increases, so is the current (power) consumption. On the other hand, when this data size decreases, the duration of the Tx mode decreases so is the current (power) consumption as shown in Figure 4-3.

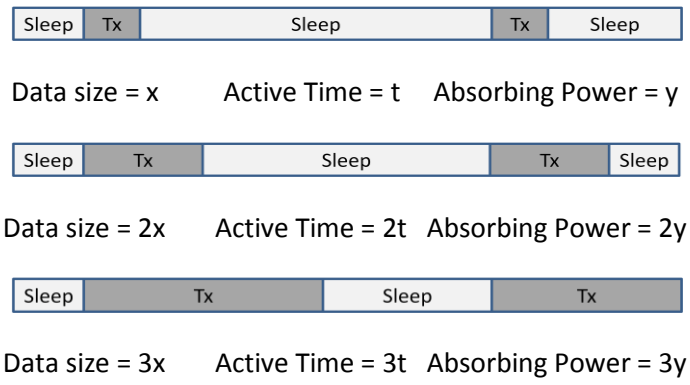


Figure 4-3: Transmission Timing Scheduling

As shown in (7), absorbing current (power) is linearly proportional with time.

$$P = VI = V\left(\frac{V}{R}.t\right) = \left(\frac{V^2}{R}\right).t \quad (7)$$

In this work, data size is assumed to be linearly proportional to the duration of transmission (absorbed Power), assuming that the traffic is idle, especially because TI WIFI chip is used in UDP mode [16]. Hence, absorbing power can be controlled according to compressed data size.

4.4.2. Quality Factor Effect

In the proposed compression algorithm, the quality of compressed data (SNDR) is controlled according to quality factor which varies from 1 to 10. This range is selected because SNDR is approximately linearly proportional with quality factor in this range only, as shown in Figure 4-4 and this characteristic is important in the proposed PANDCA as will be analyzed. Hence, there are 10 quality levels and according to this quality factor change, the compressed data size changes.

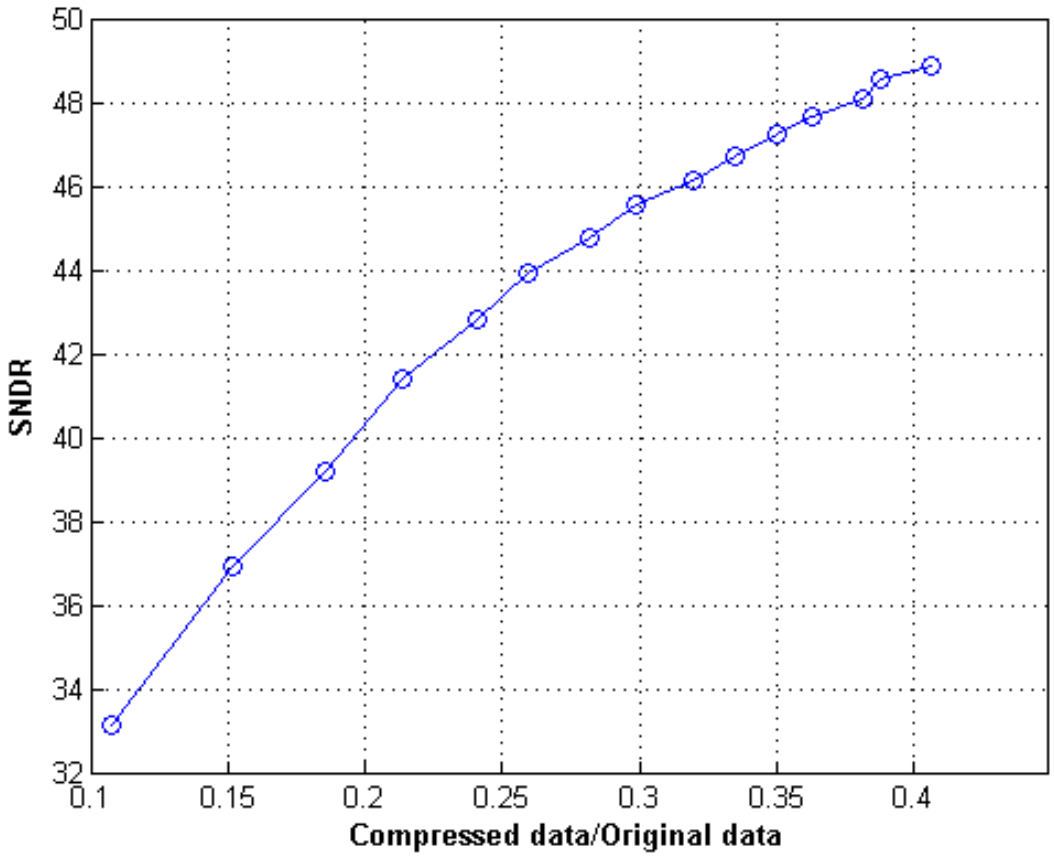


Figure 4-4: Quality Factor Effect

As shown in Figure 4-4:

- When quality factor increases, the compression ratio (compressed data size divided by original data size) increases and vice versa.
- When Quality Factor increases, the SNDR increases and vice versa.

Hence, the output compressed data size and its quality (SNDR) are controlled according to quality factor level.

4.4.3. Power Oriented Algorithm

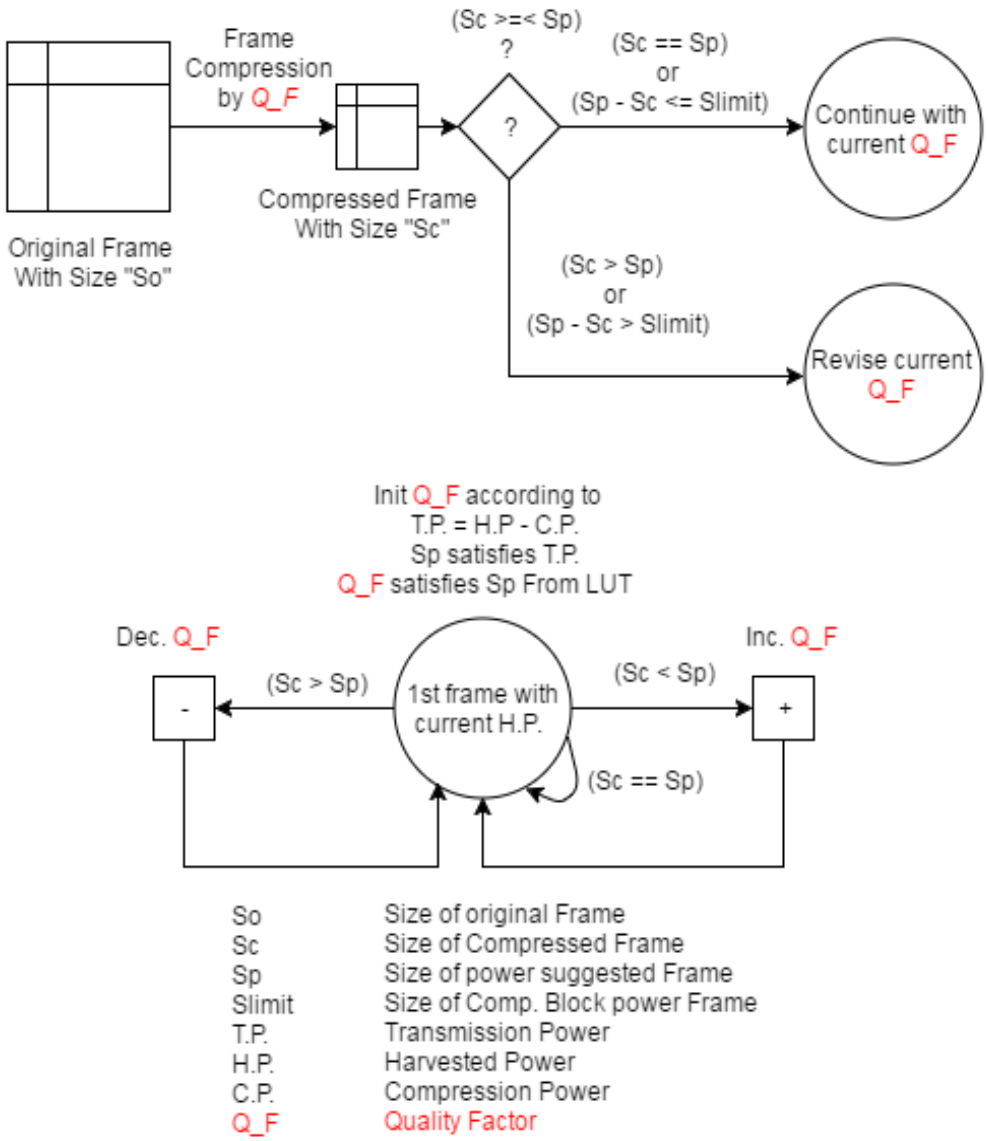


Figure 4-5: Power Oriented Design

As shown in Figure 4-5, this is the proposed algorithm to control the compression algorithm quality factor (compressed data size) according to available harvested power budget:

1. Subtract the needed power for the compression block and other needs from the total available harvested power budget to get the available power budget to transmit the compressed data.
2. Select the size of compressed data which can be transmitted with this available power (according to WIFI chip specifications).
3. Calculate the needed compression ratio (needed size of compressed data / size of original data).

4. Select the initial quality factor from the saved data in Table 4-1, this data is obtained from the previous results with the same correlation, resolution and number of electrodes (channels). This table should be recalculated if any parameter is changed.
5. If the actual compressed data size is equal to, or less than, the suggested compressed size with a specific limit, the compression algorithm should continue with the same quality factor in the next frames.
6. If it is larger than the calculated size or less with a specific limit, the compression algorithm will increment or decrement the quality factor level with quantized steps according to the error step size.
7. The selected quality factor will be in use until the power harvested budget is changed. Once the available harvested power budget is changed repeat again from step 1.

4.5. Results and Discussions

The (64-channel) results are discussed and explained in details. Then, the other channel resolutions results are provided and compared to (64-channel) results.

Table 4-1 shows the suggested initial saved values to start with at step 4 according to available power budget to transmit. Then, go up and down in the next frames according to error step, this table is for (64-channels) results:

- Column 1 divides the quality factors to 10 levels from 1 to 10, when the quality factor increases, the SNDR increases and the compressed data size increases.
- Column 2 is the compressed frame sizes according to quality factor level. It is calculated after hundreds of trials on the brain neural data.
- Column 3 is the frame (64-channel) sampling rate if the channel (electrode) rate is 20 Ksps.
- Column 4 is the needed transmission duration per one second with transmission rate 16 Mbps.
- Column 5 is the needed transmission power if the sleep duration is ignored according to the reference TI WIFI chip CC3100MOD transmission current and voltage.

All these values should be changed if any parameter from electrodes resolution, electrodes correlation, electrode sampling rate, number of channels, transmission rate or WIFI chip is changed.

Figure 4-6 shows four cases of available harvested power budget profile and the performance of the proposed PANDCA based on the harvested power scenario.

Constant harvesting power profile (Figure 4-6.a):

In this case, the available harvested power budget to transmit is 90 mW and it is fixed on this value over all the duration.

The proposed RANDCA searches on the nearest power entry on the saved table (Table 4-1), and selects entry number 6 as an initial value, because it is the lower nearest entry from available power.

To achieve this target power of 88.3 mW, the proposed compression algorithm needs to be adapted initially to quality factor level 5, to get compressed frame size around (11 B), frame rate around (220 Kbps) and transmission duration around .11 second to get the target power of 88.3 mW.

After starting with quality factor level 5, a compressed size (11 B), and this value is lower than the suggested size acceptable range, then it increases the quality factor to level 6 at the next frame trying to enter this acceptable range.

In the second frame, a compressed size (12 B), and this value is more than suggested size acceptable range, then it decreases the quality factor to level 6 again at the next frame trying to enter this acceptable range. This continues till entering the acceptable range and settles the quality factor level or continues in trying mode around the acceptable range.

