

United Arab Emirates University

**Scholarworks@UAEU**

---

Dissertations

Electronic Theses and Dissertations

---

4-2023

## **APPROXIMATE COMPUTING BASED PROCESSING OF MEA SIGNALS ON FPGA**

Mohammad Emad Hassan

Follow this and additional works at: [https://scholarworks.uaeu.ac.ae/all\\_dissertations](https://scholarworks.uaeu.ac.ae/all_dissertations)



Part of the [Electrical and Computer Engineering Commons](#)

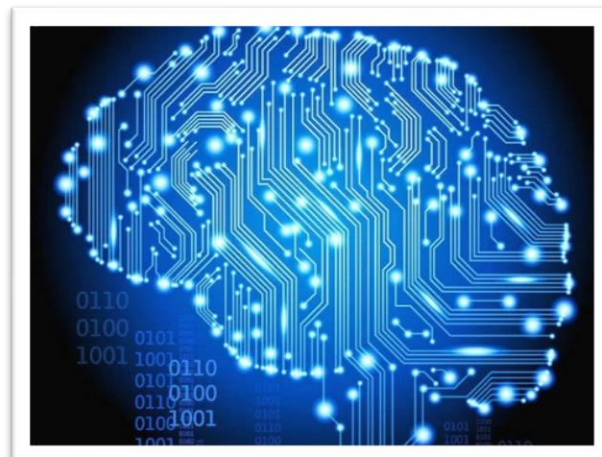
---

**DOCTORATE DISSERTATION NO. 2023: 9**

**College of Engineering**

**APPROXIMATE COMPUTING BASED PROCESSING OF  
MEA SIGNALS ON FPGA**

*Mohammad Emad Yousef Hassan*



*April 2023*

United Arab Emirates University

College of Engineering

APPROXIMATE COMPUTING BASED PROCESSING OF MEA  
SIGNALS ON FPGA

Mohammad Emad Yousef Hassan

This dissertation is submitted in partial fulfilment of the requirements for the degree of  
Doctor of Philosophy in Electrical Engineering

April 2023

Cover: Image related to brain neural signal processing system implemented in this research

(Photo: By Pablo Quezada, <https://www.medicaldevice-network.com/news/microelectrode-arrays-record-activity-deeper-brain/>)

## Declaration of Original Work

I, Mohammad Emad Yousef Hassan, the undersigned, a graduate student at the United Arab Emirates University (UAEU), and the author of this dissertation entitled “*Approximate Computing Based Processing of MEA Signals on FPGA*”, hereby, solemnly declare that this is the original research work done by me under the supervision of Prof. Falah Awwad, in the College of Engineering at UAEU. This work has not previously formed the basis for the award of any academic degree, diploma or a similar title at this or any other university. Any materials borrowed from other sources (whether published or unpublished) and relied upon or included in my dissertation have been properly cited and acknowledged in accordance with appropriate academic conventions. I further declare that there is no potential conflict of interest with respect to the research, data collection, authorship, presentation and/or publication of this dissertation.

Student's Signature:



Date: 25/4/2023

## **Advisory Committee**

1) Advisor: Falah Awwad

Title: Professor

Department of Electrical and Communication Engineering

College of Engineering

2) Co-advisor: Osman Hasan

Title: Professor

School of Electrical Engineering and Computer Science

Institute: National University of Sciences and Technology, Islamabad, Pakistan

## Approval of the Doctorate Dissertation

This Doctorate Dissertation is approved by the following Examining Committee Members:

- 1) Advisor (Committee Chair): Falah Awwad

Title: Professor

Department of Electrical and Communication Engineering

College of Engineering


Signature  Date 2/5/2023

- 2) Member: Amine El Moutaouakil

Title: Associate Professor

Department of Electrical and Communication Engineering

College of Engineering

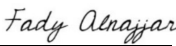
Signature  Date 2/5/2023

- 3) Member: Fady Al Najjar

Title: Associate Professor

Department of Computer Science and Software Engineering

College of Information Technology

Signature  Date 2/5/2023

- 4) Member (External Examiner): Manos M. Tentzeris

Title: Professor

Department: The School of Electrical and Computer Engineering

Institution: Georgia Institute of Technology, Atlanta, GA 30332-0250, USA

Signature  Date 4/5/2023

This Doctorate Dissertation is accepted by:

Dean of the College of Engineering: Professor Mohamed H. Al-Marzouqi

Signature Mohamed AlMarzouqi

Date May 26, 2023

Dean of the College of Graduate Studies: Professor Ali Al-Marzouqi

Signature Ali Hassan

Date 26/05/2023



## Abstract

The Microelectrode Array (MEA) is a collection of parallel electrodes that may measure the extracellular potential of nearby neurons. It is a crucial tool in neuroscience for researching the structure, operation, and behavior of neural networks. Using sophisticated signal processing techniques and architectural templates, the task of processing and evaluating the data streams obtained from MEAs is a computationally demanding one that needs time and parallel processing.

This thesis proposes enhancing the capability of MEA signal processing systems by using approximate computing-based algorithms. These algorithms can be implemented in systems that process parallel MEA channels using the Field Programmable Gate Arrays (FPGAs). In order to develop approximate signal processing algorithms, three different types of approximate adders are investigated in various configurations. The objective is to maximize performance improvements in terms of area, power consumption, and latency associated with real-time processing while accepting lower output accuracy within certain bounds.

On FPGAs, the methods are utilized to construct approximate processing systems, which are then contrasted with the precise system. Real biological signals are used to evaluate both precise and approximative systems, and the findings reveal notable improvements, especially in terms of speed and area. Processing speed enhancements reach up to 37.6%, and area enhancements reach 14.3% in some approximate system modes without sacrificing accuracy. Additional cases demonstrate how accuracy, area, and processing speed may be traded off.

Using approximate computing algorithms allows for the design of real-time MEA processing systems with higher speeds and more parallel channels. The application of approximate computing algorithms to process biological signals on FPGAs in this thesis is a novel idea that has not been explored before.

**Keywords:** Approximate Computing, Digital Systems, e-health, FPGA, Micro-electrode Arrays.

## Title and Abstract (in Arabic)

### المعالجة التقريبية في تحليل بيانات مصفوفات الأقطاب الكهربائية الدقيقة باستخدام FPGA

#### الملخص

مصفوفات الأقطاب الكهربائية الدقيقة هي الجهاز الأساسي في مجال علم الأعصاب لدراسة السلوك والتنظيم ومبادئ العمل للشبكات العصبية. تتكون المصفوفة الكاملة من الأقطاب الكهربائية المتوازية التي تعمل من خلال استشعار الجهد الكهربائي خارج الخلية للخلايا العصبية الموجودة بالقرب منها. معالجة وتحليل البيانات المتدفقة من تلك الأجهزة هي مهمة حسابية مكثفة تتطلب التوازي يتم تنفيذها باستخدام خوارزميات معالجة الإشارات المعقدة والقوالب الهيكلية.

في هذا البحث قمنا بتطوير خوارزميات حوسبة تقريبية لمعالجة الإشارات البيولوجية المتدفقة من مصفوفات الأقطاب الكهربائية عالية الكثافة على FPGA لتوفير مكاسب الأداء المثلّي في الحجم واستهلاك الطاقة وتقليل تعقيد الحساب والبطء المرتبط بالمعالجة، على حساب انخفاض دقة الناتج ضمن حدود معينة. ثلاثة أنواع من المجمعات التقريبية استخدمت في أوضاع وتراكيب مختلفة لتطوير خوارزميات معالجة البيانات وقد تم استخدام هذه الخوارزميات في أنظمة المعالجة وتحديد النبضات باستخدام FPGA ومقارنتها بالأنظمة الدقيقة.

تم اختبار الأنظمة المطورة على إشارات حيوية حقيقية وبينت النتائج زيادة في سرعة المعالجة تصل إلى 37.6% وتقليل في حجم النظام يصل إلى 14.3% بدون أي خسارة في دقة تحديد النبضات. في أوضاع أخرى للنظام المطور أظهرت النتائج تنازلاً في الدقة في مقابل الزيادة في السرعة والحجم. يفتح هذا البحث الفرص لتصميم نظم معالجة إشارات مصفوفات الأقطاب الكهربائية الدقيقة الآنية التي تعمل بسرعة معالجة أعلى وعدد أكبر من القنوات المتوازية إذ أن استخدام أنظمة الحسابات التقريبية في هذه الأنظمة هو فكرة جديدة في هذا المجال ولم تستخدم من قبل.

**مفاهيم البحث الرئيسية:** الحساب التقريبي، الأنظمة الرقمية، الصحة الاليكترونية، FPGA، مصفوفات الأقطاب الكهربائية الدقيقة.

## **Author's Contribution**

The contribution of Mohammad Hassan to the dissertation was as follows:

1. Participated in planning of the work, had main responsibility for the processing system design, experimental work, data collection and processing, and evaluation of results.

## **Author Profile**

Mohammad Hassan received his PhD in Electrical Engineering from the Department of Electrical and Communication Engineering, College of Engineering at UAE University, UAE. He also received his Master of Science in Electrical Engineering from the College of Engineering, UAE University, UAE.

## **Acknowledgements**

All praise be to Allah, who made it possible for me to do my assignment satisfactorily. I'm grateful to my father, who taught me and instilled a desire for knowledge in me. I appreciate my mother's prayers and support. Thanks to my wife, who supported me and took care of my concerns throughout my study sessions without displaying any overwork or neglect of her household responsibilities. I'm grateful to my kids for handling my absence.

I also want to express my gratitude to my supervisor, Prof. Falah Awwad, for his unwavering support, wisdom, and direction. I appreciate Prof. Osman Hassan's help and suggestions from the National University of Sciences and Technology in Pakistan's School of Electrical Engineering and Computer Science.

The committee members are thanked for their suggestions and thoughtful evaluation of my thesis.

I want to thank the entire department of electrical and communication engineering professors at United Arab Emirates University for their ongoing effort and support.

## **Dedication**

*To my beloved father, mother, wife, and family.*

## Table of Contents

Title.....	i
Declaration of Original Work.....	iii
Advisory Committee.....	iv
Approval of the Doctorate Dissertation.....	v
Abstract.....	vii
Title and Abstract (in Arabic).....	viii
Author's Contribution.....	ix
Author Profile .....	x
Acknowledgements.....	xi
Dedication.....	xii
Table of Contents.....	xiii
List of Tables .....	xv
List of Figures.....	xvi
List of Abbreviations .....	xviii
Chapter 1: Introduction.....	1
1.1 Overview .....	1
1.2 Statement of the Problem .....	2
1.3 Research Objectives .....	3
1.4 Relevant Literature .....	4
1.4.1 MEA Signal Processing Systems.....	5
Chapter 2: Methods.....	13
2.1 Research Design .....	13
2.2 System Overview .....	14
2.3 Proposed Approximate Processing System.....	16
2.3.1 Using FPGA for Implementation.....	16
2.3.2 Enhancing the Filtering Process .....	19
2.3.3 Applying Approximate Computing Algorithms .....	22
Chapter 3: Implementation .....	26
3.1 Implementation of Approximate Adders .....	26
3.1.1 Carry Prediction Full Adder (CPredA).....	27
3.1.2 Approximate Adder (AA2) .....	31

3.1.3 Generic Accuracy Configurable Adder (GeAr).....	36
3.2 Implementation of Processing Systems .....	39
3.2.1 Implementation of FIR Module .....	40
3.2.2 Implementation of Spike Detection Module.....	47
3.3 System Design Summary .....	50
Chapter 4: Results and Discussion .....	53
4.1 Evaluation of Approximate adders.....	53
4.1.1 Delay Time.....	55
4.1.2 Area.....	57
4.1.3 Power Estimation .....	58
4.1.4 Accuracy .....	60
4.2 Evaluation of Filtering Errors .....	61
4.3 Evaluation of the Approximate Processing System .....	68
4.3.1 Testing System Accuracy .....	69
4.3.2 Testing System Performance .....	71
Chapter 5: Conclusion .....	92
References.....	94
List of Publications .....	102



## List of Tables

Table 1: Benchmarking of this work with recent state of the art MEA processing systems.....	10
Table 2: Truth Table of FA and CpredA .....	28
Table 3: Truth Table of FA and AA2 .....	32
Table 4: Spike Detection Accuracy with Different FIR Orders .....	41
Table 5: Testing Results in half prediction configuration .....	54
Table 6: Testing Results in full prediction configuration.....	55
Table 7: Approximation Levels and Adders' Configurations Used in Filtering Tests .....	63
Table 8: Approximation Levels and Filters' NMED .....	65
Table 9: System Accuracy Results at Different Approximation Levels .....	70
Table 10: Approximate System Modes .....	71
Table 11: Evaluation results for all systems .....	88
Table 12: Statistical analysis of all modes.....	89

## List of Figures

Figure 1: Research Design.....	13
Figure 2: General block diagram for the neural processing system .....	15
Figure 3: Conceptual Structure of FPGA .....	17
Figure 4: Logic cell Conceptual Structure.....	17
Figure 5: ZedBoard Development Board .....	18
Figure 6: Vivado Design Suite interface .....	19
Figure 7: Symmetric FIR filter with order 8.....	22
Figure 8: Carry-Prediction Full Adder (CPredA).....	27
Figure 9: CPredA logical circuit code .....	29
Figure 10: 8-bit CPredA in full prediction configuration.....	29
Figure 11: Verilog code for 8-bit CPredA in full prediction configuration .....	30
Figure 12: 8-bit CPredA in half prediction configuration .....	30
Figure 13: Verilog code for 8-bit CPredA in half prediction configuration .....	31
Figure 14: Approximate Adder (AA2) .....	32
Figure 15: Verilog code for AA2 logical circuit .....	33
Figure 16: 8-bit AA2 in full prediction configuration.....	33
Figure 17: Verilog code for 8-bit AA2 in full prediction configuration .....	34
Figure 18: 8-bit AA2 in half prediction configuration .....	35
Figure 19: Verilog code for 8-bit AA2 adder in half prediction configuration.....	35
Figure 20: 12-bit GeAr adder in R2-P2 and full prediction configurations .....	36
Figure 21: Verilog code for 12-bit GeAr R2-P2 adder in full prediction mode.....	38
Figure 22: 12-bit GeAr in half prediction of the result .....	38
Figure 23: Verilog code for GeAr R2P2 in half prediction configuration .....	39
Figure 24: Detailed real-time processing system .....	40
Figure 25: Misdetetection of some spikes in FIR with order 4 .....	42
Figure 26: Implemented FIR filter.....	43
Figure 27: FIR filter Block 1 Verilog code .....	44
Figure 28: FIR filter Block 2 Verilog code .....	45
Figure 29: Verilog code when the approximate system uses CPredA adder .....	45
Figure 30: FIR filter Block 3 Verilog code .....	46
Figure 31: Verilog code for the full adders in Block 3.....	47
Figure 32: Verilog code when CPredA adders are selected .....	47
Figure 33: Spike detection module .....	48
Figure 34: Verilog code for the spike detection module .....	50
Figure 35: Common schematic diagram for All Systems .....	51
Figure 36: Adders delay time in half prediction configuration .....	56
Figure 37: Adders delay time in full prediction configuration.....	56
Figure 38: Adders area in half prediction configuration .....	57

Figure 39: Adders area in full prediction configuration .....	58
Figure 40: Power estimation of adders in half prediction configuration.....	59
Figure 41: Power estimation of adders in full prediction configuration .....	59
Figure 42: NMED of adders in half prediction configuration.....	60
Figure 43: NMED of adders in full prediction configuration.....	61
Figure 44: FIR Filter in Proposed Processing System.....	62
Figure 45: Sample of the filtering process output at approximation level 1 .....	64
Figure 46: NMED for the FIR filter using CPredA adder.....	66
Figure 47: NMED for the FIR filter using AA2 adder.....	66
Figure 48: NMED for the FIR filter using GeAr adder.....	67
Figure 49: Accuracy test results for processing systems in Mode 1 .....	72
Figure 50: Delay test results of processing systems in Mode 1 .....	73
Figure 51: Area test results of processing systems in Mode 1 .....	73
Figure 52: Accuracy test results for processing systems in Mode 2 .....	75
Figure 53: Delay test results of processing systems in Mode 2 .....	75
Figure 54: Area test results of processing systems in Mode 2 .....	76
Figure 55: Accuracy test results for processing systems in Mode 3 .....	77
Figure 56: Delay test results of processing systems in Mode 3 .....	77
Figure 57: Area test results of processing systems in Mode 3 .....	78
Figure 58: Sample test of signal filtering and spike detection in Mode 4.....	79
Figure 59: Accuracy test results of processing systems in Mode 4.....	80
Figure 60: Delay time test results of processing systems in Mode 4 .....	80
Figure 61: Area test results of processing systems in Mode 4 .....	81
Figure 62: Accuracy test results of processing systems in Mode 5.....	82
Figure 63: Delay time test results of processing systems in Mode 5 .....	82
Figure 64: Area test results of processing systems in Mode 5 .....	83
Figure 65: Accuracy test results of processing systems in Mode 6.....	84
Figure 66: Delay time test results of processing systems in Mode 6 .....	84
Figure 67: Area test results of processing systems in Mode 6 .....	85
Figure 68: Accuracy test results of processing systems in Mode 7.....	86
Figure 69: Delay time test results of processing systems in Mode 7 .....	86
Figure 70: Area test results of processing systems in Mode 7 .....	87

## **List of Abbreviations**

AA2	Approximate Adder 2
ACA	Almost Correct Adder
AVG	Average
C <sub>in</sub>	Input Carry
CMOS	Complementary Metal-Oxide Semiconductor
CMOS-MEA	Complementary Metal-Oxide Semiconductor Microelectrode Array
C <sub>out</sub>	Output Carry
CPredA	Carry Prediction Adder
ED	Error Distance
ECoG	Electrocorticography
EEG	Electroencephalography
FA	Full Adder
FFT	Fast Fourier Transform
FIR	Finite Impulse Response Filter
FP	Full Prediction Configuration
FPGA	Field Programmable Gate Array
GDA	Gracefully Degrading Adder
GeAr	Generic Accuracy Configurable Adder
HD-MEA	High-Density Microelectrode Array
HP	Half Prediction Configuration
IC	Integrated Circuit
LFP	Local Field Potentials
LSB	Least Significant Bit
MCU	Microcontroller Unit
MEA	Microelectrode Array

MED	Mean Error Distance
MSB	Most Significant Bit
NMED	Normalized Mean Error Distance
PC	Personal Computer
RTL	Register Transfer Level
S	Sum
SCSA	Speculative Carry Select Addition
SoC	System on Chip
STD	Standard Deviation



# Chapter 1: Introduction

## 1.1 Overview

Neuroscience studies the nervous system development, structure, and functions (Shi & Fang, 2018). It is the study of the neurological system from a scientific standpoint. Neuroscience studies not just how the nervous system works normally but also how the nervous system responds to neurological, psychiatric, or neurodevelopmental disorders (Özcan et al., 2021). Person's quality of life is affected by nervous system disorders, which are also the most challenging to cure of all the organ systems in the body. The majority of these issues are brought on by a decline in localized sensory or motor function (Slepova & Berkhova, 2019). By connecting electrodes to the residual limb's peripheral nerves, neuroprosthetics holds a lot of promise for helping people restore their sensation of touch (Ades et al., 2022). Restoring lost neural function can be done by capturing signals from active neurons and then selectively electrically stimulating a population of neurons' functions (Bavishi et al., 2019).