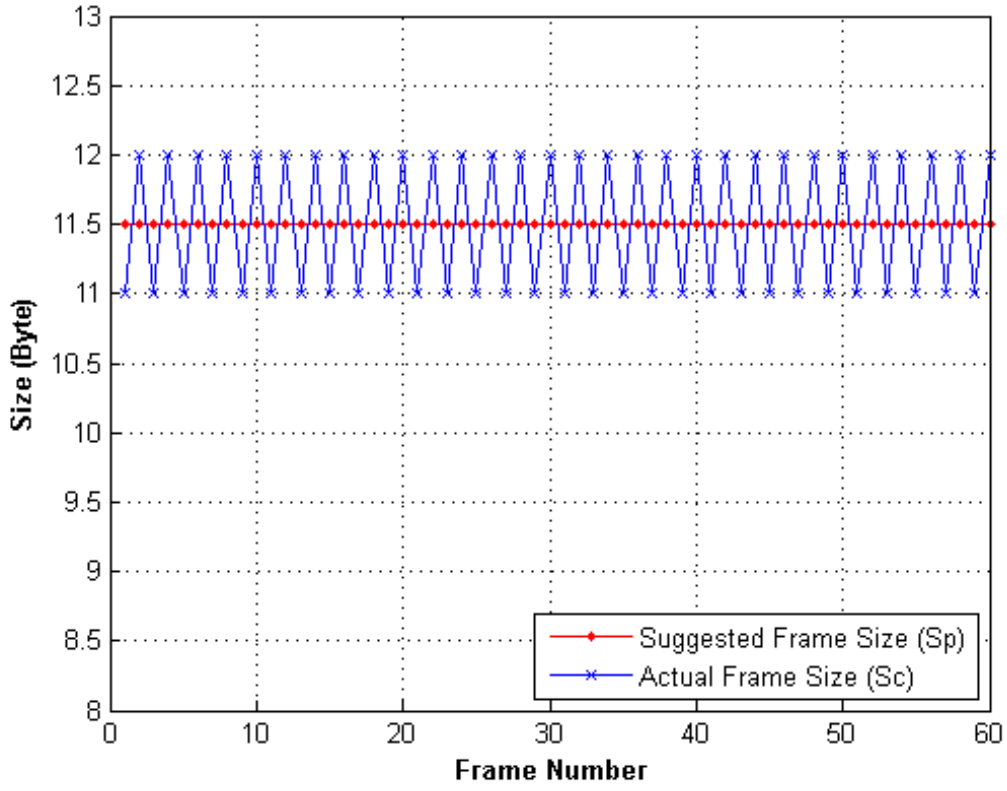


Figure 4-6.a: Constant Power Adaptive Performance

Increasing harvesting power profile (Figure 4-6.b):

In this case, the available harvested power budget to transmit at the first frame is 60 mW and it is increasing linearly over all the duration.

The proposed PANDCA searches on the nearest power entry on the saved table (Table 4-1), and selects entry number 2 as an initial value, because it is the lower nearest entry from available power.

To achieve this target power of 56.2 mW , the proposed compression algorithm needs to be adapted initially to quality factor level 2, to get compressed frame size in around (7 B), frame rate around (140 Kbps) and transmission duration around .07 second to get the target power of 56.2 mW.

After starting with quality factor level 2, a compressed size (7 B), and this value is more than suggested size acceptable range, then the quality factor is decreased to level 1 at the next frame trying to enter this acceptable range.

In the second frame, the proposed compression algorithm faces increase in the available harvesting power level, then the new comparison will be against larger suggested power due to this increase, but this increase can be handled by increasing the quality factor level trying to enter this acceptable range. This continues till entering the acceptable range and settles the quality factor level or continues in trying mode around the acceptable range.

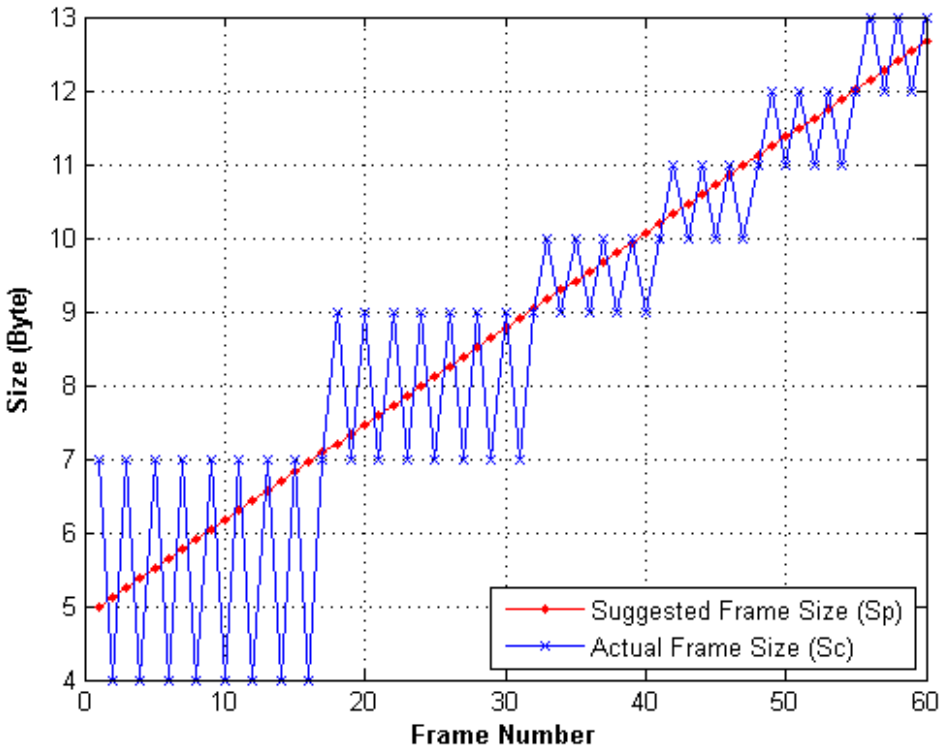


Figure 4-6.b: Increasing Power Adaptive Performance

Decreasing harvesting power profile (Figure 4-6.c):

In this case, the available harvested power budget to transmit at the first frame is 90 mW and it is decreasing linearly over all the duration.

The proposed PANDCA searches on the nearest power entry on the saved table (Table 4-1), and selects entry number 11 as an initial value, because it is the lower nearest entry from available power.

To achieve this target power of 88.3 mW , the proposed compression algorithm needs to be adapted initially to quality factor level 5, to get compressed frame size in around (11 B), frame rate around (220 Kbps) and transmission duration around .11 second to get the target power of 88.3 mW.

After starting with quality factor level 5, a compressed size (11 B), and this value is less than suggested size acceptable range, then the quality factor is increased to level 12 at the next frame trying to enter this acceptable range.

In the second frame, the proposed compression algorithm faces decrease in the available harvesting power level, then the new comparison will be against lower suggested power due to this decrease, but this decrease can be handled by decreasing the quality factor level trying to enter this acceptable range. This continues till entering the acceptable range and settles the quality factor level or continues in trying mode around the acceptable range.

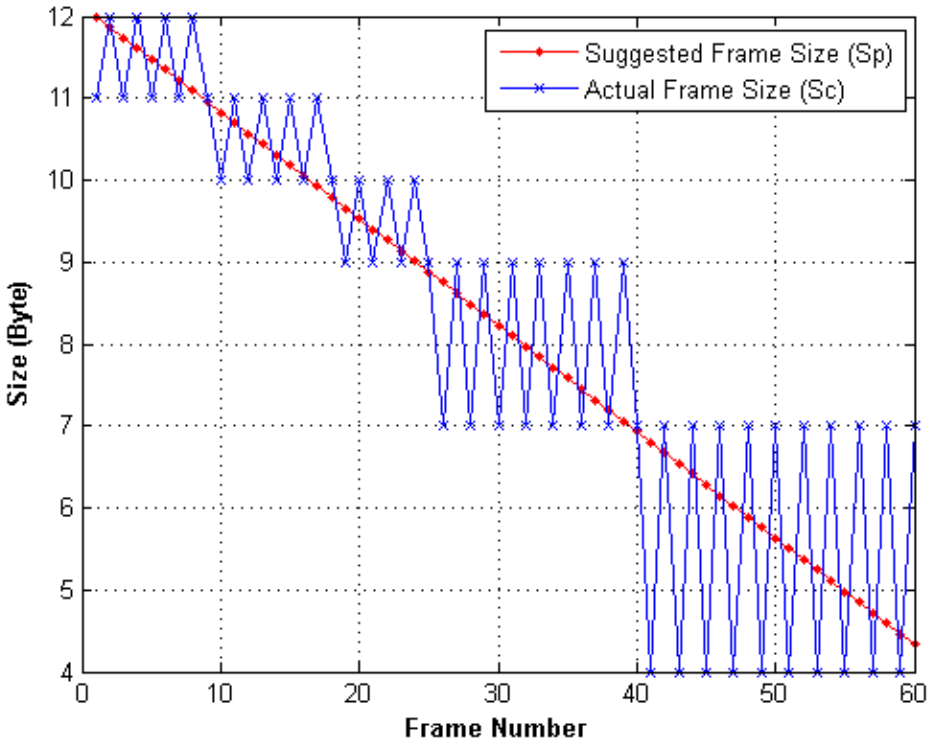


Figure 4-6.c: Decreasing Power Adaptive Performance

Real harvesting power profile (Figure 4-6.d):

In this case, the available harvested power budget to transmit at the first frame is 90 mW and it is decreasing linearly over all the duration.

The proposed PANDCA searches on the nearest power entry on the saved table (Table 4-1), and selects entry number 11 as an initial value, because it is the lower nearest entry from available power.

To achieve this target power of 88.3 mW , the proposed compression algorithm needs to be adapted initially to quality factor level 5, to get compressed frame size in around (11 B), frame rate around (220 Kbps) and transmission duration around .11 second to get the target power of 88.3 mW.

After starting with quality factor level 5, a compressed size (11 B), and this value is less than suggested size acceptable range, then the quality factor is increased to level 12 at the next frame trying to enter this acceptable range.

After few frames, the proposed compression algorithm faces decrease in the available harvesting power level, then the new comparison will be against lower suggested power due to this decrease, but this decrease can be handled by decreasing the quality factor level trying to enter this acceptable range. This continues till entering the acceptable range and settles the quality factor level or continues in trying mode around the acceptable range.

After few frames, the proposed compression algorithm faces increase in the available harvesting power level, then the new comparison will be against larger suggested power due to this increase, but this increase can be handled by increasing the quality factor level trying to enter this acceptable range. This continues till entering the acceptable range and settles the quality factor level or continues in trying mode around the acceptable range, as shown in Figure 4-6.d.

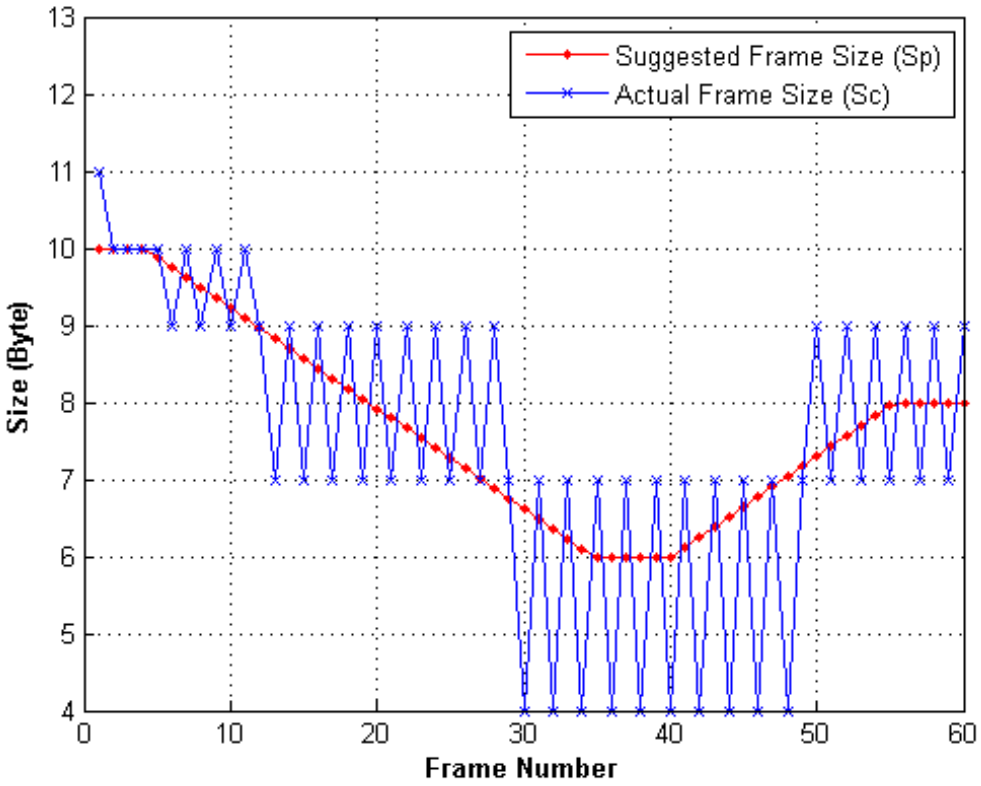


Figure 4-6.d: Real Power Adaptive Performance

As shown on these four case studies, the proposed (PANDCA) achieves the highest possible SNDR based on the available harvested power budget. In Table 4-2, the comparison between conventional algorithm which compresses the neural data with fixed compression ratio to be able to produce a suitable compressed data size to be transmitted to the outside world without any discontinuity against the proposed PANDCA. It is obvious that there is a significant enhanced performance of the proposed algorithm compared to the conventional algorithm especially in cases (b, c and d) because the harvested power is variable with time and conventional algorithms are not adaptive to these cases. Knowing that normally in the implantable devices for neural data compression, the harvested energy exhibits different profiles based on the environmental conditions

Table 4-3 and Table 4-4 show the results for two other model sizes (32-channels) and (8-channels) respectively with high resolution grid too, these small model sizes are compressed with 2D-DCT as well but with 4x4 block size.

Finally, the harvested power adaptive high-resolution neural data compression algorithm (PANDCA) is the most suitable compression algorithm for low-power implantable devices for neural data compressing. To reconstruct the data without performance degradation, higher possible SNDR over all the time should be achieved and the only obstacle to achieve that is the available harvested power.

Q_F Index	Sp (Byte)	Frame rate (KBps) ch. Rate = 20 Ksps	Tx. Duration per (1s) with rate 16 Mbps (s)	Tx. Power (mW)
1	4	80	0.04	32.112
2	7	140	0.07	56.196
3	9	180	0.09	72.252
4	10	200	0.1	80.28
5	11	220	0.11	88.308
6	12	240	0.12	96.336
7	13	260	0.13	104.364
8	14	280	0.14	112.392
9	15	300	0.15	120.42
10	16	320	0.16	128.448

Table 4-1: 64-channel results

	Constant Power (case a)		Increasing Power (case b)		Decreasing Power (case c)		Real Power (case d)	
	Avg. SNDR	Number of Tx Bytes per 60 Frame	Avg. SNDR	Number of Tx Bytes per 60 Frame	Avg. SNDR	Number of Tx Bytes per 60 Frame	Avg. SNDR	Number of Tx Bytes per 60 Frame
Normal Compression Algorithms	40	660	34	240	34	240	34	360
Power Adaptive Algorithm	41	690	38	520	36	492	38	451

Table 4-2: Performance Comparison

Q_F Index	Sp (Byte)	Frame rate (KBps) ch. rate = 20 Ksps	Tx. Duration per (1s) with rate 16 Mbps (s)	Tx. Power (mW)
1	2	40	0.02	16.056
2	4	80	0.04	32.112
3	4.5	90	0.045	36.126
4	5	100	0.05	40.14
5	5.5	110	0.055	44.154
6	5.6	112	0.056	44.9568
7	5.7	114	0.057	45.7596
8	6	120	0.06	48.168
9	6.5	130	0.065	52.182
10	7.5	150	0.075	60.21

Table 4-3: 32-channel results

Q_F Index	Sp (Byte)	Frame rate (KBps) ch. rate = 20 Ksps	Tx. Duration per (1s) with rate 16 Mbps (s)	Tx. Power (mW)
1	1	20	0.01	8.028
2	1.25	25	0.0125	10.035
3	2	40	0.02	16.056
4	2.1	42	0.021	16.8588
5	2.25	45	0.0225	18.063
6	2.5	50	0.025	20.07
7	2.75	55	0.0275	22.077
8	3	60	0.03	24.084
9	3.25	65	0.0325	26.091
10	3.5	70	0.035	28.098

Table 4-4: 16-channel results

4.6. Summary

Neural data research has a wide application today and it heavily depends on data compression to be able to extract all signal waveforms with finer resolution for further processing. This work proposes a harvested power adaptive high-resolution neural data compression (PANDCA) as the most suitable compression algorithm candidate to achieve the highest possible SNDR based on available harvested power budget without any data loss or discontinuity in the transmission to the outside world.

Chapter 5 Conclusions and Future Work

5.1. Discussion and Conclusions

This Thesis is divided to two main parts:

- A Low-Power Area-Efficient Design and Analysis for Neural Data Compression.
- Harvested Power Adaptive High-Resolution Neural Data Compression (PANDCA).

In the first part, five compression algorithms are proposed, investigated and compared, to utilize both spatial correlation between adjacent electrodes and temporal correlation between consecutive samples. These algorithms are:

a) 8x8 2D-DCT
b) 4x4 2D-DCT
c) 8x8 DF-2D-DCT
d) 4x4 DF-2D-DCT
e) Adaptive 2D-DWT

These proposed algorithms' performance, area, time and power are evaluated and compared to provide the best trade-off between hardware complexity and compression performance.
The conclusion from performance (SNDR) comparison among the compression algorithms is that the spatiotemporal algorithm (DF-DCT 8x8) is the best algorithm for most of compression ratios.
However, after hardware implementation area and hardware latency is included into the comparison to give real design insights. Adaptive DWT compression algorithm is deduced as the best algorithm for all compression ratios duo to its hardware simplicity. On the other hand, all DCT based algorithms is worse for all compression ratios duo to its hardware complexity. In addition, spatial only compression algorithms achieve higher FOM than spatiotemporal algorithms because they add extra area overhead more than their effect on the SNDR performance.
Finally, the Adaptive Quantization DWT algorithm is recommended as the most suitable compression algorithm for low-power implantable devices for neural data compressing. To reconstruct the data without performance degradation SNDR more than 42 dB needs to be achieved. Then, the size of compressed data will be 19% from the original data size. For the seizure detection, SNDR around 30 dB is adequate. Then, the size of reconstructed data will be 2.5% from the original data size.

In the second part, the goal is to get the smallest data size to be transmitted to the outside world with lowest distortion and data loss at receiver side. A new methodology is introduced to control the compression algorithm according to available harvested

power. Hence, maximum signal to noise and distortion ratio (SNDR) is achieved based on the available harvested power without any data loss.
Despite of Adaptive 2D-DWT algorithm is a most suitable solution for low-power implantable devices. But 2D-DCT compression algorithm is selected to be used in harvested power adaptive algorithm due to the linearity characteristic in the target region. And this characteristic is very important in the harvested power adaptive algorithm to be able to divide it to equal steps.
The PANDCA algorithm can achieve the highest possible SNDR based on available harvested power. After comparison between ordinary algorithm which compress with fixed compression ratio which can produce a suitable compressed data size to be transmitted to outside world without any discontinuity and PANDCA. A significant effect of PANDCA than the ordinary algorithm is deduced especially in case of unstable power source because the harvesting power is variable with time and ordinary algorithms can't handle these cases.
Finally, the power harvesting oriented high-resolution neural data compression algorithm is the most suitable compression algorithm for low-power implantable devices for neural data compressing. To reconstruct the data without performance degradation, higher possible SNDR over all the time should be achieved and the only obstacle to achieve that is the available harvested power.

5.2. Future Work

5.2.1. FPGA Demo Implementation

Figure1-1 shows the full system architecture of the implantable neural measurement system. Neural signals recorded from the multielectrode array will be amplified in the analog front-end (AFE) and converted to digital neural data using analog to digital (ADC) block. Subsequently, the neural data runs into the main digital module where data compression and signal processing take place. Hence, the low-rate wireless transceiver transmits the compressed neural data to the outside world (reconstruction base station), where signal reconstruction and decompression are performed. Since the system is fully implantable, energy has to be harvested using PZT harvesting system.

Most of digital blocks include (electrode neural data combining, compression algorithm and compressed data streaming) are ready. Once the remaining blocks:

- Analog blocks (AFE and ADC)
- Low-rate wireless transceiver (TI chip and its software application)
- PZT Harvesting device

are available. Then, this work can be hardware implemented using FPGA and some peripherals.

5.2.2. Neural Data Encryption before compression

Neural data should be secure and protected from any hacking because any interrupt in patient neural data from outside world may lead to wrong stimulation or seizure detection failure. Also neural data should be secret and no parasitical can analyze it except the specialized doctor. Therefore, firewalls, including security check protocols, security measurement, restricted network access and data encryption should be seriously considered to avoid corruption of the secret patient data. There are a lot of low power security algorithms can be used. However, the main milestone is the power consumption because security algorithms are power hungry.

5.2.3. Adaptive 2D_DWT Performance Linearization

Despite of Adaptive 2D-DWT algorithm is a most suitable solution for low-power implantable devices. But 2D-DCT compression algorithm is selected to be used in PANDCA due to the linearity characteristic in the target region as shown in Figure 3-1. This characteristic is very important in PANDCA to be able to divide it to equal steps, as described previously. Hence, Adaptive 2D-DWT algorithm needs to be linearized to be applicable with PANDCA. Hence, PANDCA can gain better performance (SNDR) than current version.

5.2.4. Real High Resolution Neural Data Simulation

To evaluate the performance of the compression algorithms for high resolution neural data, virtual recorded data is used with the same signals characteristics (spatial correlation and temporal correlation) of real data, because there is no available high resolution recorded data with these large sizes yet (1024 channels and more). So that, the proposed algorithms need to be re-simulated again once the real high resolution neural data be available to be confirmed.

References

1. T. Kim, N. S. Artan, J. Viventi, and H. J. Chao, "Spatiotemporal Compression for Efficient Storage and Transmission of High-Resolution Electrocorticography Data", Annual International Conference of the IEEE Engineering in Medicine and Biology Society (in submission), 2012.

2. P. Chen and J. Chang, "An Adaptive Quantization Scheme for 2-D DWT Coefficients", 2013.

3. S. Schmale, J. Hoeffmann, B. Knoop, G. Kreiselmeyer, H. Hamer, D. Peters-Drolshagen and S. Paul, "Exploiting Correlation In Neural Signals For Data Compression", 2014.

4. Zarlink Semiconductor (now: Microsemi), "ZL70102-Medical implantable RF transceiver MICS RF telemetry," June 2010.

5. B. Usevitch, "A tutorial on modern lossy wavelet image compression:foundations of JPEG 2000", Signal Processing Magazine, IEEE vol. 18, 2001.

6. C. Chung, L. Chen, Y. Kao and F. Jaw, "Multichannel Evoked Neural Signal Compression Using Advanced Video Compression Algorithm", 2009.

7. K. Rao and P. Yip,Discrete, "Cosine Transform-Algorithms, Advantageand Applications". Academic Press Limited, 1989.

8. A. clanku, P. Jana, "Edge detection in medical images using the Wavelet Transform", July 2011.

9. R. Dqueiroz, "Wavelet transforms in a JPEG-like image coder", IEEE Trans. on Circuits and Systems for Video Technology, vol. 7, no. 2, pp. 419-424, 1997.

10. K. Cabeen, P. Gent, "Image Compression and the Discrete Cosine Transform", 1998.

11. L. Fahrmeir, R. K̈unstler, I. Pigeot, and G. Tutz, "Statistik: Der WegzurDatenanalyse", 2007.

12. S. Schmale, B. Knoop, J. Hoeffmann, D. Peters-Drolshagen, and S. Paul, "Joint compression of neural action potentials and local field potentials," 47th AsilomarConference on Signals, Systems and Computers,Nov. 2013.

13. M.Billinghurst andT.Starner, "Wearable devices. New ways to manage information", IEEE Journals and Magazines (Computer), 1999.

14. M. Kang, W. Jung, "Recent Progress on PZT Based Piezoelectric EnergyHarvesting Technologies", 2016.

15. C.Choia and I.Seoa,"Relation between piezoelectricproperties of ceramics and output power density of energy harvester", 2011.

16. http://www.ti.com/product/CC3100MOD

17. N.Logothetis, "The underpinning of the BOLD functional magnetic resonance imaging signal", The Journal of Neuroscience, 2003.

18. A. Yakovlev and S. Kim, "Implantable Biomedical Devices: Wireless Powering and Communication", April 2012.

19. M. Ashraf and H. Mostafa, “A low-power area-efficient design and comparative analysis for high-resolution neural data compression”, Dec. 2017.

20. F. Casimiro andP. Gaspar "Aplicação do princípiopiezoeléctrico no desenvolvimento de pavimentos para aproveitamentoenergético", In III Conferência Nacional emMecânica de Fluidos, Termodinâmica e Energia: MEFTE - 2009, Bragança, Setembro, 2009.

21. M.Asenwi, T. Ismail, and H. Mostafa, "Performance Analysis of Hybrid Lossy/Lossless Compression Techniques for EEG Data", IEEE International Conference on Microelectronics (ICM 2016), Cairo, Egypt, IEEE, pp. 1-4, 2016.

22. S. Fauvel, “An energy efficient compressed sensing framework for the compression of electroencephalogram signals”, Sensors, vol. 14, no. 1, pp. 1474–1496, 2014.

23. G. Antoniol and P. Tonella, “Eeg data compression techniques”, Biomedical Engineering, IEEE Transactions on, vol. 44, no. 2, pp. 105–114, 1997.

24. A. Deshlahra, G. Shirnewar, and A. Sahoo, “A comparative study of dct, dwt & hybrid (dct-dwt) transform”, 2013.

25. S. Akhter and M. Haque, “Eeg compression using run length encoding,” in Signal Processing Conference, 2010 18th European. IEEE, 2010.