Neurons or neural cells are the essential building blocks of the neural system. Most neural proteins are made in the cell body (soma), which houses the nucleus. The axon passes messages from the cell body to other neurons through nerve impulses, and it is covered by a semipermeable plasma membrane (Varier et al., 2022).

A resting neuron has a voltage across its membrane known as the resting potential, which has a value of about -70 mV. High levels of positively charged sodium ions outside the cell, large levels of negatively charged chloride ions, and a lesser level of positively charged potassium inside the cell all contribute to this polarized condition. The slightly negative internal charge compared to the extracellular fluid causes this potential.

An action potential or nerve impulse is formed when neurons are triggered by the neighboring cell or externally by chemical, electrical, or optical stimulators. The stimulation of the neuron causes channel-shaped protein molecules in the cell membrane to partially open. Positive sodium ions enter through the partially opened channels and the action potential starts to form when that membrane region becomes less negative. The action potential, which lasts for a few milliseconds, is generated when the channels

fully open at the threshold potential of roughly -55 mV and the cell gets an injection of positive sodium ions. Depolarization also opens sodium channels in nearby membrane regions when the threshold value is exceeded, causing the impulse to travel through the axon (Betts et al., 2013).

Microelectrode Arrays (MEA) are arrays of up to thousands of metal contacts that are used to detect potentials in the extracellular environment and transform them into electrical potentials (Doliwa et al., 2022). Therefore, MEAs are fundamental equipment in neuroscience for studying neural network behavior, organization, and working principle (Huang et al., 2023).

In a closed loop MEA system, stimulating electrodes can be integrated into the same MEA with the recording electrodes, or they can be connected to an external stimulator which can be optical or electrical. This setup enables the study of neural system response by analyzing the spike activity caused by stimulating the neurons and drawing the map of bioelectrical signals with great details and resolution (Tanskanen et al., 2020). Open loop systems in these studies can be used to analyze the activity of neurons in addition to history-dependent neural network states without external effects (Kim et al., 2018). The success of neuroscience studies relies on the fabrication of MEA electrodes in contact with the neural tissue and the associated electronic processing system to obtain consistent and real-time recording signals from small groups of neurons with the highest resolution (Wang et al., 2021).

Huge technological advancements have happened in the field of neuroscience in the last decade. MEAs consisting of thousands of sensing electrodes are available that can instantly record and monitor the activity of many neural cells with high resolution (Bhaskara et al., 2022).

## **1.2 Statement of the Problem**

The analysis of neural activity is a difficult and computationally demanding operation. One challenge is reducing the latency associated with processing the raw signals received from the MEA devices to improve the stimulation response. To produce feedback stimuli with the lowest delay, the system must be capable of gathering data and processing it within the shortest amount of time. This challenge is difficult to overcome



since the processing system must filter the received raw signals and detect neural spiking events which last for a few milliseconds. Furthermore, reconfigurability is required quite frequently to run multiple setups and trials. The overall processing system should also have a high degree of parallelism to process the data received from various MEA channels in parallel. The circuit area of the processing system is also an important parameter since the system consists of parallel processing sets and programmable devices such as microcontrollers and Field Programmable Gate Array (FPGA) have limited resources that should be utilized efficiently. Hence, more processing sets can be built on the same chip and handle the data channels in parallel if the set circuit area is smaller. Less power consumption by each processing unit will also enhance the portability and decrease the overall power consumption of the system.

Recent approaches, e.g., (Lee et al., 2020), tried to overcome the previous challenges with adaptive electrode selection technique that scans the electrode arrays and record from selected electrodes where the neural spikes are detected to monitor larger number of electrodes. Other approaches, e.g., (Doliwa et al., 2022), proposed the filtering process of received signals outside the FPGA with analog electronics and utilized the chip resources for data acquisition, transmission, and other processing tasks.

### **1.3 Research Objectives**

This research aims to enhance the capability and overcome the mentioned challenges by applying approximate computing-based algorithms in a system that processes parallel MEA channels on FPGA. The proposed processing system uses approximate computing to provide optimal performance gains by reducing the computation complexity and latency associated with real-time processing. Approximate computing (Wang et al., 2022) is a computation technique that compromises the accuracy of the result while improving the area, power, and delay to meet the requirement of high-performance computing in error tolerant applications. Applications for signal processing, image processing, recognition, and data mining have all made extensive use of it.

Primary contributions of this research include:

1. Development of neural signal processing systems on FPGA using approximate computing and compare them with the precise system to assess the advantages and disadvantages. In this context, three types of approximate adders were utilized, i.e., GeAr (Raghuram & Shashank, 2022), AA2 (Gorantla & Deepa, 2020), and CPredA (Sato et al., 2019), in multiple configurations.
2. Testing the proposed approximate system using real biological signals. The results show significant enhancements, mainly in speed and area. Processing speed enhancement reaches up to 37.6%, and area reduction of 14.3% in some approximate systems compared to the accurate one without a loss in accuracy. The trade-off between processing speed, precision, and area is demonstrated by other approximation systems.

#### **1.4 Relevant Literature**

Neural interfaces were first used as basic scientific research instruments to investigate how the brain works (Kozai, 2018). Implantable microwires were used in 1958 (Strumwasser, 1958) to prove that recording neuron discharges is possible for a week or more in unrestrained animals. In that work, Felix Strumwasser implanted four to six stainless steel microwires with 80  $\mu\text{m}$  diameter into the skull of a squirrel. He observed the patterns at periods once the squirrel was dozing off, waking up, and awake.

The earliest article outlining a multielectrode array used in monitoring grown cells was authored by Thomas et al. in 1972 (Pine, 2006). The test array was constructed with two rows of 15 electrodes separated by 100  $\mu\text{m}$  (Thomas et al., 1972).

Five years later, in 1977, Gross introduced the idea of a multielectrode array similar to that of Thomas, unaware of the prior work. (Gross et al., 1977). A thermosetting polymer insulated the 36 gold electrodes in the array, which had a diameter of around 10  $\mu\text{m}$ . They were positioned 100 or 200  $\mu\text{m}$  apart and de-insulated using UV laser pulses.

The number of electrodes has recently increased as a result of MEA technological advancements. So, the prospect of constructing neural networks with a particular and well-defined topology is implied by the concept of having one neuron plated across the

surface of each electrode (Massobrio et al., 2015). A neuroelectronic connection connects neurons and micro-transducers in proximity. Long-term, noninvasive recordings of the electrical activity of extracellular neurons are made.

Spatial resolution is one of the main benefits of MEAs, allowing the accurate collection of neural signals. Modern MEAs consist of thousands of electrodes that send acquired biological signals simultaneously. Recent MEA devices use Complementary Metal Oxide Semiconductor (CMOS) technology to allow for high-speed, high-resolution imaging of electrical activity. They may have a culture- or slice-chamber to hold the sample while also allowing the use of a microscope. They also contain in a tiny chip thousands of sensing electrodes in addition to integrated stimulation circuits (Dragas et al., 2017).

#### *1.4.1 MEA Signal Processing Systems*

Different multichannel recording and stimulation systems have been introduced, including commercially available systems. Despite being well-designed, those systems are not specifically intended for each experiment's requirements, such as closed-loop setups, where processing time is critical to trigger a specific stimulus. Furthermore, due to the costs, researchers created subsystems for their unique experiments and provided cost-optimized open-source systems (Tanskanen et al., 2020).

Several works in the literature implemented systems to process the signals acquired by MEA devices and transmitted through their channels. Those systems mainly use Personal Computers (PC) with integrated and analog electronics or programmable devices such as microcontrollers and FPGAs.

##### *1.4.1.1 Processing Systems Using PCs*

A PC-based processing system is introduced by Venkatraman et al. (2009). An analog circuit was built to interface a PC with 16-electrodes MEA and filter the neural signals. The neural signals are acquired from MEA while its electrodes are implanted in an awake rat's barrel cortex and then sent to a custom software on the PC to extract the features and detect the spikes. The software was also used to trigger micro-stimulation based on the movement of the restrained rat. It processed the signals and generated stimulation within 15 ms.

Another work used a Mathworks Simulink-based application on PC to demonstrate real-time analysis of complicated high-bandwidth which is more than 10 MBit/s (Zrenner et al., 2010). The system recorded neural data while simultaneously generating specific complex electrical stimulation feedback with deterministically scheduled responses at a sub-millisecond resolution. The system was used to control an MEA with 60 electrodes and tested with predefined waveforms and stimulating electrodes. It could send stimulus signals in less than 1 ms, but changing the stimulating electrodes and waveforms took around 10 ms.

Wallach et al. (2011) used the previous system to introduce the Neuronal Response Clamp. A closed-loop technique enables control over the instantaneous response probability of the neuron.

The work by Newman et al. (2013) presented the NeuroRighter desktop application, which simplifies the design of sophisticated real-time experiments for multichannel interfacing experiments. The minimum latency achieved with this system when targeting a 64-channel MEA was around seven milliseconds.

A 320-channel active probe for high-spatial-resolution neuromonitoring and responsive neurostimulation was reported by Shulyzki et al. (2015). The work used the presented bidirectional integrated neural interface IC and a seizure-predicting application. An Integrated Circuit (IC) cell array was attached to the reverse side of a pitch-matched Microelectrode Array in the probe. The IC supported 256 neural recording sites and 64 neural stimulation sites. A PC application is used to control and predict seizures in a rat epilepsy model with online closed loop neurostimulation.

Liu et al. (2020) proposed a system based on memristor. The system uses the inherent memristor conductance modulation to extract the energy and variation information of the input neural signals. After the input signals are processed on the memristor array, the conductance of all the memristor devices can be read out by a PC workstation for further analysis, such as spike detection, feature extraction, and classification. The results were verified by demonstrating seizure prediction with high accuracy and improved power dissipation.

The work of Zhang et al. (2021) presented a fully integrated 64-channel neural recording system for local field and action potential. An analog system implemented on a 4x4 mm<sup>2</sup> chip die using the SMIC 0.18  $\mu$ m CMOS standard process. The chip was composed of 88 pins for input and output and several other parts including signal conditioning circuitry, a Successive-Approximation Register (SAR) ADC, a bandgap reference and bias circuitry, a digital logic unit for clock and control, and a unidirectional Serial Peripheral Interface (SPI). The work proposed the two-stage amplifier with high gain and the clock logic that can be used to align the switching clock as two novel ideas. The system was connected to PC software and tested on raw neural data downloaded from the internet. It recorded successfully and simultaneously multiple LFP and AP signals.

#### *1.4.1.2 Processing Systems Using Programmable Devices*

Programmable devices-based systems have been used to implement real-time processing systems of neural signals mainly because of their well-known gains in computational performance compared to PC-based systems.

Biffi et al. (2010) targeted an FPGA device to develop and validate a spike detection and classification algorithm. The algorithm was developed in the MATLAB environment and made up of an amplitude-threshold spikes detector based on noise level estimation and a hierarchical classifier to classify the spikes. The hardware design of the work was presented using FPGA and 60 electrodes MEA. Filtering, amplifying, and digitizing the acquired signal are assumed to be outside the FPGA, which was dedicated only to spike detection and classification.

The work of Muller et al. (2013) is an implementation of a closed-loop spike detection and stimuli generation system using FPGA. It processed data from 126 channels in parallel while generating stimuli on 42 different output channels in less than 1 ms.

An ASIC based processing system was presented by Zoladz et al. (2013). In that system, the 256 channels' amplification, filtering, and electrical stimulation are the ASICs' exclusive responsibility, which receives the signals at a sample rate of 14 kS/s and a resolution of 12 bits. The resultant byte stream is sent through a Universal Serial

Bus (USB) to a PC with an application for measurement, data presentation, and storage. The proposed system successfully interfaced the selected MEA and sent its signals to the PC for display and measurement.

A prototype bi-directional neural interface system with closed-loop and embedded DSP capabilities is presented by Cong et al. (2014). The system included 32-electrode stimulation capability, eight multiplexed low-noise, low-power bio-potential sensing channels with an on-chip digital FFT, and a Cortex M3-based microcontroller for implementing closed-loop algorithms.

A wireless portable 16-channel microcontroller system was provided in another work enabling two-way communication with the central nervous system (Angotzi et al., 2014). Eight of the sixteen available electrodes were used to stimulate the brain and connected directly to the stimulating unit on a rat backpack. The remaining eight electrodes were used to record the neural activity. The system architecture consisted of a remote unit, a home unit, and an optional control software running on a PC for offline processing. Detection-to-stimulation latency was 3 ms in the microcontroller closed loop for this system and 2.6 ms in the PC closed loop.

Liu et al. (2017) suggested a completely programmable, bidirectional neural device interfaced with 16-channel MEA. A general-purpose microcontroller was connected to a custom SoC that conducted noise-sensitive neural signal recording, neural stimulation, computation-intensive neural feature extraction, and on-chip closed-loop operation. The prototype system in that work interfaced the general-purpose microcontroller with a desktop PC through the universal wireless protocol (Bluetooth) for further processing and control.

Another FPGA implementation, presented by Park et al. (2017), was shown to do spike detection and sorting of 128 input channels simultaneously. However, no latency time was reported in this work. The paper focused on the hardware implementation of the 128-channel FPGA-based bidirectional neural interface system, including two 64-channel analog front-end boards. Filtering of signals was done on the analog front-end boards, while spike detection and sorting were done on the FPGA board.

An FPGA-based system was introduced to interface an MEA with 4K input channels and a sampling frequency of 18 kHz (Seu et al., 2018). The latency time of the system was less than 2 ms. The system proposed in that work was implemented entirely on an FPGA board, including spike detection and filtering of signals by digital filters.

The work by Lee et al. (2020) presented a multichannel neural recording device that records brain signals from many MEA electrodes using fewer recording channels. The system used an adaptive electrode selection technique to automatically scan the electrode arrays and record from chosen electrodes where brain spikes are identified. The grouped signals are connected to a microcontroller unit to determine the relative occurrence rate of neural spikes between scan groups and decide adaptive electrode selection. Results from experiments using pre-recorded brain data show that the proposed system can separate, amplify, and count neural spikes in real-time.

In the work of Chowdhury et al. (2020), the authors used an FPGA to prototype a low-power multichannel neuron activity extraction unit appropriate for a wireless neural interface. A neural signal extraction algorithm was proposed to achieve the low power requirement by reducing the data transmission rate. The algorithm is based on transmitting a particular channel that is recording a high-frequency signal above a specific voltage. The work results showed a reduction in the data transmission rate by up to 6000 times, which in turn consumes only 3 mW of the FPGA dynamic power.

The work of Doliwa et al. (2022) presented an analog front-end for brain computer interfaces. The system consisted of analog interface circuit to amplify the signals acquired from MEA electrodes with low noise amplifiers. The Common Average Referencing (CAR) algorithm is used in the interface circuit in order to generate an average reference signal of all electrodes that will be subtracted from the signal of each recording channel. The algorithm was used to replace the use of differential amplifier between the recording and reference electrodes and produce low noise neural signals. Signals are then digitized and forwarded to FPGA through the serial peripheral interface which transmits them to a host computer for spike detection and processing. A neural signal simulator with four channels was used to verify the processing system which achieved a signal to noise ratio of 38 dB with spike amplitudes of 100  $\mu$ V.

Finally, Saggese & Strollo (2022) proposed an energy-efficient spike detector ASIC that utilizes a hybrid version of Absolute Differential Operator (ADO) and Amplitude Slope Operator (ASO). This design employs nonlinear energy operators to differentiate neural spikes from background noise, striking a balance between hardware requirements and accuracy. By amplifying the differentiation between spikes and noise, signal to noise ratio is improved. Initially developed in MATLAB, the algorithms were later synthesized using VHDL modules to target TSMC 28 nm Complementary Metal-Oxide Semiconductor (CMOS) technology. The proposed system was implemented in a compact digital CMOS integrated circuit, featuring a multichannel architecture with the ability to process 1024 channels. The results of the work demonstrated high detection performance, moderate power consumption, and area efficiency, making it a suitable candidate for multichannel implantable neural data acquisition systems.

To the best of our knowledge, approximate computing has never been used in this context before. We believe that applying approximate computing in processing neural signals can bring significant speed enhancement and reduction in system circuit area without comprising much accuracy. With this motivation, in this research, we propose approximate computing-based algorithms for MEA signal processing, along with their FPGA implementations, and evaluate their usefulness in the medical fields of neuroscience, where MEAs are frequently used to analyze the behavior of neural networks.

Table 1 presents a benchmarking of this work with state of the art MEA processing systems implemented in recent years along with the main contributions.

Table 1: Benchmarking of this work with recent state of the art MEA processing systems

Reference	Hardware	Processing Tasks	Sampling Freq.	Latency	Input Channels	Contribution
(Shulyzki et al., 2015)	Analog IC + PC	Filtering + Seizure prediction	10 kHz	N/A	320	Predict the seizures with online closed loop stimulations.



Table 1: Benchmarking of this work with recent state of the art MEA processing systems (continued)

Reference	Hardware	Processing Tasks	Sampling Freq.	Latency	Input Channels	Contribution
(Liu et al., 2017)	SoC + Microcontroller	Digital Filtering + Spike Detection + Feature extraction	7 kHz	N/A	16	Implementation of a custom SoC that conducts recording, stimulation, feature extraction, and closed-loop operations.
(Park et al., 2017)	FPGA	Digital Filtering + Spike Detection + Spike Sorting	32.5 kHz	N/A	128	Implementation of online spike sorting and classification algorithm using FPGA device.
(Seu et al., 2018)	FPGA	Digital Filtering + Spike Detection	18 kHz	< 2 ms	4096	Implementation of real time HD-MEA interface and data acquisition using FPGA.
(Liu et al., 2020)	Memristors + PC	Spike Detection + Seizure prediction	10 kHz	N/A	16	Using memristor devices to read MEA signals and predict seizure with improved power efficiency.
(Lee et al., 2020)	Microcontroller	Adaptive electrode selection + Spike Detection	12.8 kHz	N/A	32	Implementation of adaptive electrode selection technique to record signals from MEA electrodes with fewer channels.
(Chowdhury et al., 2020)	FPGA	Digital filtering + Channel selection + spike detection	10 kHz	N/A	64	Implementation of neural signal extraction algorithm to achieve the low power requirement by reducing the data transmission rate.

Table 1: Benchmarking of this work with recent state of the art MEA processing systems (continued)

Reference	Hardware	Processing Tasks	Sampling Freq.	Latency	Input Channels	Contribution
(Zhang et al., 2021)	Analog IC + PC	Filtering + Spike Detection	19.2 kHz	N/A	64	Implementation of full neural interfacing system that uses two-stage amplifier and clock logic to synchronize the channel switching with the ADC sampling clock.
(Doliwa et al., 2022)	Analog IC + FPGA + PC	Filtering + Spike Detection	12.8 kHz	N/A	4	Implementation of analog front-end interface that uses common average referencing algorithm to enhance the signal to noise ratio.
(Saggese & Strollo, 2022)	ASIC	Digital Filtering + Spike Detection	10 kHz	N/A	1024	Implementation of low-power spike detection ASIC that uses nonlinear energy operators to distinguish neural spikes from background noise.
This Work, 2023	FPGA	Digital Filtering + Spike Detection	7 KHz	4.815 ns – 7.665 ns Per channel	N/A	Implementation of MEA processing systems based on approximate computing algorithms to reduce the processing latency and system area.

## Chapter 2: Methods

This chapter presents the research design of the developed real-time processing system based on approximate computing.

### 2.1 Research Design

Figure 1 depicts the design steps of this research.

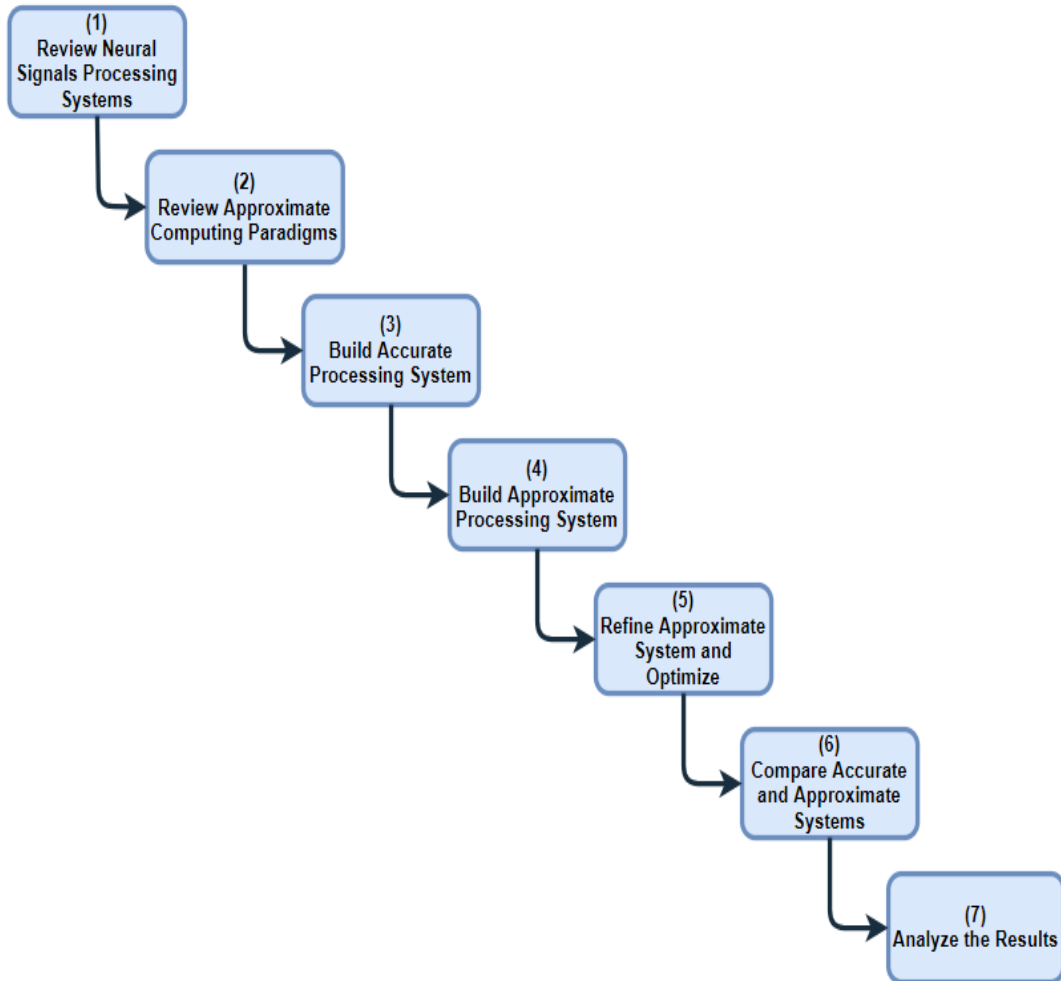


Figure 1: Research Design

The first step is to review the literature and study the different designs of neural signal processing systems since the following steps need to understand the processing system and its various parts.

Approximate computing approaches and circuits are also reviewed. Different approximate adders are reviewed and tested to pick and use the most appropriate adders

during the rest of this work. The selection has been primarily based upon the parallelism, flexibility, and area of the approximate adder circuit to take advantage of the FPGA device used.

The next step is to build an accurate processing system and test it on real biological signals. The outcomes of this step are used later as a reference for comparisons. The accurate system is built entirely using Verilog hardware description language and implemented on FPGA, as will be discussed later in the implementations chapter.

The approximate adders selected in the second step are used in building different versions of the approximate processing system, where each version is based on one of the approximate adders.

All parts of the approximate system are built identically to the accurate one except for using approximate computing circuits in their calculations. The last point is considered to maintain scientific and fair comparisons through the research.

Multiple approximation algorithms are developed, refined, and applied to the approximate system versions. Then both systems, accurate and approximate, are compared to check the validity and usefulness of the newly built approximate system.

Finally, the comparison data are collected in tables and illustrated visually in separate graphs to summarize the results of this research and extract the conclusions.

## **2.2 System Overview**

The basic block diagram for the targeted system used to process neural signals is shown in Figure 2. It consists of MEA acquisition unit, signal processing unit, and an offline processing and storage unit.

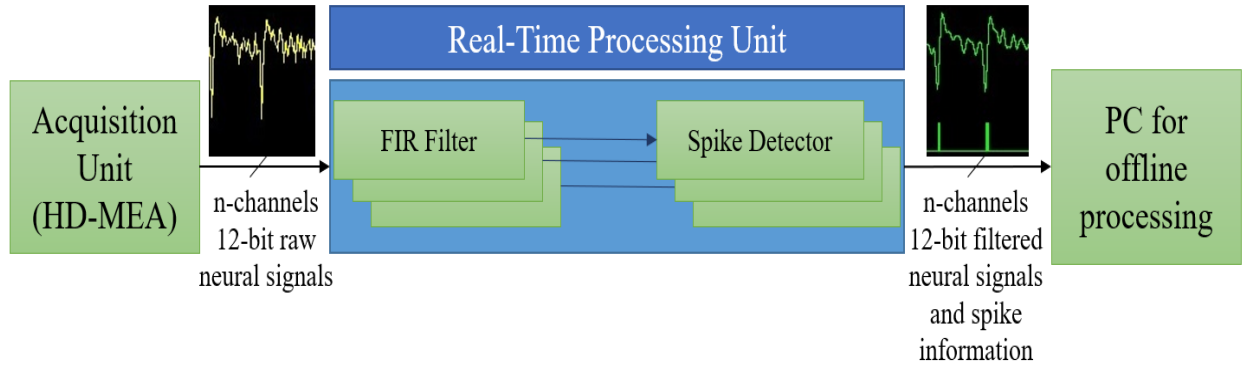


Figure 2: General block diagram for the neural processing system

The acquisition unit can be any commercial CMOS-MEA device that acquires the neural signal and outputs amplified and digitized raw neural data at a certain resolution, 12-bit for example.

CMOS technology is used in recent MEA devices to integrate active electronic components on the same substrate of the electrodes. This technology allows the transfer of data from a high number of electrodes and adds more functions to the MEA device such as amplifying the signal, analog to digital conversion, chip identification, and closed loop capabilities (Obien et al., 2017).

The output neural data from MEA is continuously sent to the real-time processing unit through multiple channels. The system's heart is the real-time processing unit that can be implemented on a programmable device, as mentioned before in the literature review.

The processing unit consists of multiple sets of filtering and spike detection modules. All sets work in parallel, where each set processes one MEA channel. Processing starts with the filtering of the amplified and digitized raw neural signal received from the MEA device in the filtering module, then the filtered signal is forwarded to the spike detection module. The spike detection module detects and counts spikes in the signal at the point where it satisfies a particular criterion depending on the detection method. A spike is detected for instance when the signal sample is below a certain negative threshold (Rey et al., 2015). It then pulls the spike detection output to a high level for a certain period to alert for spike detection. Estimating the correct

threshold value is essential since a low threshold value leads to false detections while a high value leads to an increased number of missed spikes.

An offline processing unit, such as a PC workstation, can be used to visualize and analyze the processed neural signal and spikes.

In this work, two types of systems are developed and compared: the accurate system, which uses precise calculations in all processes, and the new proposed approximate system, which employs various approximate adders in processing the acquired signals. The implementation of all processing systems is described in detail in Chapter 3.

### **2.3 Proposed Approximate Processing System**

This research aims to enhance the neural signals processing system by enhancing each filtering and spike detection set in terms of time, area, and power. Less circuit area will allow the processing system to contain more parallel filtering and spike detection sets to handle more MEA channels in parallel. In contrast, less processing delay will enhance the overall delay of the system and allow it to filter the neural signal and detect the spike with minimum latency.

Multiple design points have been considered and applied in the design of the proposed approximate system, such as:

1. Using FPGA for implementation.
2. Enhancing the filtering process.
3. Applying approximate computing algorithms.
4. The following sections describe each of the previous points in detail.

#### ***2.3.1 Using FPGA for Implementation***

Processing the biological signals acquired from MEA devices requires a high number of input channels, parallelization, speed, and reliability. The processing device should also be energy efficient and small in size. Other design factors are the flexibility in programming the device and reconfigurability options.

The Field Programmable Gate Array (FPGA) is the best programmable device that can satisfy the previous requirements. The FPGA is an integrated semiconductor device that may be programmed any time after production. It is made up of a matrix of Configurable Logic Blocks (CLB) that are linked together according to a program. Languages like VHDL and Verilog are employed to produce the higher language code for FPGA programming. Code flashing is relatively simple and similar to PROM flashing (Andina et al., 2020). The conceptual structure of FPGA is shown in Figure 3. It consists of a programmable collection of logic cells, interconnections, switches, and I/O blocks. Logic cells consist of look-up tables and D-flip flops, as shown in Figure 4. Interconnect fabric surrounds these logic cells.

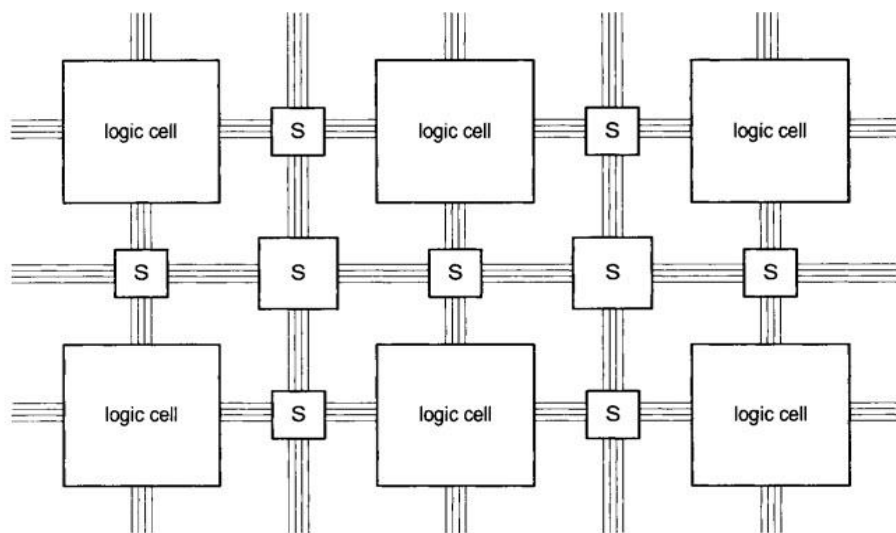


Figure 3: Conceptual Structure of FPGA (Andina et al., 2020)

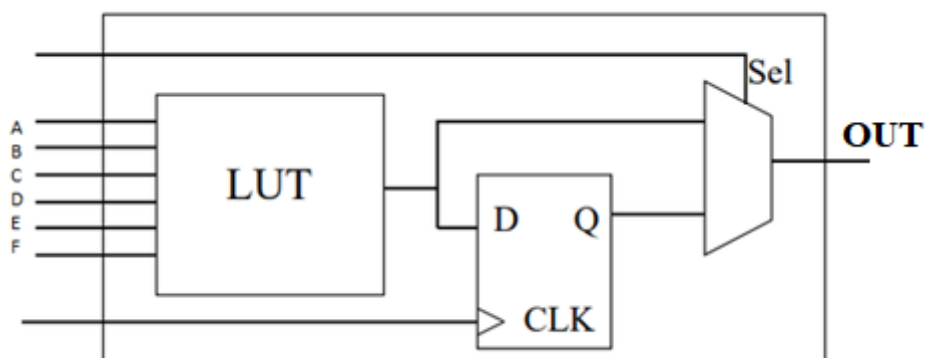


Figure 4: Logic cell Conceptual Structure (Andina et al., 2020)

FPGA device is selected for implementation in this research since it provides the advantages of better performance in terms of speed and power consumption, programmability, and a more straightforward design cycle. The design tools take care of the significant functions by themselves, including placement, routing, and timing in reference to the specifications. Parallel task performance is a significant advantage of FPGAs, where they can be designed to include multiple blocks which process data in parallel. Also, because of their highly efficient processing architecture, FPGAs are ideal for time-critical systems. They can process more data in a shorter duration of time than other choices on the market, making them ideal for real-time applications (Perepelitsyn & Kulanov, 2022).

The board used as a target board to implement the processing unit is the ZedBoard development board for the Xilinx Zynq-7000, shown in Figure 5.