26. D. Birvinskas and I. Jusas, “Fast dct algorithms for eeg data compression in embedded systems”, Computer Science and Systems, vol. 12, no. 1, pp. 49–62, 2015

27. B. A. Rajoub, “An efficient coding algorithm for the compression of eeg signals using the wavelet transform,” Biomedical Engineering, IEEE Transactions on, vol. 49, no. 4, pp. 355–362, 2002.

28. K. G. Oweiss, A. Mason, Y. Suhail, K. Thomson, and A. Kamboh, “A scalable wavelet transform VLSI architecture for real-time signal processing in mutichannel cortical implants”, IEEE Trans. On Circuits and Systems,June 2007.

29. S.Majerus, S. Garverick, M. Suster, P. Fletter andM. Damaser,“Wireless, ultra-low-power implantable sensor for chronic bladder pressure monitoring. J. Emerg. Technol” Comput. Syst, 2012.

30. A. Cheng and L.Tereshchenko, “Evolutionary innovations in cardiac pacing”, J. Electrocardiol, 2011.

31. R.Kramme, K.Hoffmann “Springer Handbook of Medical Technology”, 2012.

32. S.Boveda, S. Garrigue andP.Ritter, “The History of Cardiac Pacemakers and Defibrillators”, Italy, 2013.

33. E. Pararas, D. Borkholder and J. Borenstein, “Microsystems technologies for drug delivery to the inner ear”, Adv. Drug Deliv. Rev, 2012.

34. S. Akbari,H. Shea, “An array of 100 μm × 100 μm dielectric elastomer actuators with 80% strain for tissue engineering applications”, 2012.

35. J. Emerg, “Early history and challenges of implantable electronics”,Comput. Syst. 2012, 8, 1–9.

36. C.Occhiuzzi, G.Contri andG. Marrocco,“Design of implanted RFID tags for passive sensing of human body” the STENTag. Antennas Propag., IEEE Trans, 2012.

37. M. Nkosi,F. Mekuria and S.Gejibo, "Challenges in Mobile Bio-Sensor Based mHealth Development" In Proceedings of the 13th IEEE International Conference on e-Health Networking Applications and Services, Columbia, MO, USA, June, 2011.

38. K. Bazaka, R. Crawford, E. Nazarenko andE. Ivanova, "Bacterial Extracellular Polysaccharides", 2011.

39. H. Tao, S. Hwang, M. Liu, B.Panilaitis, M.Brenckle, D. Kaplan, R. Averitt, J. Rogers andF.Omenetto, "Fully Implantable and Resorbable Metamaterials" In Proceedings of the 2012 Conference on Lasers and Electro-Optics, San Jose, CA, USA, May 2012.

40. D. Kim, Y. Kim, J.Amsden, B. Panilaitis, D. Kaplan, F.Omenetto, M. Zakin andJ. Rogers, "Silicon electronics on silk as a path to bioresorbable, implantable devices", Appl. Phys. Lett, 2009.

41. K. Hashiguchi, T. Morioka, F. Yoshida, Y. Miyagi, "Correlation between scalp-recorded electroencephalographic and electrocorticographic activities during ictal period", 2007.

42. E. Asano, C.Juhasz, A. Shah andO.Muzik, "Origin and propagation of epileptic spasms delineated on electrocorticography", 2005.

Appendix A: System Design Code and HDL Design Code

A.1. System Codes

1- DCT 4*4:

```
////////////////////////////////////////////////////////////////////////////////
clc
clear all
%clearvars -except f_in

close all
figure

row_aspect = 32 ;
col_aspect = 32 ;

%AXIS([0.1 0.3 32 48])
%time  = [3,3,3,3,1,1];
%power = [148,146,56,55,13,13];
time  = [.75 .75 .67 .67 1];
power = [.31,.34,.98,1,.18,1];
% power = [1,1,1,1,1,1];
% time  = [1,1,1,1,1,1];
f_in = func_DATA_IN(row_aspect,col_aspect);
%f_in = func_DATA_IN_2;
for k = 1 : 1 : 7;
f_in{k} = round(f_in{k});
f_in{k} = fixptc('s7.0','CLIP_S', f_in{k}) ;
end
DWT_func = 1;

DCTQ=[...
      16   16   17   21  ;...
      16   17   21   24  ;...
      17   21   24   36  ;...
      21   24   36   57  ;];
z=[5 2 3 6 9 13 10 7 4 8 11 14 15 12 16];
quality_factor = [1:1:10];
f_out{1} = zeros(row_aspect,col_aspect);
for  type = 1 : 2
        hold on
if(type == 1 )
quality_factor = [.5:.5:5];
elseif(type == 2 )
quality_factor = [1:1:10];
        end
        for index = 1:1:10

            for frame = 1:1:4

                stream = [] ;
                if (type == 1)
```

```
                    in = f_in{frame+1};
                else
                    in = f_in{frame+1}-f_in{frame};
                end
                k=0;
                for i=1:4:row_aspect
                    for j=1:4:col_aspect
                        in4_4=in(i:i+3,j:j+3);
 %                       int = dct(in4_4,4);
 %                       dct_out = dct(transpose(int),4);
dct_out = dct_fixed(in4_4',4);        %s9.1
                        if (type == 1)
DCTQ_mux = DCTQ ;
                            mask = ones(4,4);
                        else
%                                   DCTQ_mux = DCTQ ;
%                                   mask = ones(4,4);
DCTQ_mux = round(DCTQ ) ;
                            mask = [1   1   1   1
                                    1   1   1   0
                                    1   1   0   0
                                    1   0   0   0];
                        end

out_mask = dct_out .* mask ;
DCTQ_mux_inv = 1 ./ DCTQ_mux;
DCTQ_mux_inv = fixptc('0.13','CLIP_S',DCTQ_mux_inv) ;

tc2hex(DCTQ_mux_inv,0,13,'DCTQ_mux_inv.txt',0);
                        temp = (out_mask*quality_factor(index)) .*
DCTQ_mux_inv   ;
                        %temp = (round(temp.*2)./2);
                        temp = fixptc('s9.0','CLIP_S',temp) ; %%9.0
%9.1 is critical in performance % this change afteer bit-matching
out_enc(i:i+3,j:j+3) = temp;
                        k=k+1;
zig_zag_dc(k,1) = temp(1,1)*(2^0); %%
zig_zag_ac(k,1:15) = temp(z)*(2^0); %%
                    end
                end

                %% Huffman Compression

dpcm(1,1)=zig_zag_dc(1,1);

stream=cat(2,stream,huffman_dc(dpcm(1,1)),huffman_ac(zig_zag_ac(1,1:15
)));
                for m=2:k
dpcm(m,1)=zig_zag_dc(m,1)-zig_zag_dc(m-1,1);

stream=cat(2,stream,huffman_dc(dpcm(m,1)),huffman_ac(zig_zag_ac(m,1:15
)));
                end

Compressed_image_size(frame)=floor(length(stream)/8);
        %         Compression_Ratio(quality_factor) =
Compressed_image_size/(1024);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
                %% Decoding

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

                for i=1:4:row_aspect
                    for j=1:4:col_aspect
out_mask = out_enc(i:i+3,j:j+3) .* DCTQ_mux ./ quality_factor(index);
inv_int = idct(out_mask,4);
inv_out(i:i+3,j:j+3) = idct(transpose(inv_int),4);
                        end
                end

f_out{frame+1} = inv_out;

                out =  f_out{frame+1} ;

                for i = 1:1:row_aspect
                    for j = 1:1:col_aspect
                            if(f_in{frame+1}(i,j) ~= 0 &&
abs(f_in{frame+1}(i,j)-f_out{frame+1}(i,j)) ~= 0)
                                SNDR(i,j) =
10*log10((abs(f_in{frame+1}(i,j))^2)/(abs(f_in{frame+1}(i,j)-
f_out{frame+1}(i,j))^2));
                            else
                                SNDR(i,j) = 0;
                            end
                    end
                end
                M_SNDR = mean(SNDR) ;
Mean_SNDR(frame) = mean(M_SNDR) ;
Mean_SNDR(frame) = Mean_SNDR(frame);

                Q=255;
                PSNR(frame) =  10*log10(Q*Q/(sum(sum((f_out{frame+1}-
f_in{frame+1}).^2))/row_aspect/col_aspect));
            end


Mean_frames_SNDR(index) = mean(Mean_SNDR) ;

Compressed_frames_size=sum(Compressed_image_size);
Compression_Ratio(index) =
Compressed_frames_size/(row_aspect*col_aspect*frame);

fprintf('done!\n');
fprintf('-----------   Performance   ----------------\n');

fprintf('The bitrate is %.2f bpp \n',
length(stream)/row_aspect/col_aspect);

            Q = 255;
            MSE = sum(sum((f_out{2}-
f_in{2}).^2))/row_aspect/col_aspect;
fprintf('The psnr performance is %.2f dB\n', 10*log10(Q*Q/MSE));
```

```
Mean_PSNR(index) = mean(PSNR);
        end
        hold on
      plot(Compression_Ratio,smooth((Mean_PSNR/(time(1)*power(1)))),'o
-')
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


legend ('DCT 4*4','Location','SouthEast'); %error
xlabel ('Compressed data/Original data');
ylabel ('SNDR/(LATENCY*AREA)');
%ylabel ('SNDR');
grid on

////////////////////////////////////////////////////////////////////////////////
```

2- Differential DCT 4*4:

```
////////////////////////////////////////////////////////////////////////////////
clc
clear all
%clearvars -except f_in

close all
figure

row_aspect = 32 ;
col_aspect = 32 ;

%AXIS([0.1 0.3 32 48])
%time  = [3,3,3,3,1,1];
%power = [148,146,56,55,13,13];
time  = [.75 .75 .67 .67 1];
power = [.31,.34,.98,1,.18,1];
% power = [1,1,1,1,1,1];
% time  = [1,1,1,1,1,1];
f_in = func_DATA_IN(row_aspect,col_aspect);
%f_in = func_DATA_IN_2;
for k = 1 : 1 : 7;
f_in{k} = round(f_in{k});
f_in{k} = fixptc('s7.0','CLIP_S', f_in{k}) ;
end
DWT_func = 1;

DCTQ=[...
      16   16   17   21  ;...
      16   17   21   24  ;...
      17   21   24   36  ;...
      21   24   36   57  ;];
z=[5 2 3 6 9 13 10 7 4 8 11 14 15 12 16];
quality_factor = [1:1:10];
f_out{1} = zeros(row_aspect,col_aspect);
for  type = 1 : 2
        hold on
if(type == 1 )
quality_factor = [.5:.5:5];
elseif(type == 2 )
quality_factor = [1:1:10];
        end
        for index = 1:1:10

            for frame = 1:1:4

                stream = [] ;
                if (type == 1)
                    in = f_in{frame+1};
                else
                    in = f_in{frame+1}-f_in{frame};
                end
                k=0;
                for i=1:4:row_aspect
                    for j=1:4:col_aspect
                        in4_4=in(i:i+3,j:j+3);
 %                       int = dct(in4_4,4);
 %                       dct_out = dct(transpose(int),4);
```

```
dct_out = dct_fixed(in4_4',4);        %s9.1
                        if (type == 1)
DCTQ_mux = DCTQ ;
                            mask = ones(4,4);
                        else
%                                DCTQ_mux = DCTQ ;
%                                mask = ones(4,4);
DCTQ_mux = round(DCTQ ) ;
                            mask = [1   1   1   1
                                    1   1   1   0
                                    1   1   0   0
                                    1   0   0   0];
                        end

out_mask = dct_out .* mask ;
DCTQ_mux_inv = 1 ./ DCTQ_mux;
DCTQ_mux_inv = fixptc('0.13','CLIP_S',DCTQ_mux_inv) ;

tc2hex(DCTQ_mux_inv,0,13,'DCTQ_mux_inv.txt',0);
                        temp = (out_mask*quality_factor(index)) .*
DCTQ_mux_inv   ;
                        %temp = (round(temp.*2)./2);
                        temp = fixptc('s9.0','CLIP_S',temp) ; %%9.0
%9.1 is critical in performance % this change afteer bit-matching
out_enc(i:i+3,j:j+3) = temp;
                        k=k+1;
zig_zag_dc(k,1) = temp(1,1)*(2^0); %%
zig_zag_ac(k,1:15) = temp(z)*(2^0); %%
                    end
                end

                %% Huffman Compression

dpcm(1,1)=zig_zag_dc(1,1);

stream=cat(2,stream,huffman_dc(dpcm(1,1)),huffman_ac(zig_zag_ac(1,1:15
)));
                for m=2:k
dpcm(m,1)=zig_zag_dc(m,1)-zig_zag_dc(m-1,1);

stream=cat(2,stream,huffman_dc(dpcm(m,1)),huffman_ac(zig_zag_ac(m,1:15
)));
                end

Compressed_image_size(frame)=floor(length(stream)/8);
        %         Compression_Ratio(quality_factor) =
Compressed_image_size/(1024);




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
                %% Decoding

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%

                for i=1:4:row_aspect
                    for j=1:4:col_aspect
```

```
out_mask = out_enc(i:i+3,j:j+3) .* DCTQ_mux ./ quality_factor(index);
inv_int = idct(out_mask,4);
inv_out(i:i+3,j:j+3) = idct(transpose(inv_int),4);
                        end
                end

f_out{frame+1} = f_out{frame}+inv_out;

                out =  f_out{frame+1} ;

                for i = 1:1:row_aspect
                    for j = 1:1:col_aspect
                            if(f_in{frame+1}(i,j) ~= 0 &&
abs(f_in{frame+1}(i,j)-f_out{frame+1}(i,j)) ~= 0)
                                SNDR(i,j) =
10*log10((abs(f_in{frame+1}(i,j))^2)/(abs(f_in{frame+1}(i,j)-
f_out{frame+1}(i,j))^2));
                            else
                                SNDR(i,j) = 0;
                            end
                    end
                end
                M_SNDR = mean(SNDR) ;
Mean_SNDR(frame) = mean(M_SNDR) ;
Mean_SNDR(frame) = Mean_SNDR(frame);

                Q=255;
                PSNR(frame) =  10*log10(Q*Q/(sum(sum((f_out{frame+1}-
f_in{frame+1}).^2))/row_aspect/col_aspect));
            end


Mean_frames_SNDR(index) = mean(Mean_SNDR) ;

Compressed_frames_size=sum(Compressed_image_size);
Compression_Ratio(index) =
Compressed_frames_size/(row_aspect*col_aspect*frame);

fprintf('done!\n');
fprintf('-----------   Performance   ----------------\n');

fprintf('The bitrate is %.2f bpp \n',
length(stream)/row_aspect/col_aspect);

            Q = 255;
            MSE = sum(sum((f_out{2}-
f_in{2}).^2))/row_aspect/col_aspect;
fprintf('The psnr performance is %.2f dB\n', 10*log10(Q*Q/MSE));

Mean_PSNR(index) = mean(PSNR);
        end
        hold on

      plot(Compression_Ratio,smooth((Mean_PSNR/(time(2)*power(2)))),'x
-')
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

legend ('DF DCT 4*4','Location','SouthEast'); %error
```

```
xlabel ('Compressed data/Original data');
ylabel ('SNDR/(LATENCY*AREA)');
%ylabel ('SNDR');
grid on


////////////////////////////////////////////////////////////////////////////////
```