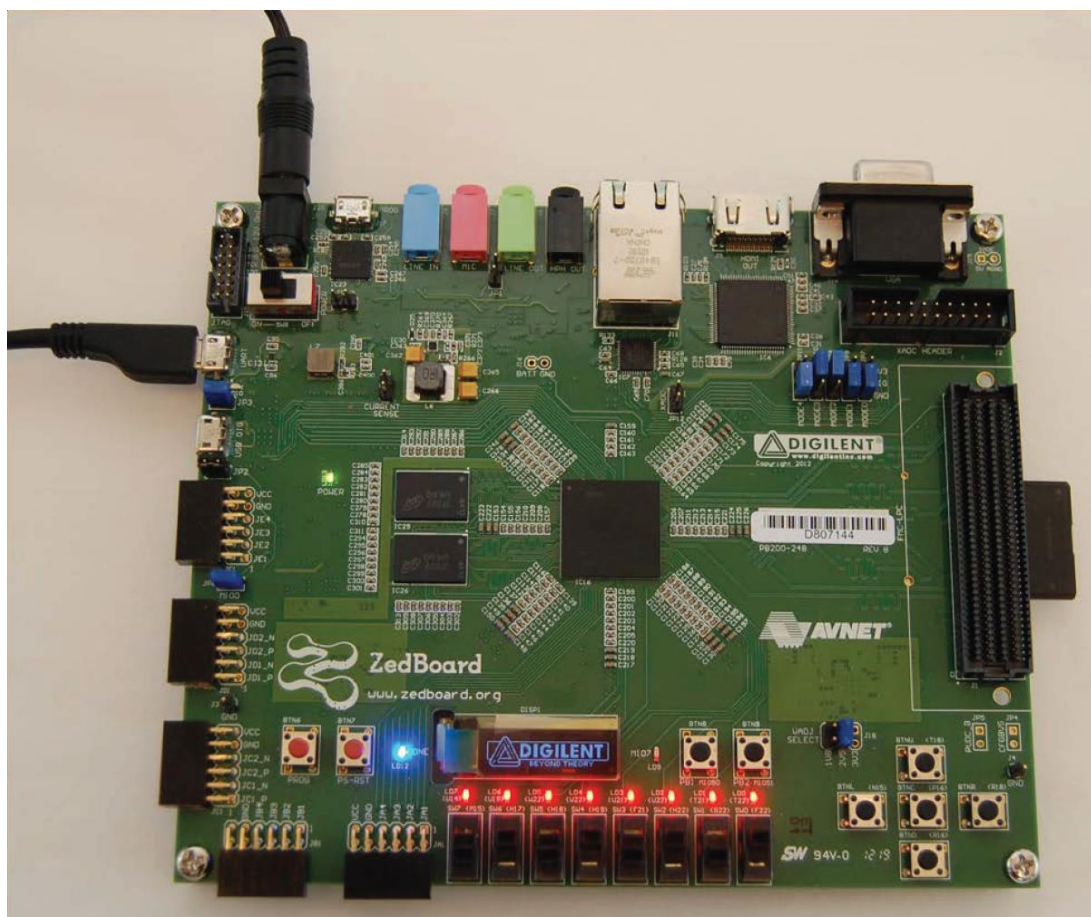


Figure 5: ZedBoard Development Board



ZedBoard is a comprehensive development kit for exploring designs using the Xilinx Zynq-7000 All Programmable SoC. The board is equipped with all the essential interfaces and functionality to support a wide range of applications. It features an XC7Z020-CLG484-1 Xilinx chip, 512 MB DDR3 and 256 Mb Quad-SPI Flash memories, eight user switches, and seven user push buttons, among other high specifications and connection options (ZedBoard, 2023).

The proposed processing system is programmed with Verilog hardware description language using Xilinx Vivado Design Suite. The suite offers synthesis and implementation of Hardware Description Language (HDL) designs, which replaces Xilinx ISE and adds functionality for the system on chip development and high-level synthesis. It also supports simulation and debugging of the hardware program modules (O'Loughlin et al., 2014). Figure 6 shows the Vivado Design Suite interface with the RTL project selected.

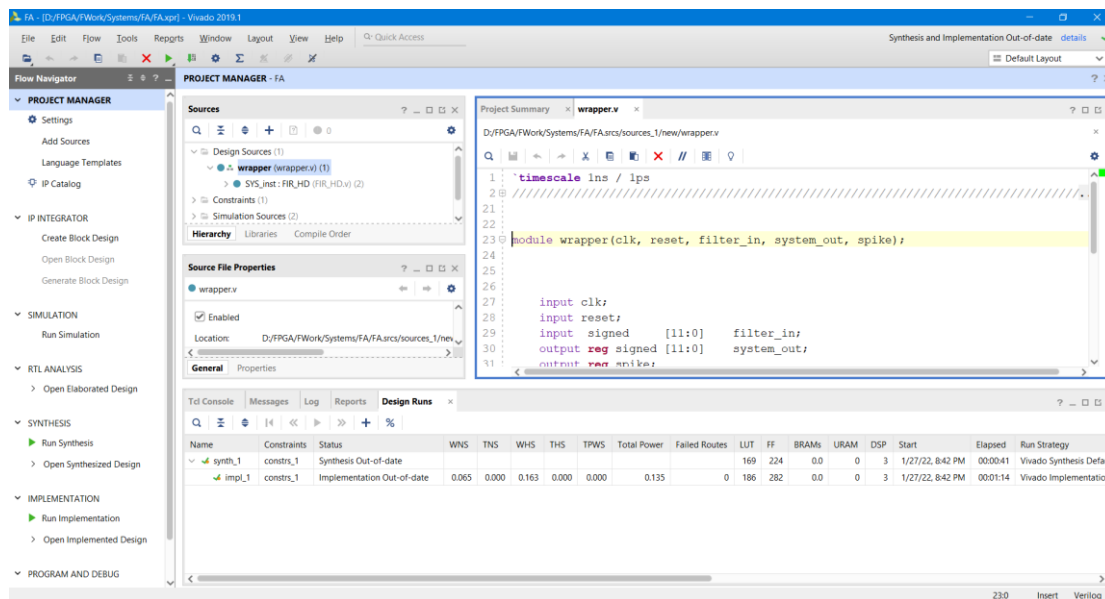


Figure 6: Vivado Design Suite interface

### 2.3.2 Enhancing the Filtering Process

Choosing the specifications of the filters is a critical factor in enhancing the processing system's performance. The processing unit acquires the digitized signals from the MEA with 12-bit resolution. It feeds them to low-pass filters to isolate typical frequency components of neural spikes.

The recorded spiking activity contains biological noise produced by neurons adjacent to the recording electrode called population spikes, in addition to the thermal and high noise power coming from the analog front end. The noise and the weak capacitive coupling between the nearby neurons and the sensor limit the signal to noise ratio of the electrodes to 3 - 6 dB (Saggese et al., 2021).

To preserve the spiking activity in the neural signal while removing high frequency noises, each filter implements a Finite-Impulse Response (FIR) low pass filter, designed with a cut-off frequency of 1 kHz. The cut-off frequency of the filter can be configured by setting the FIR filter coefficients to support various signal types.

This work uses the FIR filter for its many advantages over the IIR filter. The impulse response (or reaction to any input of limited length) of this filter has a finite duration since it settles to zero in a finite amount of time. This response differs from Infinite Impulse Response (IIR) filters, which can have internal feedback and respond forever (usually decaying) (Pal, 2017). FIR filter is a digitally implemented filter structure that may implement practically any frequency response. It is also the most popular type of filter implemented in software. These filters are typically built using a series of delays, multipliers, and adders to construct the filter's output.

Linear phase is another advantage of the FIR filters. As all frequencies are shifted in time by the same amount, no phase distortion is introduced into the signal to be filtered, preserving their respective harmonic connections (i.e., constant group and phase delay). Maintaining the phase is not valid for IIR filters, which have a non-linear phase characteristic. FIRs also have no feedback and can never become unstable for any form of an input signal, giving them an advantage over IIR filters which is stability (Ye et al., 2022).

The output of the FIR filter with order  $N$  is a convolution of input signal  $x[n]$  and coefficients  $h[n]$ . It is a sequence of values where each value is the weighted sum of the most recent input values, as shown in Equation 2.1.

$$y[n] = \sum_{i=0}^N h_i \cdot x[n - i] \quad (2.1)$$

$x[n - i]$  in each input sample belongs to the sliding window of the last  $N$  received samples. Each sample is weighted by coefficient  $h_i$  and then added together to obtain the output sample  $y[n]$ .

#### 2.3.2.1 Filter Order

The order of the filter is another critical parameter that must be considered during the filter design. Higher filter order provides better low pass selectivity but will require higher resources and longer processing time. This work aims to design a real-time processing system with minimal latency and resource usage. Thus, we designed FIR filters with different orders of 4, 8, 12, 16, and 24, then tested them with the accurate system on real biological signals with around 106,000 samples to precisely detect all spikes. Eventually, we decided to use the FIR with the order of  $N=8$ , as it is the minimum order required to detect precisely all spikes in the accurate system. This filter uses the last nine samples to produce the new filtered sample. The filter order selection will be described more in chapter 3.

#### 2.3.2.2 Filter Symmetry

Equation 2.1 shows that the FIR filter needs many additions and multiplication operations. Therefore, symmetric coefficients are used to reduce the number of calculations. Using symmetric coefficients minimizes the number of operations by almost 50%. Every two opposite samples are added, then the addition result is multiplied by the coefficient and then accumulated. Since the filter features linear phase response characteristics, distortion of the original signal is avoided, and the shape of the spike is kept.

#### 2.3.2.3 Parallelization

The required FIR filter is built on FPGA using Verilog hardware description language targeting Xilinx Zynq-7000 All Programmable SoC ZedBoard development board.

Parallelization is one of the most significant advantages of FPGAs (Matyukha et al., 2022). Therefore, the FIR filter circuit is broken into separate blocks that run in parallel. Parallelization is also considered when programming each particular block, where non-blocking assignments are used whenever possible. The non-blocking assignment allows assignments to be scheduled without blocking the execution of the following statements and is specified by a  $\leq$  symbol.

Figure 7 shows the schematic diagram of the FIR filter used in this work's accurate and approximate processing systems.

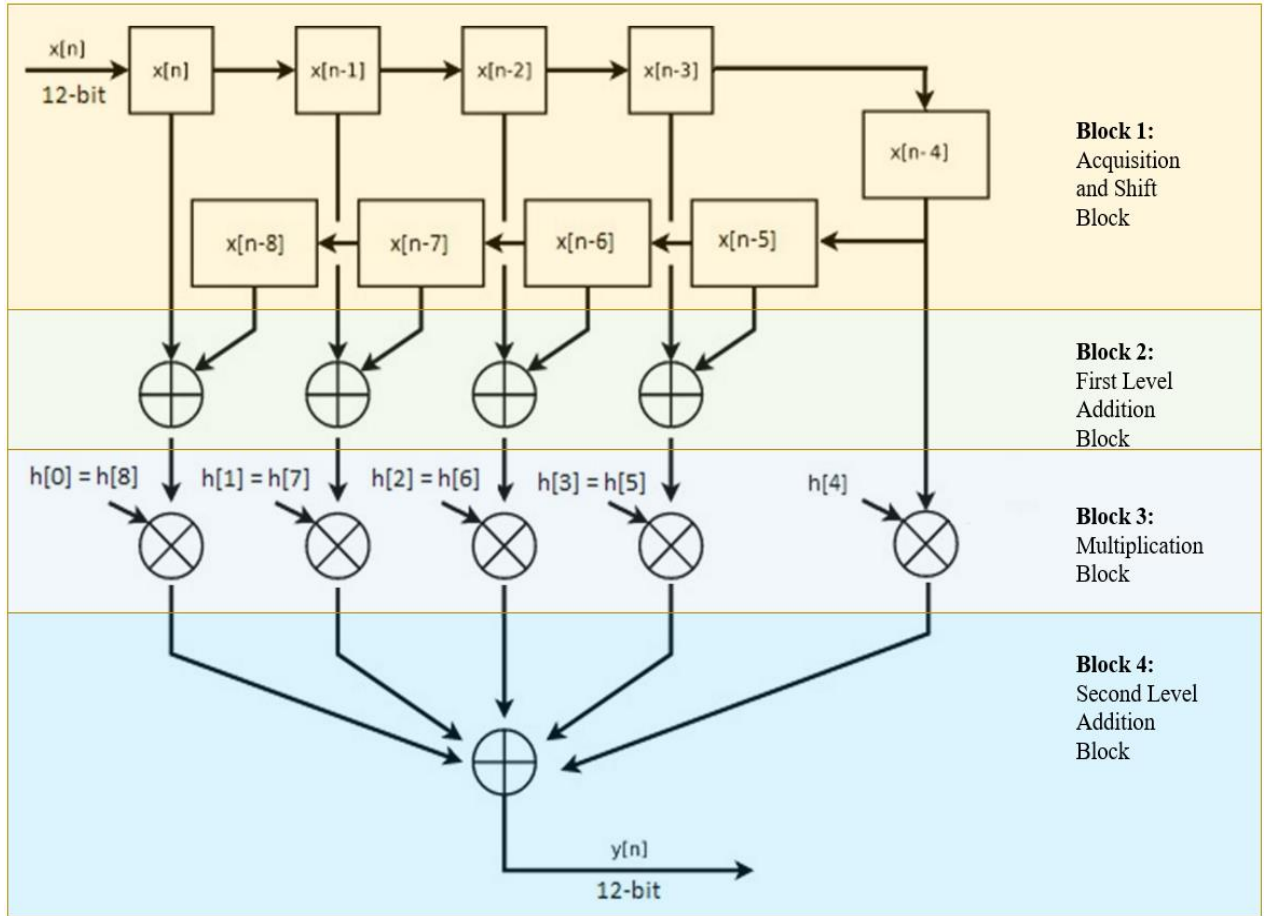


Figure 7: Symmetric FIR filter with order 8

### 2.3.3 Applying Approximate Computing Algorithms

The mathematical model of the FIR filter in Equation 2.1 shows that addition is the most frequent operation compared to multiplication. The number of additions required depends on the order of the filter  $N$ . Other essential factors affecting the workload value are the sampling frequency and the number of channels, which are some

of the MEA device specifications. Every two opposite samples in a symmetric FIR filter are added together, and the result is multiplied by a coefficient and then accumulated with other multiplication results. The addition operation is used more intensively in FIR filters, and its workload can be described by Equation 2.2, Where  $N$  is the order of the filter and  $f_s$  is the sampling frequency in Hz.

$$\text{Addition Workload} = f_s \cdot N \quad (2.2)$$

For our symmetric FIR filter of order 8 in this work and a sampling frequency of 7 kHz for the sampled signals, the number of addition operations is 56,000 per second for each MEA channel. Therefore, reducing the time required to process the addition operations can significantly improve the overall system's performance.

We chose to focus on approximating the addition operation to prove the concept of this research since it is more extensively used in the FIR filter than multiplication operation. More enhancements in the system parameters can be expected when approximating the multiplication operation too, but on the other hand, more approximate operations will impose the use of error correction techniques which will require more processing time, power and circuit area.

Approximate computing algorithms can achieve high performance by tolerating some quality loss (Huang et al., 2021). In this context, we applied approximate computing in the most computationally intensive system parts using three different types of adders in different modes. We tested the implementations with real biological signals to compare the results with the accurate system in terms of delay time, area, and power.

The adder is one of the primary circuits used in fault-tolerant applications. It is essential to evaluate how quickly and how much power a Digital Signal Processing (DSP) system uses. The development of approximation adders has been aided by the requirement for high speed and power efficiency and the fault tolerance aspect of applications (Bhargav & Huynh, 2021). Relaxing the constraints of exact computation opens new possibilities with potential performance, area, and power gains of orders of magnitude (Bosio et al., 2017).

Many approximation approaches have been offered by lowering the accurate adder's critical route and hardware complexity. In general, approximate approaches can be classified into two categories (Priyadharshni & Kumaravel, 2019).

#### *2.3.3.1 Gate Level Approximation*

In this approach, the accurate n-bit ripple carry adder is designed for varied bit widths and various levels of approximation. The approximate full adders calculate the carry and sum of the Less Significant Bits (LSB) using simpler circuits and gates. Carry Prediction Adder (CPredA) (Sato et al., 2019), and Approximate Adder 2 (AA2) (Gorantla & Deepa, 2020) are examples of approximate full adders.

Gracefully Degrading Adder (GDA) (Ye et al., 2013) and Almost Correct Adder (ACA) (Verma et al., 2008) are other examples of this approach. GDA and ACA utilize the hierarchical carry look-ahead structure for the carry prediction, which makes their circuits large and power-consuming.

Generic Accuracy Configurable Adder (GeAr) (Raghuram & Shashank, 2022) uses this approach to split an n-bit ripple carry adder into multiple smaller sub-adders. These sub-adders run in parallel with fixed carry inputs. Therefore, the carry propagation chain is truncated into shorter segments, reducing the delay in calculations.

#### *2.3.3.2 Transistor Level Approximation*

Construction of approximate adder cells is done at the transistor level in this approach. This approach reduces the conventional ripple carry adder circuit complexity at the transistor level to enable a shorter critical path and voltage scaling. Approximate XOR-based Adder 1 (AXA1) and Approximate XNOR-based Adder 2 (AXA2) (Yang et al., 2013) are examples of transistor level approximation. The number of transistors is reduced from 10 in the accurate full adder to 8 in AXA1 and 6 in AXA2.

Many approximate adders are available in the literature. Transistor level approximation is not applicable in this research. Therefore, the proposed approximate processing system uses three adders with gate level approximation approach: CPredA, AA2, and GeAr.

The selection of the adders has been primarily based upon the parallelism in producing the result, the flexibility of combining the adder with accurate adders to control the precision, and the area of the adder circuit to utilize the advantages of FPGA and reduce the system size.

CPredA adder is chosen as an example of the adders which approximate the output carry in their algorithms. At the same time, AA2 is chosen as an example of the adders which approximate the output sum in their algorithms. GeAr is an example of the adders which use segmentation in their algorithms.

Chapter 3 includes more details about the selected approximate adders, their designs, and their implementations.