3- DCT 8*8:

```
////////////////////////////////////////////////////////////////////////////////
clc
clear all
%clearvars -except f_in

close all
figure

row_aspect = 32 ;
col_aspect = 32 ;

%AXIS([0.1 0.3 32 48])
%time  = [3,3,3,3,1,1];
%power = [148,146,56,55,13,13];
time  = [.75 .75 .67 .67 1];
power = [.31,.34,.98,1,.18,1];
% power = [1,1,1,1,1,1];
% time  = [1,1,1,1,1,1];
f_in = func_DATA_IN(row_aspect,col_aspect);
%f_in = func_DATA_IN_2;
for k = 1 : 1 : 7;
f_in{k} = round(f_in{k});
f_in{k} = fixptc('s7.0','CLIP_S', f_in{k}) ;
end
DWT_func = 1;
DCTQ=[...
      16   11   10   16   24   40   51   61   ;...
      12   12   14   19   26   58   60   55   ;...
      14   13   16   24   40   57   69   56   ;...
      14   17   22   29   51   87   80   62   ;...
      18   22   37   56   68   109  103  77   ;...
      24   36   55   64   81   194  113  92   ;...
      49   64   78   87   103  121  120  101  ;...
      72   92   95   98   112  100  103  99   ;];
z=[...
    9 2 3 10 17 25 18 11 4 5 12 19 26 ...
    33 41 34 27 20 13 6 7 14 21 28 35 ...
    42 49 57 50 43 36 29 22 15 8 16 23 ...
    30 37 44 51 58 59 52 45 38 31 24 32 ...
    39 46 53 60 61 54 47 40 48 55 62 63 56 64];
quality_factor = [1:1:10];
f_out{1} = zeros(row_aspect,col_aspect);
for  type = 1 : 1
        hold on
if(type == 1)
quality_factor = [.5:1:10.5];
elseif(type == 2)
quality_factor = [1:2:20];
        end
        for index = 1:1:10

            for frame = 1:1:4

                stream = [] ;
                if (type == 1)
                    in = f_in{frame+1};                          %s7.0
                else
```

```
                    in = f_in{frame+1}-f_in{frame};         %s7.0
                end
                k=0;
                for i=1:8:row_aspect
                    for j=1:8:col_aspect
                        in8_8=in(i:i+7,j:j+7);              %s7.0
 %                       int = dct(in8_8,8);
 %                       dct_out = dct(transpose(int),8);
dct_out = dct_fixed(in8_8',8);        %s9.1
                        if (type == 1)
DCTQ_mux = DCTQ ;
                            mask = ones(8,8);
                        else
DCTQ_mux = round(DCTQ ) ;
                            mask = [1   1   1   1   1   1   1   0
                                    1   1   1   1   1   1   0   0
                                    1   1   1   1   1   0   0   0
                                    1   1   1   1   0   0   0   0
                                    1   1   1   0   0   0   0   0
                                    1   1   0   0   0   0   0   0
                                    1   0   0   0   0   0   0   0
                                    0   0   0   0   0   0   0   0 ];
                        end
out_mask = dct_out .* mask ;
DCTQ_mux_inv = 1 ./ DCTQ_mux;
DCTQ_mux_inv = fixptc('0.13','CLIP_S',DCTQ_mux_inv) ;

%tc2hex(DCTQ_mux_inv,0,13,'DCTQ_mux_inv.txt',0);
                        temp = (out_mask*quality_factor(index)) .*
DCTQ_mux_inv   ;
                        %temp = (round(temp.*2)./2);
                        temp = fixptc('s9.0','CLIP_S',temp) ; %%9.0
%9.1 is critical in performance % this change afteer bit-matching
out_enc(i:i+7,j:j+7) = temp;
                        k=k+1;
zig_zag_dc(k,1) = temp(1,1)*(2^0);
zig_zag_ac(k,1:63) = temp(z)*(2^0);
                    end
                end

                %% Huffman Compression

dpcm(1,1)=zig_zag_dc(1,1);

stream=cat(2,stream,huffman_dc(dpcm(1,1)),huffman_ac(zig_zag_ac(1,1:63
)));
                for m=2:k
dpcm(m,1)=zig_zag_dc(m,1)-zig_zag_dc(m-1,1);

stream=cat(2,stream,huffman_dc(dpcm(m,1)),huffman_ac(zig_zag_ac(m,1:63
)));
                end

Compressed_image_size(frame)=floor(length(stream)/8);
        %         Compression_Ratio(quality_factor) =
Compressed_image_size/(1024);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
                %% Decoding

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

                for i=1:8:row_aspect
                    for j=1:8:col_aspect
out_mask = out_enc(i:i+7,j:j+7) .* DCTQ_mux ./ quality_factor(index);
inv_int = idct(out_mask,8);
inv_out(i:i+7,j:j+7) = idct(transpose(inv_int),8);
                    end
                end

                if (type == 1)
f_out{frame+1} = inv_out;
                else
f_out{frame+1} = f_out{frame}+inv_out;
                end

                out =  f_out{frame+1} ;

                for i = 1:1:row_aspect
                    for j = 1:1:col_aspect
                            if(f_in{frame+1}(i,j) ~= 0 &&
abs(f_in{frame+1}(i,j)-f_out{frame+1}(i,j)) ~= 0)
                                SNDR(i,j) =
10*log10((abs(f_in{frame+1}(i,j))^2)/(abs(f_in{frame+1}(i,j)-
f_out{frame+1}(i,j))^2));
                            else
                                SNDR(i,j) = 0;
                            end
                    end
                end
                M_SNDR = mean(SNDR) ;
Mean_SNDR(frame) = mean(M_SNDR) ;
Mean_SNDR(frame) = Mean_SNDR(frame);

                Q=255;
                PSNR(frame) =  10*log10(Q*Q/(sum(sum((f_out{frame+1}-
f_in{frame+1}).^2))/row_aspect/col_aspect));
            end


Mean_frames_SNDR(index) = mean(Mean_SNDR) ;

Compressed_frames_size=sum(Compressed_image_size);
Compression_Ratio(index) =
Compressed_frames_size/(row_aspect*col_aspect*frame);

fprintf('done!\n');
fprintf('-----------   Performance   ----------------\n');

fprintf('The bitrate is %.2f bpp \n',
length(stream)/row_aspect/col_aspect);

            Q = 255;
```

```
            MSE = sum(sum((f_out{2}-
f_in{2}).^2))/row_aspect/col_aspect;
fprintf('The psnr performance is %.2f dB\n', 10*log10(Q*Q/MSE));

Mean_PSNR(index) = mean(PSNR);
        end
        hold on
if(type == 1)

plot(Compression_Ratio,smooth((Mean_PSNR/(time(3)*power(3)))),'ro-')
elseif(type == 2)

plot(Compression_Ratio,smooth((Mean_PSNR/(time(4)*power(4)))),'rx-')
        end
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

legend ('DCT 8*8','Location','SouthEast'); %error
xlabel ('Compressed data/Original data');
ylabel ('SNDR/(LATENCY*AREA)');
%ylabel ('SNDR');
grid on

////////////////////////////////////////////////////////////////////////////////
```

4- Differential DCT 8*8:

```
////////////////////////////////////////////////////////////////////////////////

clc
clear all
%clearvars -except f_in

close all
figure

row_aspect = 32 ;
col_aspect = 32 ;

%AXIS([0.1 0.3 32 48])
%time  = [3,3,3,3,1,1];
%power = [148,146,56,55,13,13];
time  = [.75 .75 .67 .67 1];
power = [.31,.34,.98,1,.18,1];
% power = [1,1,1,1,1,1];
% time  = [1,1,1,1,1,1];
f_in = func_DATA_IN(row_aspect,col_aspect);
%f_in = func_DATA_IN_2;
for k = 1 : 1 : 7;
f_in{k} = round(f_in{k});
f_in{k} = fixptc('s7.0','CLIP_S', f_in{k}) ;
end
DWT_func = 1;
DCTQ=[...
      16   11   10   16   24   40   51   61   ;...
      12   12   14   19   26   58   60   55   ;...
      14   13   16   24   40   57   69   56   ;...
      14   17   22   29   51   87   80   62   ;...
      18   22   37   56   68   109  103  77   ;...
      24   36   55   64   81   194  113  92   ;...
      49   64   78   87   103  121  120  101  ;...
      72   92   95   98   112  100  103  99   ;];
z=[...
    9 2 3 10 17 25 18 11 4 5 12 19 26 ...
    33 41 34 27 20 13 6 7 14 21 28 35 ...
    42 49 57 50 43 36 29 22 15 8 16 23 ...
    30 37 44 51 58 59 52 45 38 31 24 32 ...
    39 46 53 60 61 54 47 40 48 55 62 63 56 64];
quality_factor = [1:1:10];
f_out{1} = zeros(row_aspect,col_aspect);
for  type = 2 : 2
        hold on
if(type == 1)
quality_factor = [.5:1:10.5];
elseif(type == 2)
quality_factor = [1:2:20];
        end
        for index = 1:1:10

            for frame = 1:1:4

                stream = [] ;
                if (type == 1)
                    in = f_in{frame+1};                          %s7.0
```

```
                else
                    in = f_in{frame+1}-f_in{frame};         %s7.0
                end
                k=0;
                for i=1:8:row_aspect
                    for j=1:8:col_aspect
                        in8_8=in(i:i+7,j:j+7);              %s7.0
 %                       int = dct(in8_8,8);
 %                       dct_out = dct(transpose(int),8);
dct_out = dct_fixed(in8_8',8);        %s9.1
                        if (type == 1)
DCTQ_mux = DCTQ ;
                            mask = ones(8,8);
                        else
DCTQ_mux = round(DCTQ ) ;
                            mask = [1   1   1   1   1   1   1   0
                                    1   1   1   1   1   1   0   0
                                    1   1   1   1   1   0   0   0
                                    1   1   1   1   0   0   0   0
                                    1   1   1   0   0   0   0   0
                                    1   1   0   0   0   0   0   0
                                    1   0   0   0   0   0   0   0
                                    0   0   0   0   0   0   0   0 ];
                        end
out_mask = dct_out .* mask ;
DCTQ_mux_inv = 1 ./ DCTQ_mux;
DCTQ_mux_inv = fixptc('0.13','CLIP_S',DCTQ_mux_inv) ;

%tc2hex(DCTQ_mux_inv,0,13,'DCTQ_mux_inv.txt',0);
                        temp = (out_mask*quality_factor(index)) .*
DCTQ_mux_inv   ;
                        %temp = (round(temp.*2)./2);
                        temp = fixptc('s9.0','CLIP_S',temp) ; %%9.0
%9.1 is critical in performance % this change afteer bit-matching
out_enc(i:i+7,j:j+7) = temp;
                        k=k+1;
zig_zag_dc(k,1) = temp(1,1)*(2^0);
zig_zag_ac(k,1:63) = temp(z)*(2^0);
                    end
                end

                %% Huffman Compression

dpcm(1,1)=zig_zag_dc(1,1);

stream=cat(2,stream,huffman_dc(dpcm(1,1)),huffman_ac(zig_zag_ac(1,1:63
)));
                for m=2:k
dpcm(m,1)=zig_zag_dc(m,1)-zig_zag_dc(m-1,1);

stream=cat(2,stream,huffman_dc(dpcm(m,1)),huffman_ac(zig_zag_ac(m,1:63
)));
                end

Compressed_image_size(frame)=floor(length(stream)/8);
        %         Compression_Ratio(quality_factor) =
Compressed_image_size/(1024);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
                %% Decoding

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

                for i=1:8:row_aspect
                    for j=1:8:col_aspect
out_mask = out_enc(i:i+7,j:j+7) .* DCTQ_mux ./ quality_factor(index);
inv_int = idct(out_mask,8);
inv_out(i:i+7,j:j+7) = idct(transpose(inv_int),8);
                    end
                end

                if (type == 1)
f_out{frame+1} = inv_out;
                else
f_out{frame+1} = f_out{frame}+inv_out;
                end

                out =  f_out{frame+1} ;

                for i = 1:1:row_aspect
                    for j = 1:1:col_aspect
                            if(f_in{frame+1}(i,j) ~= 0 &&
abs(f_in{frame+1}(i,j)-f_out{frame+1}(i,j)) ~= 0)
                                SNDR(i,j) =
10*log10((abs(f_in{frame+1}(i,j))^2)/(abs(f_in{frame+1}(i,j)-
f_out{frame+1}(i,j))^2));
                            else
                                SNDR(i,j) = 0;
                            end
                    end
                end
                M_SNDR = mean(SNDR) ;
Mean_SNDR(frame) = mean(M_SNDR) ;
Mean_SNDR(frame) = Mean_SNDR(frame);

                Q=255;
                PSNR(frame) =  10*log10(Q*Q/(sum(sum((f_out{frame+1}-
f_in{frame+1}).^2))/row_aspect/col_aspect));
            end


Mean_frames_SNDR(index) = mean(Mean_SNDR) ;

Compressed_frames_size=sum(Compressed_image_size);
Compression_Ratio(index) =
Compressed_frames_size/(row_aspect*col_aspect*frame);

fprintf('done!\n');
fprintf('-----------   Performance   ----------------\n');

fprintf('The bitrate is %.2f bpp \n',
length(stream)/row_aspect/col_aspect);

            Q = 255;
```

```
            MSE = sum(sum((f_out{2}-
f_in{2}).^2))/row_aspect/col_aspect;
fprintf('The psnr performance is %.2f dB\n', 10*log10(Q*Q/MSE));

Mean_PSNR(index) = mean(PSNR);
        end
        hold on
if(type == 1)

plot(Compression_Ratio,smooth((Mean_PSNR/(time(3)*power(3)))),'ro-')
elseif(type == 2)

plot(Compression_Ratio,smooth((Mean_PSNR/(time(4)*power(4)))),'rx-')
        end
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
legend ('DF DCT 8*8','Location','SouthEast'); %error
xlabel ('Compressed data/Original data');
ylabel ('SNDR/(LATENCY*AREA)');
%ylabel ('SNDR');
grid on

////////////////////////////////////////////////////////////////////////////////
```