## Chapter 3: Implementation

Two types of processing systems are implemented and compared in this work: the accurate system, which uses precise full adders in all processes, and the new proposed approximate system, which uses different types of approximate adders at various levels of approximation. Three versions of the approximate system are implemented based on each of the approximate adders selected (CPredA, GeAr, and AA2).

The implementation of the proposed approximate processing system starts with creating the basic blocks of the selected approximate adders. These basic blocks are then used to build higher-order adders with widths of 8, 12, 24, and 32-bits. They are then tested at different approximation levels to investigate and benchmark their performance against the full adder.

12 and 24-bit approximate adders are used to evaluate the filtering process before building the complete system. The error in the produced filtered signal at different approximation levels is calculated with reference to the accurate one. For benchmarking with the accurate system and conclusion, the approximate system is run and tested in all three versions on real biological signals.

### 3.1 Implementation of Approximate Adders

Three approximate adders are selected and implemented in this work as mentioned in chapter 2:

- Carry Prediction Full Adders (CpredA)
- Approximate Adder (AA2)
- Generic Accuracy Configurable Adder (GeAr)

CPredA is an example of the adders which approximate the output carry in their algorithms. AA2 is an example of the adders with approximated sum. At the same time, GeAr is an example of the adders which use segmentation in their algorithms.

The three adders also have the advantage of producing their results parallelly, reducing the time consumed in calculations and utilizing the FPGA device. The adders



are also serving other research purposes by reducing the system area since their circuits are smaller than the full adder circuit and consume less or similar power.

### 3.1.1 Carry Prediction Full Adder (CPredA)

Figure 8 shows the circuit diagram of CPredA. Sum (S) and Carry (Cout) are produced as shown in Equations 3.1 and 3.2:

$$S = C_{in} \oplus (\overline{A \cdot B}) \cdot (A + B) \quad (3.1)$$

$$C_{out} = A \cdot B \quad (3.2)$$

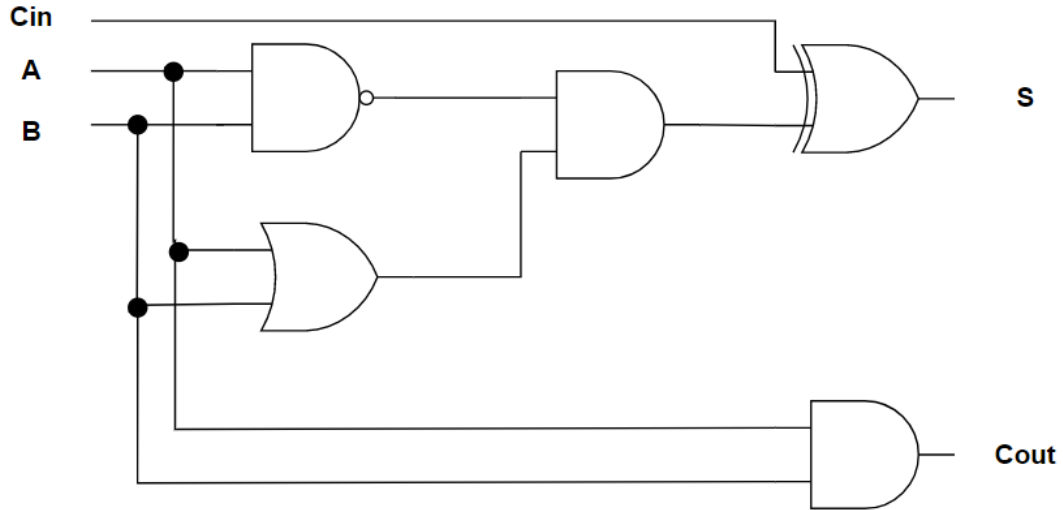


Figure 8: Carry-Prediction Full Adder (CPredA)

Table 2 compares the output of Full Adder (FA) and CpredA in prediction state. CpredA always generates the correct S and two incorrect Cout values when Cin is 1.

Table 2: Truth Table of FA and CPredA

Cin	A	B	FA		CPreadA	
			Cout	S	Cout	S
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	1	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

The architecture of the CPredA adder shows that Cout values are predicted almost correctly depending only on the current values of A and B without using the carry chain. The last point is critical when using this adder in FPGA-based systems, where parallelism is one of its main advantages. Flexibility is another significant feature of CPredA. Higher order adders can be built by combining CPredA with accurate full adders in any configuration, controlling the accuracy to the required level. CPredA also

reduces the circuit area required by implementing more straightforward logic to produce its results.

The logical circuit of CPredA is programmed on the Vivado tool using the Verilog language code in Figure 9.

```

module CpredA(

    input A,
    input B,
    input Cin,
    output S,
    output Cout

);

    assign S    = Cin^(!(A&B) & (A|B));
    assign Cout = A&B;

endmodule

```

Figure 9: CPredA logical circuit code

Several instances of the previous basic block are cascaded according to the width and approximation level required to build higher order adders of 8, 12, 24, and 32-bit. For example, eight instances are cascaded to build an 8-bit CPredA that fully predicts the result, as illustrated in Figure 10.

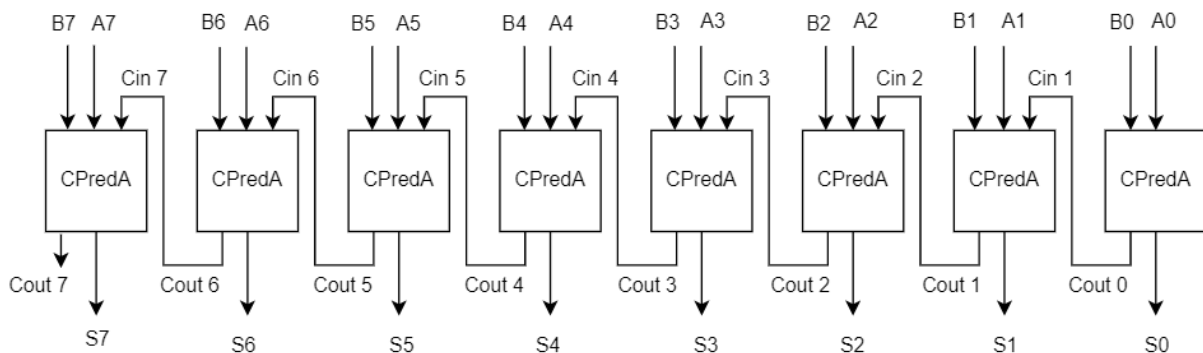


Figure 10: 8-bit CPredA in full prediction configuration

The previous 8-bit CPredA adder in Figure 10 is created using the Verilog code in Figure 11.

```

module CpredA_bitsAdder(
    input    signed [7:0] A,
    input    signed [7:0] B,
    output   signed [7:0] S
);

    wire [8:0] repCarry;

    genvar i;
    assign repCarry[0]=0;

    generate
        for(i=0;i<8;i=i+1) begin
            CpredA cells(A[i],B[i],repCarry[i],S[i],repCarry[i+1]);
        end
    endgenerate

endmodule

```

Figure 11: Verilog code for 8-bit CPredA in full prediction configuration

The approximation level is controlled by combining CPredA instances with full adders. For the same previous 8-bit adder, if the required approximation level is half the result, then four CPredA instances can be combined with four full adder instances. Thus, the lower nibble of the result will be predicted by CPredA, while full adders will calculate the upper nibble. Figure 12 illustrates the 8-bit CPredA adder in half prediction configuration.

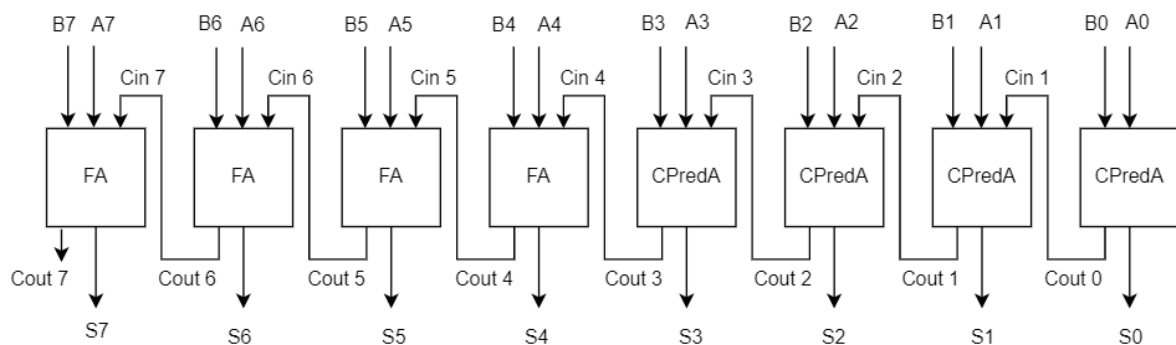


Figure 12: 8-bit CPredA in half prediction configuration

The 8-bit CPredA in half prediction configuration is expressed in Verilog with the code in Figure 13.

```

module CpredA_bitsAdder(
    input    signed [7:0] A,
    input    signed [7:0] B,
    output   signed [7:0] S
);

    wire [8:0] repCarry;

    genvar i;
    assign repCarry[0]=0;

    generate
        for(i=0;i<4;i=i+1) begin
            CpredA cells(A[i],B[i],repCarry[i],S[i],repCarry[i+1]);
        end
    endgenerate

    generate
        for(i=4;i<8;i=i+1) begin
            FullAdder cells(A[i],B[i],repCarry[i],S[i],repCarry[i+1]);
        end
    endgenerate

endmodule

```

Figure 13: Verilog code for 8-bit CPredA in half prediction configuration

### 3.1.2 Approximate Adder (AA2)

Figure 14 shows the circuit diagram of the AA2 adder. The design approach of AA2 is approximated on the Sum (S) alone.

The Sum (S) and the Carry (Cout) can be calculated using Equations 3.3 and 3.4 as follows:

$$C_{out} = (A.B) + (A.C_{in}) + (B.C_{in}) \quad (3.3)$$

$$S = \overline{C_{out}} \quad (3.4)$$

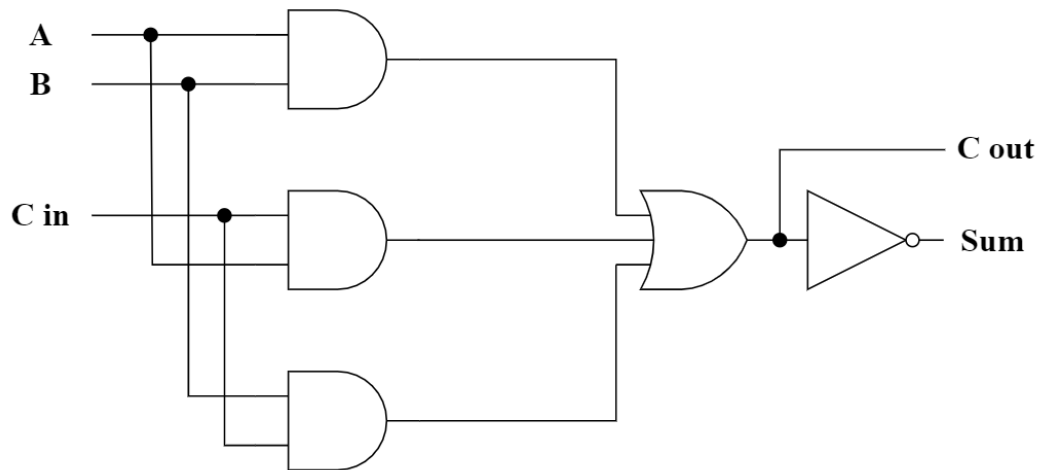


Figure 14: Approximate Adder (AA2)

The Sum is precise in 6 out of 8 cases, and Cout is precise in all cases in AA2, as shown in Table 3.

Table 3: Truth Table of FA and AA2

Cin	A	B	FA		AA2	
			Cout	S	Cout	S
0	0	0	0	0	0	1
0	0	1	0	1	0	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	1	0	1	0
1	1	0	1	0	1	0
1	1	1	1	1	1	0

The advantage of AA2 is that the sum is easily produced by inverting the value of the carry signal, which reduces the logic complexity of the adder and the area needed. Flexibility is also another advantage. Like previous adders, accuracy can be controlled by combining it with accurate adders.

The logical circuit of AA2 is expressed using Verilog language with the code in Figure 15.

```

module AA2 (
    input A,
    input B,
    input Cin,
    output S,
    output Cout
);

    assign S = ~Cout;
    assign Cout = (A&B) | (B&Cin) | (A&Cin);

endmodule

```

Figure 15: Verilog code for AA2 logical circuit

Several instances of AA2 basic block are cascaded according to the width and approximation level required. Figure 16 illustrates an 8-bit AA2 adder that fully predicts the addition result. Eight instances of AA2 are cascaded to build this adder.

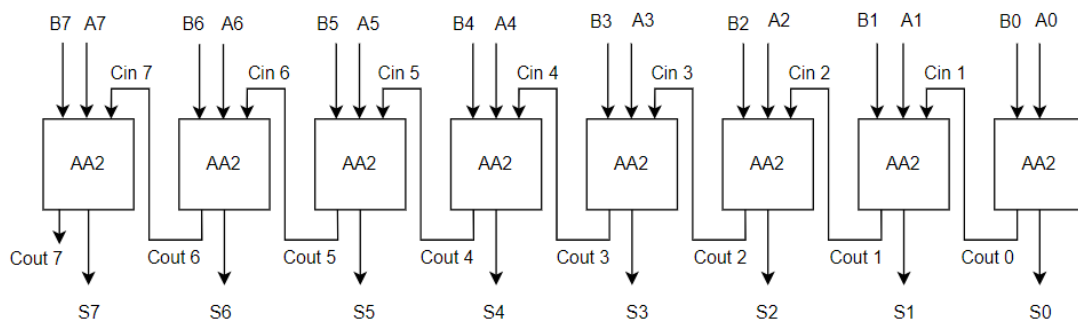


Figure 16: 8-bit AA2 in full prediction configuration

The previous 8-bit AA2 adder is created using the Verilog code in Figure 17.

```
module AA2_bitsAdder(
    input    signed [7:0] A,
    input    signed [7:0] B,
    output   signed [7:0] S
);

    wire [8:0] repCarry;

    genvar i;
    assign repCarry[0]=0;

    generate
        for(i=0;i<8;i=i+1) begin
            AA2 cells(A[i],B[i],repCarry[i],A[i],repCarry[i+1]);
        end
    endgenerate

endmodule
```

Figure 17: Verilog code for 8-bit AA2 in full prediction configuration

The accuracy of calculations is also controlled by combining AA2 instances with full adders according to the required level. For half prediction accuracy in the previous 8-bit adder, four AA2 instances will predict the lower nibble of the result, while four full adders will predict the upper half. Figure 18 shows the 8-bit AA2 adder in half prediction configuration.



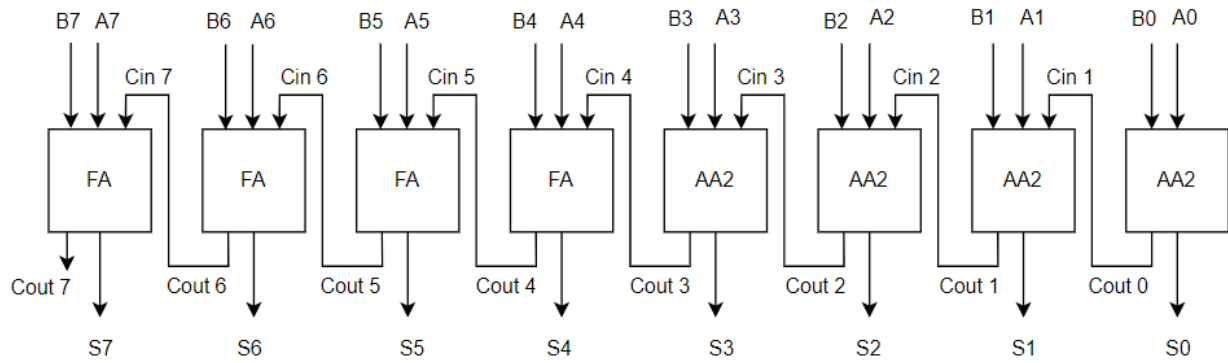


Figure 18: 8-bit AA2 in half prediction configuration

The Verilog code in Figure 19 expresses the AA2 adder in half prediction configuration.

```

module AA2_bitsAdder(
    input  signed [7:0] A,
    input  signed [7:0] B,
    output signed [7:0] S
);

    wire [8:0] repCarry;

    genvar i;
    assign repCarry[0]=0;

    generate
        for(i=0;i<4;i=i+1) begin
            AA2 cells(A[i],B[i],repCarry[i],S[i],repCarry[i+1]);
        end
    endgenerate

    generate
        for(i=4;i<8;i=i+1) begin
            FullAdder cells(A[i],B[i],repCarry[i],S[i],repCarry[i+1]);
        end
    endgenerate

endmodule

```

Figure 19: Verilog code for 8-bit AA2 adder in half prediction configuration

### 3.1.3 Generic Accuracy Configurable Adder (GeAr)

GeAr adder breaks the carry chain of the full adder by using K L-bit sub-adders to perform the approximate addition of N-bit length operands. The length of each sub-adder (L) is less than or equal to the size of each operand (N). Each sub-adder produces an R-bit result depending on the number of previous bits (P) used for the carry prediction except the first sub-adder, which produces an L-bit result where  $L=R+P$ . The number of required sub-adders K can be calculated using Equation 3.5:

$$K = ((N - L) / R) + 1 \quad (3.5)$$

The result of the first sub-adder can be calculated using Equation 3.6:

$$\text{Sum}[L - 1:0] = A[L - 1:0] + B[L - 1:0] \quad (3.6)$$

The result of the *i*th sub-adder can be calculated using Equation 3.7:

$$\begin{aligned} \text{Sum}[(R \times i) + P - 1 : R \times (I - 1) + P] = & A[(R \times i) + P - 1 : R \\ & \times (I - 1)] + B[(R \times i) + P - 1 : R \times (I - 1)] \end{aligned} \quad (3.7)$$

Figure 20 **Error! Reference source not found.** shows the 12-bit GeAr adder in the R2-P2 and full prediction configurations, where SR is the sub-result of each sub-adder.

A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
								Sub-Adder 1			
								SR1 [3]	SR1 [2]	SR1 [1]	SR1 [0]
								Sub-Adder 2			
								SR2 [3]	SR2 [2]	SR2 [1]	SR2 [0]
								Sub-Adder 3			
								SR3 [3]	SR3 [2]	SR3 [1]	SR3 [0]
								Sub-Adder 4			
								SR4 [3]	SR4 [2]	SR4 [1]	SR4 [0]
								Sub-Adder 5			
								SR5 [3]	SR5 [2]	SR5 [1]	SR5 [0]
SUM [11]	SUM [10]	SUM [9]	SUM [8]	SUM [7]	SUM [6]	SUM [5]	SUM [4]	SUM [3]	SUM [2]	SUM [1]	SUM [0]

Figure 20: 12-bit GeAr adder in R2-P2 and full prediction configurations

In this configuration:

- The adder length (N) = 12 bits.
- The length of each sub-adder (L) = 4 bits.
- The first sub-adder contributes to the final sum's first four bits.
- Each sub-adder, except the first one, produce 2 bits of the final result depending on the last 2 bits. Therefore, R = 2 and P = 2.
- The number of required sub-adders (K) is  $((12 - 4) / 2) + 1 = 5$  sub-adders.
- The final sum can be formed as:

$$\text{Sum} = [ \text{SR5} [3:2], \text{SR4} [3:2], \text{SR3} [3:2], \text{SR2} [3:2], \text{SR1} [3:0] ].$$

All sub-results in GeAr are produced in parallel and combined to create the final sum. Compared to an N-bit full adder, the delay is reduced by breaking the carry chain into smaller segments. Thus, the carry propagation will only be limited to the length of the segment represented by L.

GeAr R2-P2 configuration described before is used to build the adders in different widths since it offers a high degree of flexibility in controlling the adder accuracy and low latency due to the small sub-adder length.

The Verilog code in Figure 21 is used to implement the 12-bit GeAr R2-P2 adder in full prediction configuration.

```

module GeAr_bitsAdder(

    input    signed [11:0] A,
    input    signed [11:0] B,
    output   signed [11:0] SUM
);

    wire [3:0] SR1,SR2,SR3,SR4,SR5;

    FA_4bit  sub_adder1 (A[3:0],B[3:0],SR1[3:0]);
    FA_4bit  sub_adder2 (A[5:2],B[5:2],SR2[3:0]);
    FA_4bit  sub_adder3 (A[7:4],B[7:4],SR3[3:0]);
    FA_4bit  sub_adder4 (A[9:6],B[9:6],SR4[3:0]);
    FA_4bit  sub_adder5 (A[11:8],B[11:8],SR5[3:0]);

    assign SUM[11:0] = {SR5[3:2],SR4[3:2],SR3[3:2],SR2[3:2],SR1[3:0]};

endmodule

```

Figure 21: Verilog code for 12-bit GeAr R2-P2 adder in full prediction mode

The full adder is broken into five segments, where each segment performs the addition of 4 bits of the operands. The sum bits are assigned as illustrated in **Error! Reference source not found.** to obtain the approximated result.

Controlling the result accuracy in GeAr can be done by combining it with longer segments of the accurate adder. If the upper six bits of the result are required to be calculated accurately in the previous 12-bit adder, then two 4-bit sub-adders can be combined with a 6-bit full adder, as illustrated in Figure 22.

A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0	
								Sub-Adder 1				
								SR1 [3]	SR1 [2]	SR1 [1]	SR1 [0]	
								Sub-Adder 2				
						SR2 [3]	SR2 [2]	SR2 [1]	SR2 [0]			
						6-bit Full Adder						
FR[5]	FR[4]	FR[3]	FR[2]	FR[1]	FR[0]							
SUM [11]	SUM [10]	SUM [9]	SUM [8]	SUM [7]	SUM [6]	SUM [5]	SUM [4]	SUM [3]	SUM [2]	SUM [1]	SUM [0]	

Figure 22: 12-bit GeAr in half prediction of the result

GeAr in half prediction is expressed with Verilog by the code in Figure 23.

```
module GeAr_bitsAdder(  
    input    signed [11:0] A,  
    input    signed [11:0] B,  
    output   signed [11:0] SUM  
);  
  
    wire [3:0] SR1,SR2;  
    wire [5:0] FR;  
  
    FA_4bit sub_adder1 (A[3:0],B[3:0],SR1[3:0]);  
    FA_4bit sub_adder2 (A[5:2],B[5:2],SR2[3:0]);  
  
    FA_6bit adder (A[11:6],B[11:6],FR[5:0]);  
  
    assign SUM[11:0] = {FR[5:0],SR2[3:2],SR1[3:0]};  
endmodule
```

Figure 23: Verilog code for GeAr R2P2 in half prediction configuration

### 3.2 Implementation of Processing Systems

The primary function of the processing system is to receive the digitized raw neural signals sent by the MEA device to filter them and detect the spikes. The system consists of parallel processing sets. Each processing set is dedicated to one MEA channel to perform the filtering and spike detection of the signals received through that channel. This research aims to enhance the processing system by reducing each set's processing time and circuit area. This will allow the system to perform faster and include more processing sets to handle more MEA channels in parallel.

Two types of processing systems are implemented on the FPGA device in this research, the accurate and approximate processing systems. The accurate system uses full adders, while the approximate system uses approximate adders in its calculations.

Both systems are identical in their specifications and implementations except for the type of adders used in calculations. The last point is considered to maintain consistent results and scientific benchmarking.

As illustrated in Figure 24, the processing system receives the system clock and reset signals generated by the FPGA device for synchronization. It also receives the amplified and digitized 12-bit raw neural signal generated by the MEA device and forwards it to the filtering module. The output filtered signal is then forwarded to the spike detection module, which will detect the spike and generate a pulse when the sample value received is below the threshold specified.

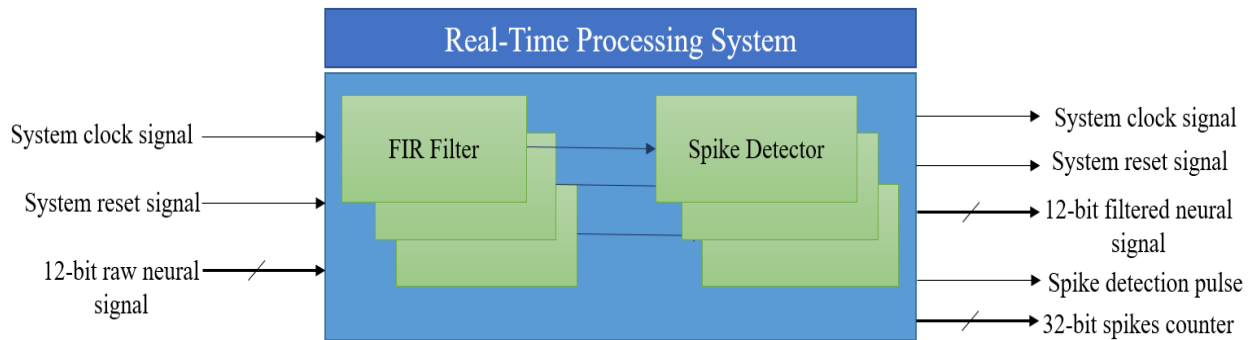


Figure 24: Detailed real-time processing system

The system outputs the processed neural signal along with the spike detection pulses and counts at its output terminal for further offline processing. It also outputs the system clock and reset signals needed for the offline device.

The following sections describe the implementation of the processing system's FIR and spike detection modules.

### 3.2.1 Implementation of FIR Module

As mentioned before in chapter 2, the FIR filter is designed with the following specifications:

- Low-pass equiripple FIR filter with a cutoff frequency of 1 kHz.
- Symmetric coefficients.
- Parallel execution of processes.

Different filters are created with orders of 4, 8, 12, 16, and 24 to select the minimum FIR order. The MATLAB Filter Design & Analysis tool is used to find the optimal set of coefficients. The coefficients are then enlarged and rounded to integer

values to be used in the Verilog programming code of the filter's implementation on FPGA.

We tested the implemented filters with different orders in the accurate processing system using real biological signals downloaded from 3Brain (3Brain, 2023), containing more than 106,000 samples. The samples are fed into the system to detect all spikes precisely. The accuracy of spike detection is then calculated based on the number of spikes detected by BrainWave X, the official software from 3Brain. Table 4 shows the accuracy results for different FIR orders.

Table 4: Spike Detection Accuracy with Different FIR Orders

Filter order	Accuracy
4	87.5%
8	100%
12	100%
16	100%
24	100%

Figure 25 shows a sample of the test done with different filter orders. It shows the misdetection of the spikes when using the FIR filter with order 4, while all higher order filters detect precisely all spikes.

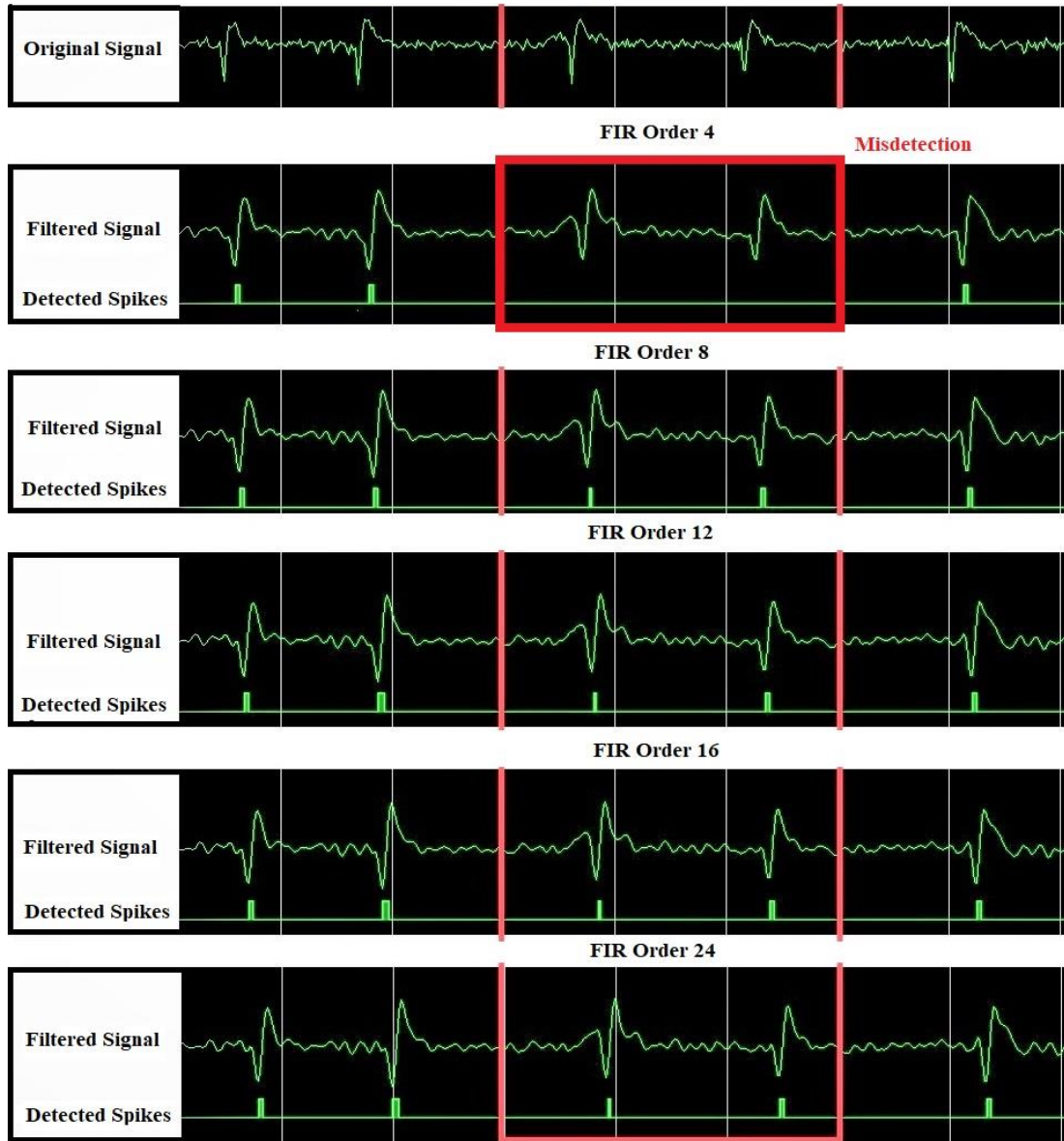


Figure 25: Misdetection of some spikes in FIR with order 4

This research aims to reduce the time and area required by the processing system. Therefore, we decided to implement the FIR filter with a minimum order of 8 in both accurate and approximate systems.

We implemented the FIR filter in Verilog by dividing it into four blocks which work in parallel at each positive edge of the clock, as illustrated in Figure 26.



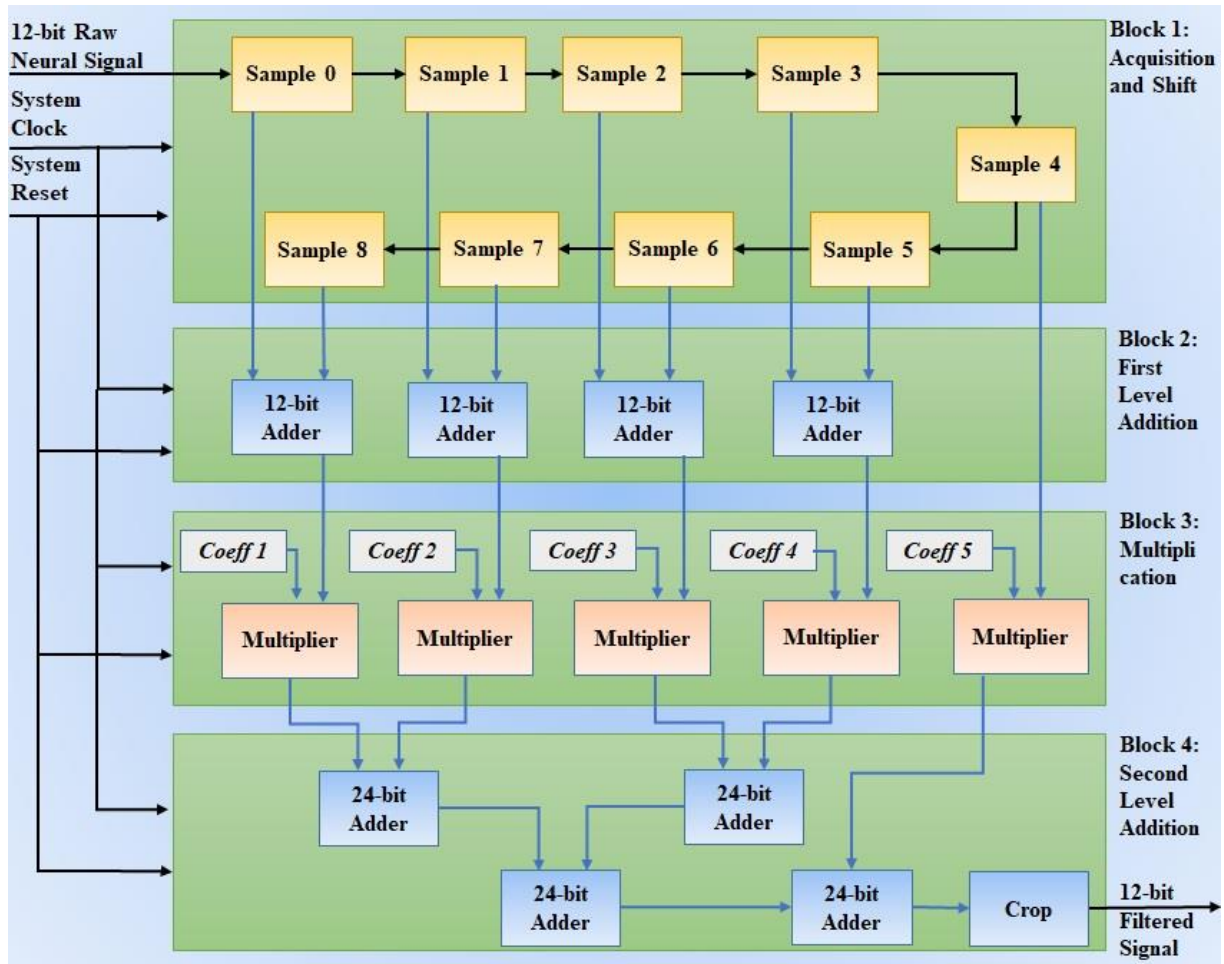


Figure 26: Implemented FIR filter

The first block acquires the new sample and assigns it to the first tab. At each clock cycle, it also shifts the old samples to adjacent tabs since it uses non-blocking assignments in its Verilog code. The Verilog code in Figure 27 is used to create the first block.

```

always@(posedge I_clk or posedge I_rst_p)
begin
    if(I_rst_p)
    begin
        sample_1 <= 'h0;
        sample_2 <= 'h0;
        sample_3 <= 'h0;
        sample_4 <= 'h0;
        sample_5 <= 'h0;
        sample_6 <= 'h0;
        sample_7 <= 'h0;
        sample_8 <= 'h0;
    end
    else
    begin
        sample_0 <= I_data;
        sample_1 <= sample_0;
        sample_2 <= sample_1;
        sample_3 <= sample_2;
        sample_4 <= sample_3;
        sample_5 <= sample_4;
        sample_6 <= sample_5;
        sample_7 <= sample_6;
        sample_8 <= sample_7;
    end
end
end