5- Adaptive DWT:

```
////////////////////////////////////////////////////////////////////////////////
clc
clear all
%clearvars -except f_in

close all
figure

row_aspect = 32 ;
col_aspect = 32 ;

%AXIS([0.1 0.3 32 48])
%time  = [3,3,3,3,1,1];
%power = [148,146,56,55,13,13];
time  = [.75 .75 .67 .67 1];
power = [.31,.34,.98,1,.18,1];
% power = [1,1,1,1,1,1];
% time  = [1,1,1,1,1,1];
f_in = func_DATA_IN(row_aspect,col_aspect);
%f_in = func_DATA_IN_2;
for k = 1 : 1 : 7;
f_in{k} = round(f_in{k});
f_in{k} = fixptc('s7.0','CLIP_S', f_in{k}) ;
end
DWT_func = 1;

if (row_aspect == 32)

z1=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,33,34,35,36,37,38,39,40,41,
42,43,44,45,46,47,48,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,9
7,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,129,130,13
1,132,133,134,135,136,137,138,139,140,141,142,143,144,161,162,163,164,
165,166,167,168,169,170,171,172,173,174,175,176,193,194,195,196,197,19
8,199,200,201,202,203,204,205,206,207,208,225,226,227,228,229,230,231,
232,233,234,235,236,237,238,239,240,257,258,259,260,261,262,263,264,26
5,266,267,268,269,270,271,272,289,290,291,292,293,294,295,296,297,298,
299,300,301,302,303,304,321,322,323,324,325,326,327,328,329,330,331,33
2,333,334,335,336,353,354,355,356,357,358,359,360,361,362,363,364,365,
366,367,368,385,386,387,388,389,390,391,392,393,394,395,396,397,398,39
9,400,417,418,419,420,421,422,423,424,425,426,427,428,429,430,431,432,
449,450,451,452,453,454,455,456,457,458,459,460,461,462,463,464,481,48
2,483,484,485,486,487,488,489,490,491,492,493,494,495,496,17,18,19,20,
21,22,23,24,25,26,27,28,29,30,31,32,49,50,51,52,53,54,55,56,57,58,59,6
0,61,62,63,64,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,113,114,
115,116,117,118,119,120,121,122,123,124,125,126,127,128,145,146,147,14
8,149,150,151,152,153,154,155,156,157,158,159,160,177,178,179,180,181,
182,183,184,185,186,187,188,189,190,191,192,209,210,211,212,213,214,21
5,216,217,218,219,220,221,222,223,224,241,242,243,244,245,246,247,248,
249,250,251,252,253,254,255,256,273,274,275,276,277,278,279,280,281,28
2,283,284,285,286,287,288,305,306,307,308,309,310,311,312,313,314,315,
316,317,318,319,320,337,338,339,340,341,342,343,344,345,346,347,348,34
9,350,351,352,369,370,371,372,373,374,375,376,377,378,379,380,381,382,
383,384,401,402,403,404,405,406,407,408,409,410,411,412,413,414,415,41
6,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,448,465,
466,467,468,469,470,471,472,473,474,475,476,477,478,479,480,497,498,49
9,500,501,502,503,504,505,506,507,508,509,510,511,512,513,514,515,516,
517,518,519,520,521,522,523,524,525,526,527,528,545,546,547,548,549,55
0,551,552,553,554,555,556,557,558,559,560,577,578,579,580,581,582,583,
```

```
584,585,586,587,588,589,590,591,592,609,610,611,612,613,614,615,616,61
7,618,619,620,621,622,623,624,641,642,643,644,645,646,647,648,649,650,
651,652,653,654,655,656,673,674,675,676,677,678,679,680,681,682,683,68
4,685,686,687,688,705,706,707,708,709,710,711,712,713,714,715,716,717,
718,719,720,737,738,739,740,741,742,743,744,745,746,747,748,749,750,75
1,752,769,770,771,772,773,774,775,776,777,778,779,780,781,782,783,784,
801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,833,83
4,835,836,837,838,839,840,841,842,843,844,845,846,847,848,865,866,867,
868,869,870,871,872,873,874,875,876,877,878,879,880,897,898,899,900,90
1,902,903,904,905,906,907,908,909,910,911,912,929,930,931,932,933,934,
935,936,937,938,939,940,941,942,943,944,961,962,963,964,965,966,967,96
8,969,970,971,972,973,974,975,976,993,994,995,996,997,998,999,1000,100
1,1002,1003,1004,1005,1006,1007,1008,529,530,531,532,533,534,535,536,5
37,538,539,540,541,542,543,544,561,562,563,564,565,566,567,568,569,570
,571,572,573,574,575,576,593,594,595,596,597,598,599,600,601,602,603,6
04,605,606,607,608,625,626,627,628,629,630,631,632,633,634,635,636,637
,638,639,640,657,658,659,660,661,662,663,664,665,666,667,668,669,670,6
71,672,689,690,691,692,693,694,695,696,697,698,699,700,701,702,703,704
,721,722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,753,7
54,755,756,757,758,759,760,761,762,763,764,765,766,767,768,785,786,787
,788,789,790,791,792,793,794,795,796,797,798,799,800,817,818,819,820,8
21,822,823,824,825,826,827,828,829,830,831,832,849,850,851,852,853,854
,855,856,857,858,859,860,861,862,863,864,881,882,883,884,885,886,887,8
88,889,890,891,892,893,894,895,896,913,914,915,916,917,918,919,920,921
,922,923,924,925,926,927,928,945,946,947,948,949,950,951,952,953,954,9
55,956,957,958,959,960,977,978,979,980,981,982,983,984,985,986,987,988
,989,990,991,992,1009,1010,1011,1012,1013,1014,1015,1016,1017,1018,101
9,1020,1021,1022,1023,1024;];
elseif (row_aspect == 24)

z1=[1,2,3,4,5,6,7,8,9,10,11,12,25,26,27,28,29,30,31,32,33,34,35,36,49,
50,51,52,53,54,55,56,57,58,59,60,73,74,75,76,77,78,79,80,81,82,83,84,9
7,98,99,100,101,102,103,104,105,106,107,108,121,122,123,124,125,126,12
7,128,129,130,131,132,145,146,147,148,149,150,151,152,153,154,155,156,
169,170,171,172,173,174,175,176,177,178,179,180,193,194,195,196,197,19
8,199,200,201,202,203,204,217,218,219,220,221,222,223,224,225,226,227,
228,241,242,243,244,245,246,247,248,249,250,251,252,265,266,267,268,26
9,270,271,272,273,274,275,276,13,14,15,16,17,18,19,20,21,22,23,24,37,3
8,39,40,41,42,43,44,45,46,47,48,61,62,63,64,65,66,67,68,69,70,71,72,85
,86,87,88,89,90,91,92,93,94,95,96,109,110,111,112,113,114,115,116,117,
118,119,120,133,134,135,136,137,138,139,140,141,142,143,144,157,158,15
9,160,161,162,163,164,165,166,167,168,181,182,183,184,185,186,187,188,
189,190,191,192,205,206,207,208,209,210,211,212,213,214,215,216,229,23
0,231,232,233,234,235,236,237,238,239,240,253,254,255,256,257,258,259,
260,261,262,263,264,277,278,279,280,281,282,283,284,285,286,287,288,28
9,290,291,292,293,294,295,296,297,298,299,300,313,314,315,316,317,318,
319,320,321,322,323,324,337,338,339,340,341,342,343,344,345,346,347,34
8,361,362,363,364,365,366,367,368,369,370,371,372,385,386,387,388,389,
390,391,392,393,394,395,396,409,410,411,412,413,414,415,416,417,418,41
9,420,433,434,435,436,437,438,439,440,441,442,443,444,457,458,459,460,
461,462,463,464,465,466,467,468,481,482,483,484,485,486,487,488,489,49
0,491,492,505,506,507,508,509,510,511,512,513,514,515,516,529,530,531,
532,533,534,535,536,537,538,539,540,553,554,555,556,557,558,559,560,56
1,562,563,564,301,302,303,304,305,306,307,308,309,310,311,312,325,326,
327,328,329,330,331,332,333,334,335,336,349,350,351,352,353,354,355,35
6,357,358,359,360,373,374,375,376,377,378,379,380,381,382,383,384,397,
398,399,400,401,402,403,404,405,406,407,408,421,422,423,424,425,426,42
7,428,429,430,431,432,445,446,447,448,449,450,451,452,453,454,455,456,
469,470,471,472,473,474,475,476,477,478,479,480,493,494,495,496,497,49
8,499,500,501,502,503,504,517,518,519,520,521,522,523,524,525,526,527,
```

```
528,541,542,543,544,545,546,547,548,549,550,551,552,565,566,567,568,56
9,570,571,572,573,574,575,576];
elseif (row_aspect == 16)
    z1 =
[1,2,3,4,5,6,7,8,17,18,19,20,21,22,23,24,33,34,35,36,37,38,39,40,49,50
,51,52,53,54,55,56,65,66,67,68,69,70,71,72,81,82,83,84,85,86,87,88,97,
98,99,100,101,102,103,104,113,114,115,116,117,118,119,120,9,10,11,12,1
3,14,15,16,25,26,27,28,29,30,31,32,41,42,43,44,45,46,47,48,57,58,59,60
,61,62,63,64,73,74,75,76,77,78,79,80,89,90,91,92,93,94,95,96,105,106,1
07,108,109,110,111,112,121,122,123,124,125,126,127,128,129,130,131,132
,133,134,135,136,145,146,147,148,149,150,151,152,161,162,163,164,165,1
66,167,168,177,178,179,180,181,182,183,184,193,194,195,196,197,198,199
,200,209,210,211,212,213,214,215,216,225,226,227,228,229,230,231,232,2
41,242,243,244,245,246,247,248,137,138,139,140,141,142,143,144,153,154
,155,156,157,158,159,160,169,170,171,172,173,174,175,176,185,186,187,1
88,189,190,191,192,201,202,203,204,205,206,207,208,217,218,219,220,221
,222,223,224,233,234,235,236,237,238,239,240,249,250,251,252,253,254,2
55,256;]
elseif (row_aspect == 8)
    z1 =
[1,2,3,4,9,10,11,12,17,18,19,20,25,26,27,28,5,6,7,8,13,14,15,16,21,22,
23,24,29,30,31,32,33,34,35,36,41,42,43,44,49,50,51,52,57,58,59,60,37,3
8,39,40,45,46,47,48,53,54,55,56,61,62,63,64];
end

quality_factor = [1:1:10];
f_out{1} = zeros(row_aspect,col_aspect);
for  type = 1
if(type == 1)
            Q1 = [2 4 8 12 13 15 16 16 18 20];
            Q2 = [2 2 2 2 2  2  3  4  9  10];
            Q3 = [2 2 2 2 3  3  3  4  9  10];
            Q4 = [.1 .1 .1 .1 .1  .1  .1  .1  .1  .1];
%             Q1 = [ 1  2  3  4  5  8 10   2   2   3   3   4   4   5
5   8   8  10  10   2   2   3   3   4   4   5   5   8   8  10  10  2
3  4  5  8 10  2  3  4  5  8 10  2  3  4   4  6  8  10 4  6  8  10 4
6  8  10    2  3  4  5  2  3  4  5  2  3  4   4  6  8  10 4  6  8  10
4  6  8  10  ];
%             Q2 = [.1 .1 .1 .1 .1 .1 .1   1  .1   1  .1   1  .1   1
.1   1  .1   1  .1   1   1   1   1   1   1   1   1   1   1   1  2
2  2  2  2  2  1  1  1  1  1  1  2  2  2   4  4  4  4  1  1  1  1  4
4  4  4    2  2  2  2  1  1  1  1  2  2  2   4  4  4  4  1  1  1  1
4  4  4  4   ];
%             Q3 = [.1 .1 .1 .1 .1 .1 .1  .1   1  .1   1  .1   1  .1
1  .1   1  .1   1   1   1   1   1   1   1   1   1   1   1   1  1
1  1  1  1  1  2  2  2  2  2  2  2  2  2   1  1  1  1  4  4  4  4  4
4  4  4    1  1  1  1  2  2  2  2  2  2  2   1  1  1  1  4  4  4  4
4  4  4  4   ];
%             Q4 = [ 1 .1 .1 .1 .1 .1 .1  .1  .1  .1  .1  .1  .1  .1
.1  .1  .1  .1  .1  .1  .1  .1  .1  .1  .1  .1  .1  .1  .1  .1  .1 .1
.1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1  .1 .1 .1 .1 .1 .1 .1 .1 .1
.1 .1 .1    1  1  1  1  1  1  1  1  1  1  1   1  1  1  1  1  1  1  1
1  1  1  1   ];


elseif(type == 2)
%             Q1 = [2 2 4 4 4 8 8 16 16 16 ];
%             Q2 = [1 2 1 2 4 4 8  8  8 16 ];
%             Q3 = [1 2 1 2 4 4 8  8  8 16 ];
%             Q4 = [1 2 1 1 1 2 2  2  4  4 ];
           Q1 = [2 4 6 8 10 12 14 16 18 20];
```

```
            Q2 = [2 2 3 4 5  6  7  8  9  10];
            Q3 = [2 2 3 4 5  6  7  8  9  10];
            Q4 = [.1 .1 .1 .1 .1  .1  .1  .1  .1  .1];
        end
        for index = 1:1:10

            for frame = 1:1:4

                stream = [] ;
                if (type == 1)
                    in = f_in{frame+1};
                else
                    in = f_in{frame+1}-f_in{frame};
                end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
filter_type = 'bior5.5';   %wavelet basis
                level = 1;
                [Lo_D,Hi_D,Lo_R,Hi_R] = wfilters(filter_type);
                if (DWT_func == 0)
                    [img_wavedata, S] = func_DWT(in, level, Lo_D,
Hi_D);
                    LL = img_wavedata(1               :(row_aspect/2)
, 1               :(col_aspect/2));
                    HL = img_wavedata(1               :(row_aspect/2)
, (col_aspect/2)+1: col_aspect)   ;
                    LH = img_wavedata((row_aspect/2)+1: row_aspect
, 1               :(col_aspect/2));
                    HH = img_wavedata((row_aspect/2)+1: row_aspect
, (col_aspect/2)+1: col_aspect)
elseif (DWT_func == 1)
filter_type = 'bior3.5';   %wavelet basis
                    [Lo_D,Hi_D,Lo_R,Hi_R] = wfilters(filter_type);
%                     [LL,LH,HL,HH] = dwt2(in,Lo_D,Hi_D,'mode','per');
                    tc2hex(in',7,0,'in.txt',0);
                    for r = 1: 1 :row_aspect
                        [L1(r,:),H1(r,:)] =
dwt_fix(in(r,:),Lo_D,Hi_D,'mode','per');
                    end
                    for c = 1: 1 :col_aspect/2
                        [LL(:,c),LH(:,c)] =
dwt_fix(L1(:,c),Lo_D,Hi_D,'mode','per');
                        [HL(:,c),HH(:,c)] =
dwt_fix(H1(:,c),Lo_D,Hi_D,'mode','per');  %s8.1
                    end
                end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                for k = 1 : 1 : 4
                    if k == 1
                        XX = LL ;
                        Q = Q1(index) ;
elseif k == 2
                        XX = LH ;
                        Q = Q2(index) ;
elseif k == 3
                        XX = HL ;
                        Q = Q3(index) ;
```

```
                    else
                        XX = HH ;
                        Q = Q4(index) ;
                    end
                    median =
floor(mean(reshape(XX,1,(row_aspect*col_aspect/4)))*2)/2; %s8.1
XX_median = XX - median ; %s8.1
max_width = abs(max(max(XX_median))/Q);
min_width = abs(min(min(XX_median))/Q);
max_width = fixptc('s8.5','CLIP_S',max_width) ;
min_width = fixptc('s8.5','CLIP_S',min_width) ;
XX_comp   = zeros((row_aspect/2),(col_aspect/2));
                    for i = 1:1:(row_aspect/2)
                        for j = 1:1:(col_aspect/2)
                            if XX_median(i,j) > 0
XX_temp = (XX_median(i,j)/max_width);
XX_comp(i,j) = sign(XX_median(i,j))*abs(XX_temp);
XX_comp(i,j) = fixptc('s9.0','CLIP_S',XX_comp(i,j)) ;
elseifXX_median(i,j) < 0
XX_temp = (XX_median(i,j)/min_width);
XX_comp(i,j) = sign(XX_median(i,j))*abs(XX_temp);
XX_comp(i,j) = fixptc('s9.0','CLIP_S',XX_comp(i,j)) ;
                            end
                        end
                    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
                %% Decoding

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%

                    for i = 1:1:(row_aspect/2)
                        for j = 1:1:(col_aspect/2)
                            if XX_comp(i,j) >= 0
XX_re(i,j) = (XX_comp(i,j)*max_width)+median;
                            else
XX_re(i,j) = (XX_comp(i,j)*min_width)+median;
                            end
                        end
                    end
                    if k == 1
LL_re = XX_re;
LL_comp = XX_comp;
elseif k == 2
LH_re = XX_re;
LH_comp = XX_comp;
elseif k == 3
HL_re = XX_re;
HL_comp = XX_comp;
                    else
HH_re = XX_re;
HH_comp = XX_comp;
                    end
                end

                if (DWT_func == 0)
img_wavedata_dec = [LL_re,HL_re;LH_re,HH_re];
```

```
                    Xrec2 = func_InvDWT(img_wavedata_dec, S, Lo_R,
Hi_R, level);
elseif (DWT_func == 1)
                    Xrec2 =
idwt2(LL_re,LH_re,HL_re,HH_re,Lo_R,Hi_R,'mode','per');
                end
inv_out = Xrec2 ;
                %% Huffman Compression

                temp = [LL_comp,LH_comp;HL_comp,HH_comp];
                stream = [];
                zig_zag_ac1(1:row_aspect*col_aspect) = temp(z1);

stream=huffman_ac(zig_zag_ac1(1:row_aspect*col_aspect));
Compressed_image_size(frame)=floor(length(stream)/8);

                if (type == 1)
f_out{frame+1} = inv_out;
                else
f_out{frame+1} = f_out{frame}+inv_out;
                end

                Q=255;
                PSNR(frame) =  10*log10(Q*Q/(sum(sum((f_out{frame+1}-
f_in{frame+1}).^2))/row_aspect/col_aspect));
            end

Compressed_frames_size=sum(Compressed_image_size);
Compression_Ratio(index) =
Compressed_frames_size/(row_aspect*col_aspect*frame);

Mean_PSNR(index) = mean(PSNR);
        end
        hold on
if(type == 1)

plot(Compression_Ratio,smooth((Mean_PSNR/(time(5)*power(5)))),'kx-')
elseif(type == 2)

plot(Compression_Ratio,smooth((Mean_PSNR/(time(6)*power(6)))),'gx-')
        end
    end

legend ('Adaptive DWT','Location','SouthEast'); %error
xlabel ('Compressed data/Original data');
ylabel ('SNDR/(LATENCY*AREA)');
%ylabel ('SNDR');
grid on

////////////////////////////////////////////////////////////////////////////////
```