```

Figure 27: FIR filter Block 1 Verilog code

The second block of the FIR filter uses 12-bit adders to add every two opposite samples. It stores the results in the first level addition temporary registers at each positive edge of the clock. The 12-bit adders are either full adders in the case of the accurate processing system or approximate adders in the approximate system. The Verilog code in Figure 28 expresses this block in the precise system that uses full adders.

```

FA_12bit Adder0(sample_0, sample_8,tempsum_0);
FA_12bit Adder1(sample_1, sample_7,tempsum_1);
FA_12bit Adder2(sample_2, sample_6,tempsum_2);
FA_12bit Adder3(sample_3, sample_5,tempsum_3);

always@(posedge I_clk or posedge I_rst_p)
begin
    if(I_rst_p)
        begin
            add_data_0 <= 'h0;
            add_data_1 <= 'h0;
            add_data_2 <= 'h0;
            add_data_3 <= 'h0;
            add_data_4 <= 'h0;
        end
    else
        begin
            add_data_0 <= tempsum_0;
            add_data_1 <= tempsum_1;
            add_data_2 <= tempsum_2;
            add_data_3 <= tempsum_3;
            add_data_4 <= sample_4 ;
        end
    end
end

```

Figure 28: FIR filter Block 2 Verilog code

The first four lines of the previous code are changed when implementing the block in the approximate system. For instance, if the approximate system uses CPredA adder, the code in Figure 29 is used.

```

CPredA_12bit Adder0(sample_0, sample_8,tempsum_0);
CPredA_12bit Adder1(sample_1, sample_7,tempsum_1);
CPredA_12bit Adder2(sample_2, sample_6,tempsum_2);
CPredA_12bit Adder3(sample_3, sample_5,tempsum_3);

```

Figure 29: Verilog code when the approximate system uses CPredA adder

The third block multiplies the addition registers produced by the second block with the corresponding coefficient at each clock cycle. It also stores the multiplication results in separate registers using the code in Figure 30.

```
always@(posedge I_clk or posedge I_rst_p)
begin
    if(I_rst_p)
        begin
            mult_1 <= 'h0;
            mult_2 <= 'h0;
            mult_3 <= 'h0;
            mult_4 <= 'h0;
            mult_5 <= 'h0;

        end
    else
        begin
            mult_1 <= add_data_0 * coeff1;
            mult_2 <= add_data_1 * coeff2;
            mult_3 <= add_data_2 * coeff3;
            mult_4 <= add_data_3 * coeff4;
            mult_5 <= add_data_4 * coeff5;
        end
    end
end
```

Figure 30: FIR filter Block 3 Verilog code

The last block adds the results of the multiplication of the third block at each clock cycle using 24-bit adders. It also crops and preserves the sign of the final result to keep the output signal in its 12-bit format.

The adders in this block are selected according to the type of the system. In the accurate design, full adders are selected as illustrated in Figure 31.

```

FA_24bit Acc1 (mult_1, mult_2, tempacc_1);
FA_24bit Acc2 (mult_3, mult_4, tempacc_2);
FA_24bit Acc3 (second_level_sum_1, second_level_sum_2, tempacc_3);
FA_24bit Acc4 (tempacc_3, second_level_sum_3, data_out);

always@(posedge I_clk or posedge I_rst_p)
begin
    if(I_rst_p)
        begin
            second_level_sum_1 <= 'h0;
            second_level_sum_2 <= 'h0;
            second_level_sum_3 <= 'h0;
        end
    else
        begin
            second_level_sum_1 <= tempacc_1;
            second_level_sum_2 <= tempacc_2;
            second_level_sum_3 <= mult_5 ;
        end
    end
assign O_data = (data_out[23:0]>>>12);

endmodule

```

Figure 31: Verilog code for the full adders in Block 3

The first four lines of the last code specify the 24-bit full adders as target adders. These lines are changed in the approximate system. For instance, if the approximate system uses the CPredA adder, the code in Figure 32 is used.

```

CPredA_24bit Acc1 (mult_1, mult_2, tempacc_1);
CPredA_24bit Acc2 (mult_3, mult_4, tempacc_2);
CPredA_24bit Acc3 (second_level_sum_1, second_level_sum_2, tempacc_3);
CPredA_24bit Acc4 (tempacc_3, second_level_sum_3, data_out);

```

Figure 32: Verilog code when CPredA adders are selected

### 3.2.2 Implementation of Spike Detection Module

As illustrated in Figure 33, the spike detection module receives the 12-bit filtered sample produced by the FIR filter at each positive edge of the system clock. It uses the method of comparing the sample value with a negative threshold and detects the signal's

spike below that threshold. At that point, it pulls the spike output to a high level for a certain period to alert for spike detection and increment the count of the spikes. Estimating the correct threshold value is essential since a low threshold value leads to false detections while a high value leads to a high number of missed spikes. The threshold value is estimated in the accurate system to detect precisely all spikes and kept unchanged in all approximate system versions.

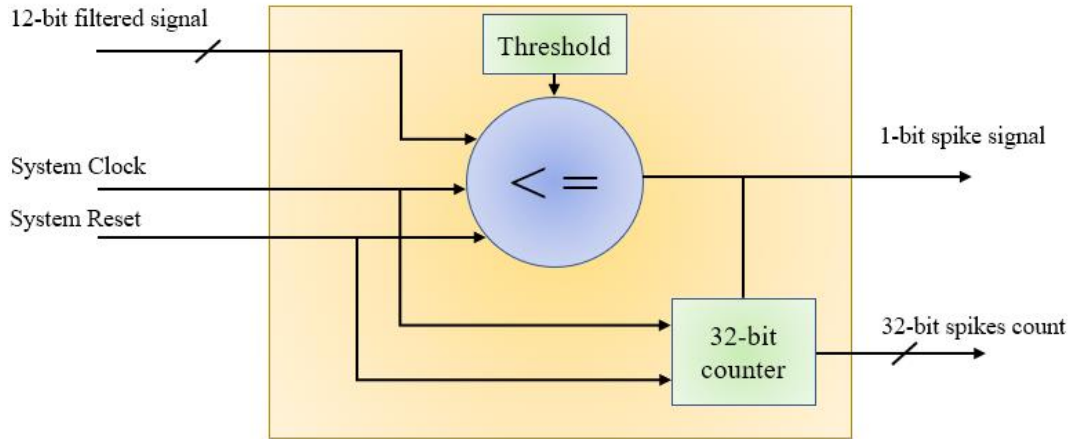


Figure 33: Spike detection module

The previous spike detection method is used due to its simple hardware implementation to prove the concept of this research. Other algorithms are also available in the literature. The algorithm proposed by Liu et al. (2018) detects the spikes in extracellular recordings by extracting the minimum distance between the trough and peak in a slice of recorded signal. The absolute distance value is then incorporated into the differential operator applied to the rectified signal to analyze the difference between spikes and noise. The signal is then passed through a convolution filter to suppress the noise. Spikes are finally detected when the sample value exceeds a preset threshold proportional to the mean value of the filtered signal. An adaptive spike detection algorithm is presented by Zhang & Constandinou (2021). The algorithm first removes the local field potentials from the recorded signal by mean subtraction and using a moving average filter. The signal-to-noise ratio is then enhanced by using an amplitude slope operator and finally spikes are detected by comparing the samples with an adaptive threshold considering the local signal statistics. Strollo (2021) proposed a low-power

spike detector using latch-based RAM. The threshold in this algorithm is estimated by dynamically calculating the standard deviation of noise in the new samples and updating the old value. Saggese et al. (2021) investigated a multiple spike detection algorithms for Multi-Transistors Arrays (MTA) based on some variants of the Smoothed Non-linear Energy Operator (SNEO). The latter work has shown that the performance of the spike detector benefits from the correlation of the signals detected by the MTA pixels but degrades when a high firing rate of neurons occurs.

The output of the implemented spike detection module is the spike pulse which can be high if a spike is detected, and the 32-bit spike counter that counts the number of spikes detected. This module is expressed in Verilog with the code in Figure 34.

```

module Spike_Detector(I_clk, I_rst_p, datain, spike_pulse, count);

input I_clk;
input I_rst_p;
input signed [11:0] datain;
output reg spike_pulse;
output reg [31:0] count;

reg signed [11:0] threshold = -12'd670;

always @ (posedge I_clk or posedge I_rst_p) begin
    if(I_rst_p) begin
        spike_pulse <= 1'b0;
        count <= 32'd0;
    end

    else begin
        if(datain <= threshold) begin
            spike_pulse <=1'b1;
        end
        else begin
            spike_pulse <=1'b0;
        end
    end
end

always @ (posedge spike_pulse) begin

    count <= count+1;

end

endmodule

```

Figure 34: Verilog code for the spike detection module

### 3.3 System Design Summary

Two processing systems are implemented: the accurate system, which uses 12 and 24-bit full adders, and the approximate system, which uses 12 and 24-bit approximate adders. Three versions of the approximate system are built, where each version uses one of the three approximate adders selected previously (CPredA, GeAr, and AA2). All system specifications, such as clock, I/O widths, and settings of the FIR filter and spike detection modules, are identical in all systems to maintain fair decisions.



Figure 35 shows the common schematic diagram of all processing systems implemented in this research.

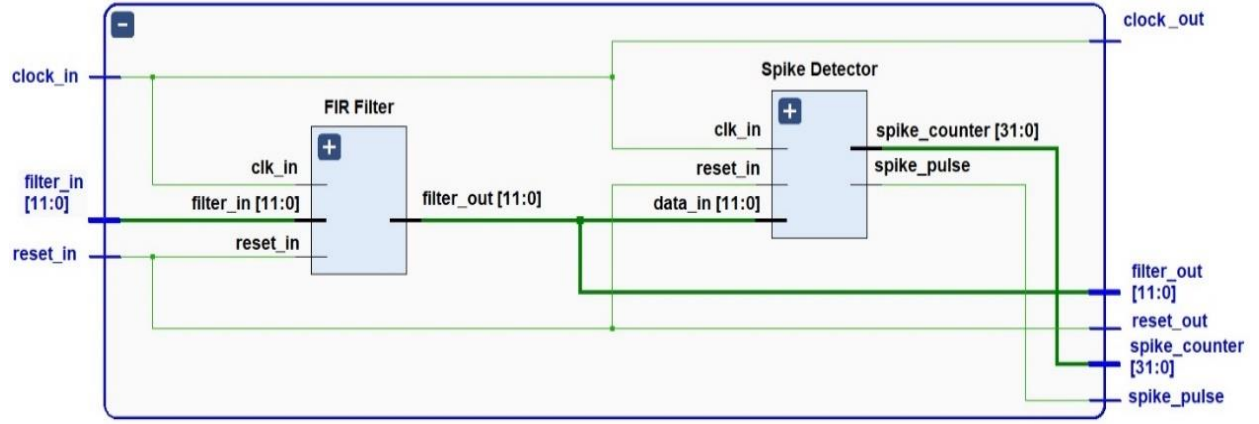


Figure 35: Common schematic diagram for All Systems

The outer wrapper module manages filtering and spike detection modules to synchronize inputs and outputs, clocking, and dataflow. This module is connected to the FPGA device's clock and reset signals. It receives the raw 12-bit neural sample generated by the MEA device at each clock cycle and forwards it to the FIR filter module for filtering. It also triggers the spike detection module to acquire the filtered sample from the FIR filter and compare it against the threshold value.

One filtered sample is produced at the output port of the wrapper module at each clock cycle, along with the spike information and count.

All modules are implemented using the Verilog hardware description language and Xilinx Vivado Design Suite with ZedBoard development board for the Xilinx Zynq - 7000 specified as the target board.

The FIR module receives three signals: the system clock signal, the system reset signal and the digitized 12-bit raw neural data. The FIR filter is a low-pass equiripple filter of order 8 with symmetric coefficients, which uses the last nine samples to produce the new filtered sample. The sample sliding window is shifted at each positive edge of the clock to include a new sample and discard the oldest sample in the window. Every two opposite samples are added using 12-bit adders, multiplied by the coefficient, and

then accumulated using 24-bit adders. The output of this module is a 12-bit filtered neural signal sent to the spike detection module.

The spike detector also receives three input signals, including the system clock signal used for synchronization, the system reset signal and the filtered neural data generated by the FIR filter. It compares the received sample at each positive edge of the clock with the negative threshold value and outputs a spike detection signal if the sample value is below the threshold. It also counts the number of spikes detected.

## Chapter 4: Results and Discussion

### 4.1 Evaluation of Approximate adders

The three selected approximate adders (CPredA, GeAr, and AA2) are used to build higher order adders with widths of 8, 12, 24, and 32 bits, then test them in two configurations: Half Prediction (HP) and Full Prediction (FP). In the Half Prediction configuration, the addition result is produced by predicting the result of the lower half bits using an approximate adder, while the upper half bits are added precisely using the full adder. For instance, to test an 8-bit adder in the Half Prediction configuration, the lower four bits of the operands are added using CPredA adders, while the upper four bits are added using full adders. In the Full Prediction configuration, the addition result is entirely produced using only approximate adders.

All adders are expressed using the Verilog language and implemented on the Xilinx Vivado Design Tool with ZedBoard development board for the Xilinx Zynq - 7000 specified as the target board.

The design delay time is not reported directly in the Vivado software. Therefore, we measured it by using a wrapper module and giving successively tighter timing constraints until the design fails the implementation step. The design delay time is determined by the last timing constraint that succeeded. Power and area are also obtained using the total on-chip power and chip utilization values in the implementation reports. Normalized Mean Error Distance (NMED) (Masadeh et al., 2018) is used to evaluate accuracy by testing the adders with various widths on 10,000 randomly generated samples. The Error Distance (ED) is defined as the difference between the accurate sum (S) and approximate sum (S'), as shown in Equation 4.1:

$$ED = |S - S'| \quad (4.1)$$

Mean Error Distance (MED) is then calculated as the average of ED as shown in Equation 4.2:

$$MED = ED / \text{Number of Samples} \quad (4.2)$$

NMED is finally calculated as the ratio between MED and the maximum exact result of tested addition operations, as shown in Equation 4.3:

$$\text{NMED} = \text{MED} / \text{Max. Result} \quad (4.3)$$

All characteristics are finally compared to the full adder as a reference. Table 5 and

Table 6 summarize the characteristics of the three approximate adders in Half Prediction and Full Prediction configurations. The tables show the delay time for each approximate adder in different widths along with the delay time of the full adder for comparison. They also show the reduction in time, area, and power normalized to the full adder.

Table 5: Testing Results in half prediction configuration

Module	bits	Total Delay (ns)	Reduction in time (%)	Reduction in area (%)	Reduction in power (%)	NMED
Full Adder	8	2.292				
	12	2.906				
	24	4.983				
	32	5.957				
CpredA	8	1.578	31.2%	18.2%	0.0%	0.0511307
	12	2.083	28.3%	33.3%	0.9%	0.0143707
	24	2.885	42.1%	14.6%	0.8%	0.0003206
	32	3.570	40.1%	17.2%	1.6%	0.0000077
AA2	8	2.110	7.9%	0.0%	0.9%	0.0281827
	12	2.650	8.8%	9.5%	1.8%	0.0071528
	24	4.503	9.6%	2.4%	2.5%	0.0001146
	32	5.330	10.5%	10.3%	3.9%	0.0000071
GeAr	8	1.551	32.3%	9.1%	0.0%	0.1076535
	12	1.777	38.9%	23.8%	0.9%	0.0295809
	24	2.817	43.5%	7.3%	0.8%	0.0004443
	32	3.270	45.1%	3.4%	0.0%	0.0000153

Table 6: Testing Results in full prediction configuration

Module	bits	Delay (ns)	Reduction in time (%)	Reduction in area (%)	Reduction in power (%)	NMED
<b>Full Adder</b>	8	2.292				
	12	2.906				
	24	4.983				
	32	5.957				
<b>CpredA</b>	8	1.519	33.7%	27.3%	0.0%	0.3141
	12	1.567	46.1%	42.9%	0.9%	0.3203
	24	1.581	68.3%	41.5%	1.6%	0.3155
	32	1.485	75.1%	44.8%	2.3%	0.3193
<b>AA2</b>	8	2.136	6.8%	9.1%	1.8%	0.3860
	12	2.640	9.2%	23.8%	2.7%	0.3844
	24	4.450	10.7%	22.0%	5.7%	0.3810
	32	5.250	11.9%	24.1%	3.9%	0.3823
<b>GeAr (R2-P2)</b>	8	1.626	29.1%	0.0%	0.0%	0.0904
	12	1.702	41.4%	19.0%	0.0%	0.0975
	24	1.843	63.0%	14.6%	0.8%	0.1070
	32	1.760	70.5%	19.0%	0.8%	0.0985

#### 4.1.1 Delay Time

Figure 36Error! Reference source not found.-37 show the delay time for all adders, including the full adder for reference.