6- PANDCA:

```
////////////////////////////////////////////////////////////////////////////////
clc
clear all
%clearvars -except f_in

close all
figure

row_aspect = 32 ;
col_aspect = 32 ;

%AXIS([0.1 0.3 32 48])
%time  = [3,3,3,3,1,1];
%power = [148,146,56,55,13,13];
time  = [.75 .75 .67 .67 1];
power = [.31,.34,.98,1,.18,1];
% power = [1,1,1,1,1,1];
% time  = [1,1,1,1,1,1];
f_in = func_DATA_IN(row_aspect,col_aspect);
%f_in = func_DATA_IN_2;
for k = 1 : 1 : 7;
f_in{k} = round(f_in{k});
f_in{k} = fixptc('s7.0','CLIP_S', f_in{k}) ;
end
DWT_func = 1;
DCTQ=[...
      16   11   10   16   24   40   51   61   ;...
      12   12   14   19   26   58   60   55   ;...
      14   13   16   24   40   57   69   56   ;...
      14   17   22   29   51   87   80   62   ;...
      18   22   37   56   68   109  103  77   ;...
      24   36   55   64   81   194  113  92   ;...
      49   64   78   87   103  121  120  101  ;...
      72   92   95   98   112  100  103  99   ;];
z=[...
    9 2 3 10 17 25 18 11 4 5 12 19 26 ...
    33 41 34 27 20 13 6 7 14 21 28 35 ...
    42 49 57 50 43 36 29 22 15 8 16 23 ...
    30 37 44 51 58 59 52 45 38 31 24 32 ...
    39 46 53 60 61 54 47 40 48 55 62 63 56 64];
quality_factor = [1:1:10];
f_out{1} = zeros(row_aspect,col_aspect);
for  type = 1 : 1
        hold on
if(type == 1)
quality_factor = [.5:1:10.5];
elseif(type == 2)
quality_factor = [1:2:20];
        end
        for index = 1:1:10

            for frame = 1:1:4

                stream = [] ;
                if (type == 1)
                    in = f_in{frame+1};                    %s7.0
                else
                    in = f_in{frame+1}-f_in{frame};        %s7.0
```

```
                end
                k=0;
                for i=1:8:row_aspect
                    for j=1:8:col_aspect
                        in8_8=in(i:i+7,j:j+7);              %s7.0
 %                       int = dct(in8_8,8);
 %                       dct_out = dct(transpose(int),8);
dct_out = dct_fixed(in8_8',8);        %s9.1
                        if (type == 1)
DCTQ_mux = DCTQ ;
                            mask = ones(8,8);
                        else
DCTQ_mux = round(DCTQ ) ;
                            mask = [1   1   1   1   1   1   1   0
                                    1   1   1   1   1   1   0   0
                                    1   1   1   1   1   0   0   0
                                    1   1   1   1   0   0   0   0
                                    1   1   1   0   0   0   0   0
                                    1   1   0   0   0   0   0   0
                                    1   0   0   0   0   0   0   0
                                    0   0   0   0   0   0   0   0 ];
                        end
out_mask = dct_out .* mask ;
DCTQ_mux_inv = 1 ./ DCTQ_mux;
DCTQ_mux_inv = fixptc('0.13','CLIP_S',DCTQ_mux_inv) ;

%tc2hex(DCTQ_mux_inv,0,13,'DCTQ_mux_inv.txt',0);
                        temp = (out_mask*quality_factor(index)) .*
DCTQ_mux_inv   ;
                        %temp = (round(temp.*2)./2);
                        temp = fixptc('s9.0','CLIP_S',temp) ; %%9.0
%9.1 is critical in performance % this change afteer bit-matching
out_enc(i:i+7,j:j+7) = temp;
                        k=k+1;
zig_zag_dc(k,1) = temp(1,1)*(2^0);
zig_zag_ac(k,1:63) = temp(z)*(2^0);
                    end
                end

                %% Huffman Compression

dpcm(1,1)=zig_zag_dc(1,1);

stream=cat(2,stream,huffman_dc(dpcm(1,1)),huffman_ac(zig_zag_ac(1,1:63
)));
                for m=2:k
dpcm(m,1)=zig_zag_dc(m,1)-zig_zag_dc(m-1,1);

stream=cat(2,stream,huffman_dc(dpcm(m,1)),huffman_ac(zig_zag_ac(m,1:63
)));
                end

Compressed_image_size(frame)=floor(length(stream)/8);
        %         Compression_Ratio(quality_factor) =
Compressed_image_size/(1024);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
                %% Decoding

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

                for i=1:8:row_aspect
                    for j=1:8:col_aspect
out_mask = out_enc(i:i+7,j:j+7) .* DCTQ_mux ./ quality_factor(index);
inv_int = idct(out_mask,8);
inv_out(i:i+7,j:j+7) = idct(transpose(inv_int),8);
                    end
                end

                if (type == 1)
f_out{frame+1} = inv_out;
                else
f_out{frame+1} = f_out{frame}+inv_out;
                end

                out =  f_out{frame+1} ;

                for i = 1:1:row_aspect
                    for j = 1:1:col_aspect
                            if(f_in{frame+1}(i,j) ~= 0 &&
abs(f_in{frame+1}(i,j)-f_out{frame+1}(i,j)) ~= 0)
                                SNDR(i,j) =
10*log10((abs(f_in{frame+1}(i,j))^2)/(abs(f_in{frame+1}(i,j)-
f_out{frame+1}(i,j))^2));
                            else
                                SNDR(i,j) = 0;
                            end
                    end
                end
                M_SNDR = mean(SNDR) ;
Mean_SNDR(frame) = mean(M_SNDR) ;
Mean_SNDR(frame) = Mean_SNDR(frame);

                Q=255;
                PSNR(frame) =  10*log10(Q*Q/(sum(sum((f_out{frame+1}-
f_in{frame+1}).^2))/row_aspect/col_aspect));
            end


Mean_frames_SNDR(index) = mean(Mean_SNDR) ;

Compressed_frames_size=sum(Compressed_image_size);
Compression_Ratio(index) =
Compressed_frames_size/(row_aspect*col_aspect*frame);

fprintf('done!\n');
fprintf('-----------   Performance   ----------------\n');

fprintf('The bitrate is %.2f bpp \n',
length(stream)/row_aspect/col_aspect);

            Q = 255;
```

```
            MSE = sum(sum((f_out{2}-
f_in{2}).^2))/row_aspect/col_aspect;
fprintf('The psnr performance is %.2f dB\n', 10*log10(Q*Q/MSE));

Mean_PSNR(index) = mean(PSNR);
        end
        hold on
if(type == 1)

plot(Compression_Ratio,smooth((Mean_PSNR/(time(3)*power(3)))),'ro-')
elseif(type == 2)

plot(Compression_Ratio,smooth((Mean_PSNR/(time(4)*power(4)))),'rx-')
        end
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

legend ('DCT 8*8','Location','SouthEast'); %error
xlabel ('Compressed data/Original data');
ylabel ('SNDR/(LATENCY*AREA)');
%ylabel ('SNDR');
grid on

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

Sc        =
[4,4,4,4;7,7,7,7;9,9,9,8;10,10,10,10;11,11,11,10;12,13,12,11;13,13,12,
11;13,13,14,13;13,14,15,13;14,15,17,16;]; %results from previous part

Sc_frames = [ Sc(:,1),Sc(:,2),Sc(:,3),Sc(:,4),Sc(:,3),Sc(:,2)];

Sc_frames = repmat (Sc_frames,1,10);

H_p = repmat(97.427808,1,60);
% Sp = repmat(11.5,1,60);
% Q_F(1) = 5;
% Sp = (5:.13:14)
% Q_F(1) = 2;
% Sp = (12:-.13:3)
Sp = [10 10 10 (10:-.13:6) 6 6 6 6 6 (6:.13:8) 8 8 8 8 8]
Q_F(1) = 5;
for i = 1:1:60
    if(Sc_frames(Q_F(i)) > Sp(i))
        Q_F(i+1) = Q_F(i) - 1;
    elseif(Sc_frames(Q_F(i)) < Sp(i))
        Q_F(i+1) = Q_F(i) + 1;
    elseif(Sc_frames(Q_F(i)) == Sp(i))
        Q_F(i+1) = Q_F(i);
    end
Sc_saved(i) = Sc_frames(Q_F(i));
end
%plot ([1:1:61],Q_F)
ylim([4 13])
hold on
plot ([1:1:60],Sp(1:1:60),'r.-')
```

```
hold on
plot ([1:1:60],Sc_saved(1:1:60),'bx-')
legend ('Suggested Frame Size (Sp)','Actual Frame Size
(Sc)','Location','SouthEast'); %error
xlabel ('Frame Number');
ylabel ('Size (Byte)');
grid on
```

ملخص الرسالة

في الوقت الحاضر، يعتمد التقدم العلمي البحثي في الانشطه الدماغيه على ضغط الإشارة العصبيه الدماغيه العالية الدقه، من أجل التخزين الفعال وخفض معدل النقل للتمكن من الاتصال اللاسلكي بالعالم الخارجي.

وبدون ضغط البيانات، فإن معدلات البيانات هذه سوف تتعارض مع القيود العصبية الفيزيولوجية من حيث انخفاض الطاقة وانخفاض استهلاك مساحه الرقائق الالكترونيه.

بحيث يكون ضغط البيانات العصبية في موقع الزرع ضروري من أجل التوافق مع القيود للمعدلات الاسلكية. في هذه الأطروحة، يتم استخدام الارتباط المكاني العالي لزيادة نسبة ضغط البيانات.

ثم نقوم بالتحقيق ومقارنة خمسة خوارزميات مختلفة لضغط الطاقة المنخفضة المقترحة على أساس تحويل جيب التمام المنفصلة وتحويل المويجات المنفصلة لتوفير أفضل مقايضة بين تعقيد الأجهزة وكفاءه الضغط. وبالتال ينستنتج أن خوارزمية تحويل المويجات المنفصلة الثنائى الابعاد هو الحل الواعد للأجهزة المزروعه من اجل خفض الطاقة.

أجهزة العلاج الحالية تحتاج إلى استخراج المعلومات الكاملة لذاكره طويله لكل القنوات بدلا من استخراج ميزات إشارية خاصة فقط لتكون قادرة على الكشف وتشخيص الاضطرابات العصبيه الدماغيه. لذلك يجب أن نضمن أن البيانات العصبية يمكن أن تنتقل باستمرار دون أي توقف أو فقدان البيانات وأيضا يجب أن نضمن أن البيانات المضغوطة يمكن فك ضغطها في الجانب الآخر بجودة عالية دون تشويه كبير.

من هذا الهدف ،اقترحنا استخدام ميزانية الطاقة المحصودة المتاحة لتكيف خوارزمية ضغطنا لتكون قادرة على نقل البيانات العصبية المضغوطة باستمرار وتحقيق أقصى قدر من الكفاءه لنقل الإشارة وفق الميزانية الطاقة حصاد المتاحة دون أي فقدان للبيانات.

مهندس: محمد أشرف حسن إينال

تاريخ الميلاد: ١٩٩١\٠٩\٢١

الجنسية: مصرية

تاريخ التسجيل: ٢٠١٤\٣\١

تاريخ المنح: --\--\٢٠١٧

القسم: هندسة الإلكترونيات والاتصالات الكهربية

الدرجة: ماجستير العلوم

المشرفون:

أ.د. احمد العدوى

د. حسن مصطفي حسن مصطفي

الممتحنون:

أ.د.احمد العدوى (المشرف الرئيسي)

أ.د.---------- (الممتحن الداخلي)

أ.د. ---------- (الممتحن الخارجي)

عنوان الرسالة:

تصميم نموذج قابل للتكيف مع الطاقه لضغط البيانات العصبيه عالية الدقة

الكلمات الدالة:

التصميم الموفر للطاقه، التصميم الموفر للمساحه، ضغط البيانات العصبيه

ملخص الرسالة:

في الوقت الحاضر، يعتمد التقدم العلمي البحثي في الانشطه الدماغيه على ضغط الإشارة العصبيه الدماغيه العالية الدقه، من أجل التخزين الفعال وخفض معدل النقل للتمكن من الاتصال اللاسلكي بالعالم الخارجي. وبدون ضغط البيانات، فإن معدلات البيانات هذه سوف تتعارض مع القيود العصبية الفيزيولوجية من حيث انخفاض الطاقة وانخفاض استهلاك مساحه الرقائق الالكترونيه . بحيث يكون ضغط البيانات العصبية في موقع الزرع ضروري من أجل التوافق مع القيود للمعدلات الاسلكية. في هذه الأطروحة، يتم استخدام الارتباط المكاني العالي لزيادة نسبة ضغط البيانات. ثم نقوم بالتحقيق ومقارنة خمسة خوارزميات مختلفة لضغط الطاقة المنخفضة المقترحة على أساس تحويل جيب التمام المنفصلة وتحويل المويجات المنفصلة لتوفير أفضل مقايضة بين تعقيد الأجهزة

وكفاءه الضغط. وبالتال ينستنتج أن خوارزمية تحويل المويجات المنفصلة الثنائى الابعاد هو الحل الواعد للأجهزة المزروعه من اجل خفض الطاقة.

تصميم نموذج قابل للتكيف مع الطاقه لضغط البيانات العصبيه عالية الدقة

اعداد
محمد أشرف حسن إينال

رسالة مقدمة إلى
كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول على درجة
ماجستير العلوم
في
هندسة الإلكترونيات والاتصالات الكهربية

يعتمد من لجنة الممتحنين:

الاستاذ الدكتور: احمد العدوى المشرف الرئيسى

الاستاذ الدكتور: ---------- الممتحن الداخلي

الاستاذ الدكتور:---------- الممتحن الخارجي
(----------)

كليــة الهندســة - جامعــة القاهــرة
الجيـزة - جمهوريـة مصـرالعربيــة
٢٠١٧

تصميم نموذج قابل للتكيف مع الطاقه لضغط البيانات العصبيه عالية الدقة

اعداد
محمد أشرف حسن إينال

رسالة مقدمة إلى
كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول علىدرجة
ماجستير العلوم
في
هندسة الإلكترونيات والاتصالات الكهربية

تحت اشراف

أ.د. احمد العدوى
أستاذ
قسم هندسة الإلكترونيات
والاتصالات الكهربية
كليــة الهندســة - جامعــةالقاهــرة

د. حسن مصطفي حسن مصطفي
مدرس
قسم هندسة الإلكترونيات
والاتصالات الكهربية
كليــة الهندســة - جامعــة القاهــرة

كليــة الهندســة - جامعــة القاهــرة
الجيـزة - جمهوريـة مصـرالعربيــة
٢٠١٧

تصميم نموذج قابل للتكيف مع الطاقه لضغط البيانات العصبيه عالية الدقة

اعداد

محمد أشرف حسن إينال

رسالة مقدمة إلى كلية الهندسة – جامعة القاهرة

كجزء من متطلبات الحصول علي درجة

ماجستير العلوم

في

هندسة الإلكترونيات والاتصالات الكهربية

كليــة الهندســة - جامعــة القاهــرة

الجيـزة- جمهوريـة مصـرالعربيــة

٢٠١٧