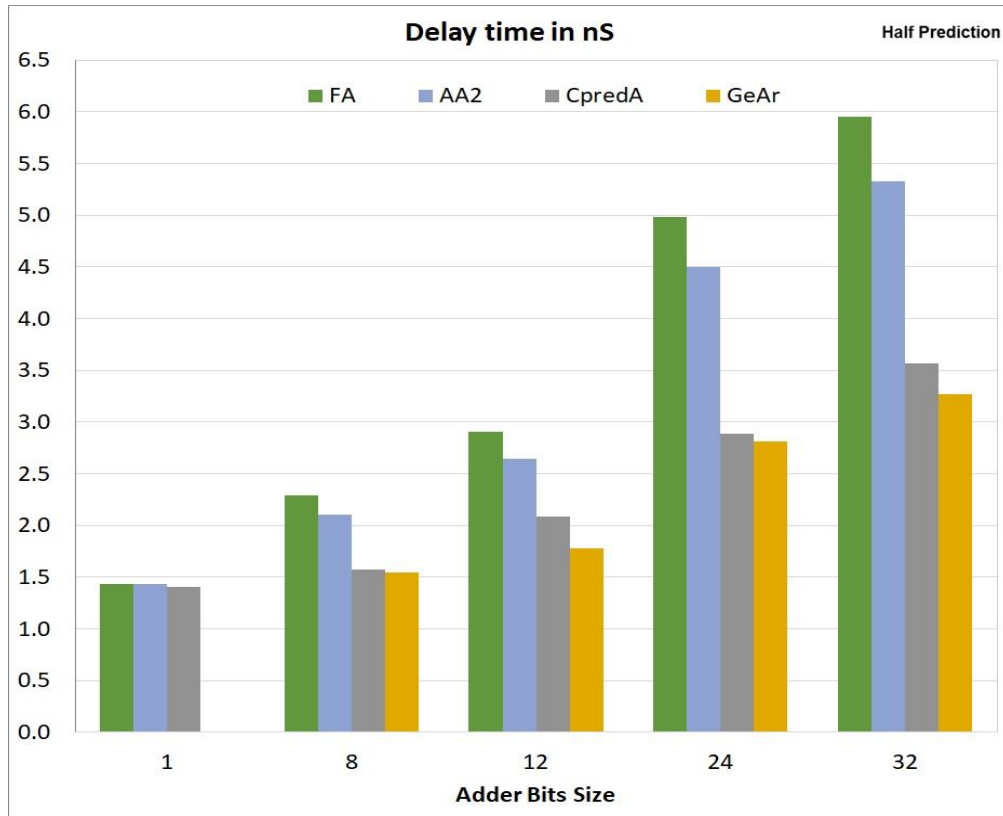


Figure 36: Adders delay time in half prediction configuration

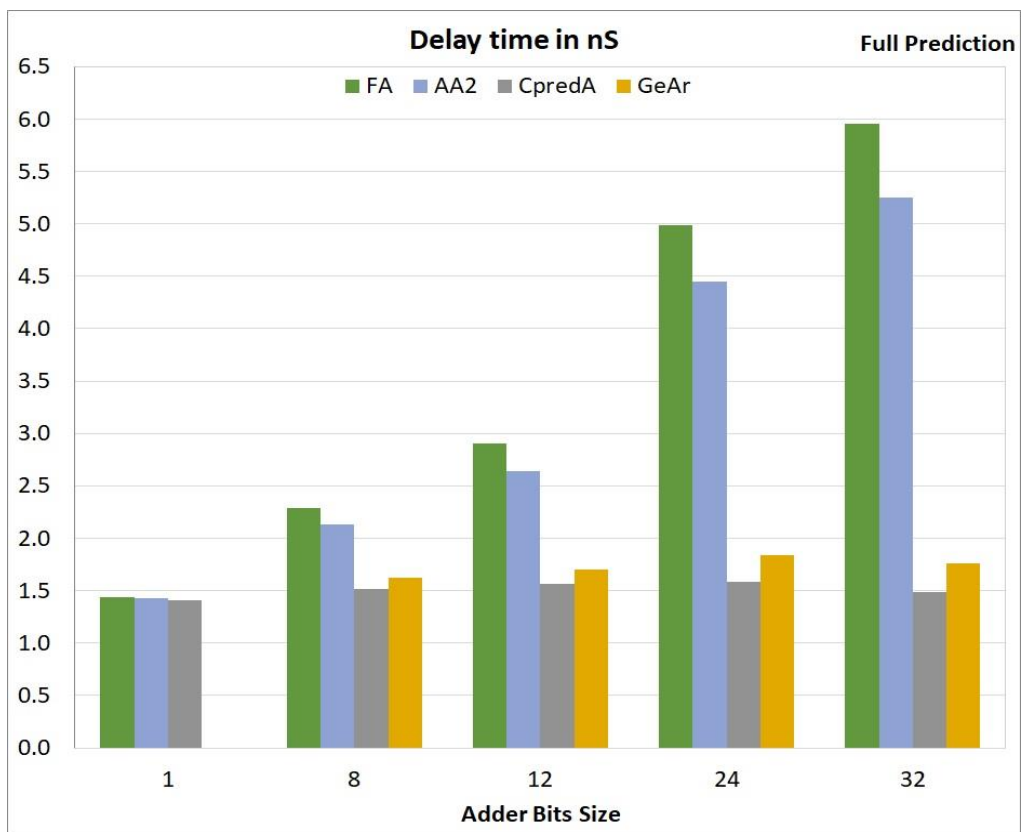


Figure 37: Adders delay time in full prediction configuration

In both configurations, CPredA and GeAr significantly reduce the delay time due to the parallelism in their architecture. CPredA predicts the carry using the values of A and B only. While GeAr breaks the carry chain into smaller chunks. The delay of AA2 is higher than CPredA and GeAr but still less than the full adder since this adder simplifies the production of outputs without breaking the carry chain.

#### 4.1.2 Area

Figure 38 **Error! Reference source not found.**-39 show the resource utilization, which measures the adder area by reporting the usage of chip resources.

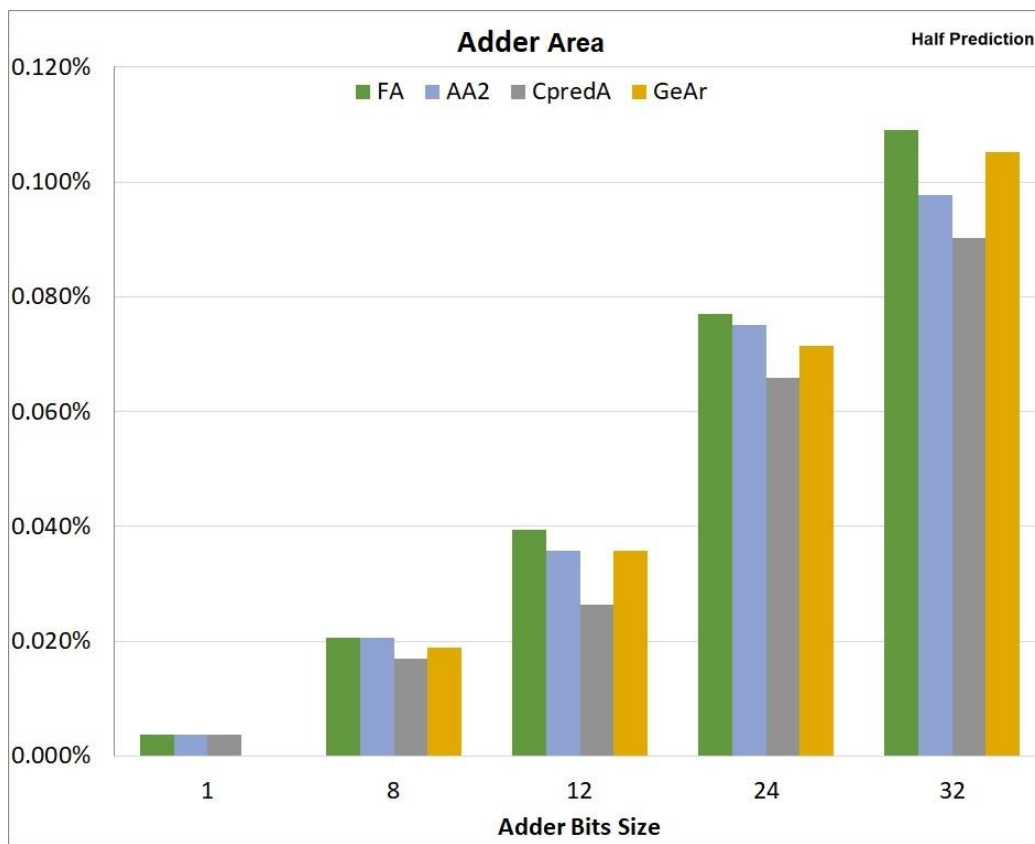


Figure 38: Adders area in half prediction configuration

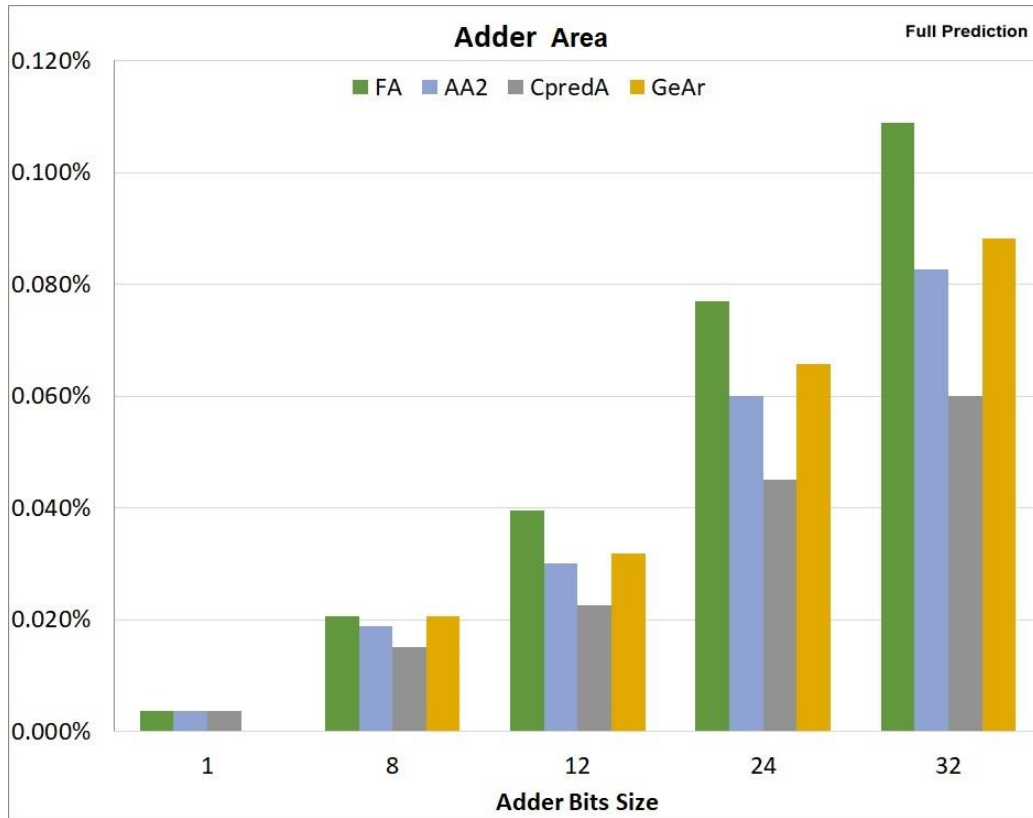


Figure 39: Adders area in full prediction configuration

GeAr is formed by dividing the original full adder into smaller segments. Therefore, the closest adder area to the full adder is the GeAr. CPredA and AA2 have smaller areas because their logical circuits are more straightforward than the regular full adder circuit.

#### 4.1.3 Power Estimation

Figure 40-41 show the power estimation for all approximate and full adders.



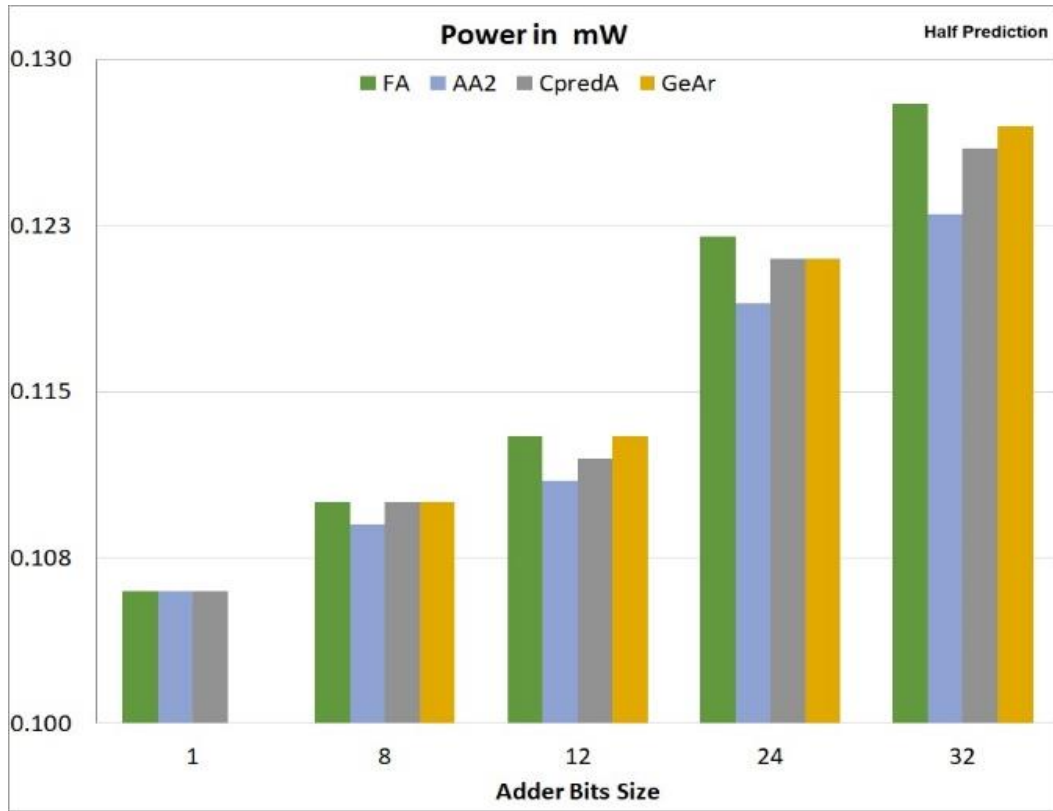


Figure 40: Power estimation of adders in half prediction configuration

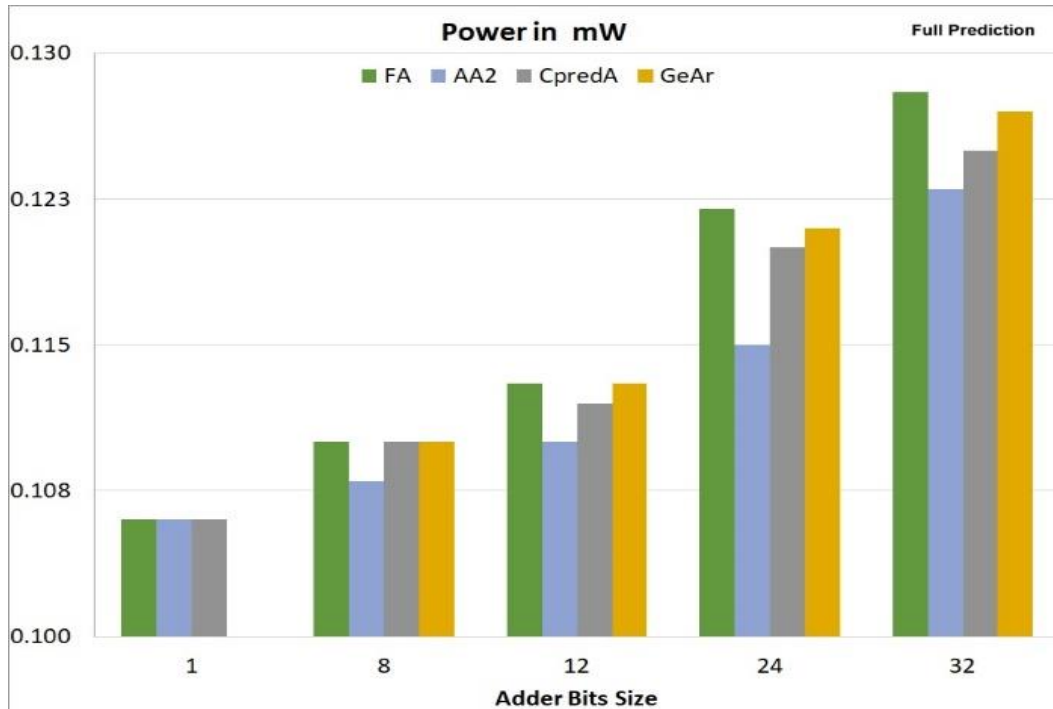


Figure 41: Power estimation of adders in full prediction configuration

All adders consume the same amount or slightly less power than the full adder. As expected, GeAr is showing the closest values to the full adder since it is multiple segments of it. AA2 shows less power consumption due to its simplified circuit where the sum is an inverted version of the output carry.

#### 4.1.4 Accuracy

Figure 42-43 show the normalized mean error distance, which measures the adder accuracy. The vertical axis in the graphs uses a logarithmic scale and shows  $-10/\log$  (NMED) to enhance the readability of the small values. The original values are shown in Table 5 and Table 6 and

Table 6.

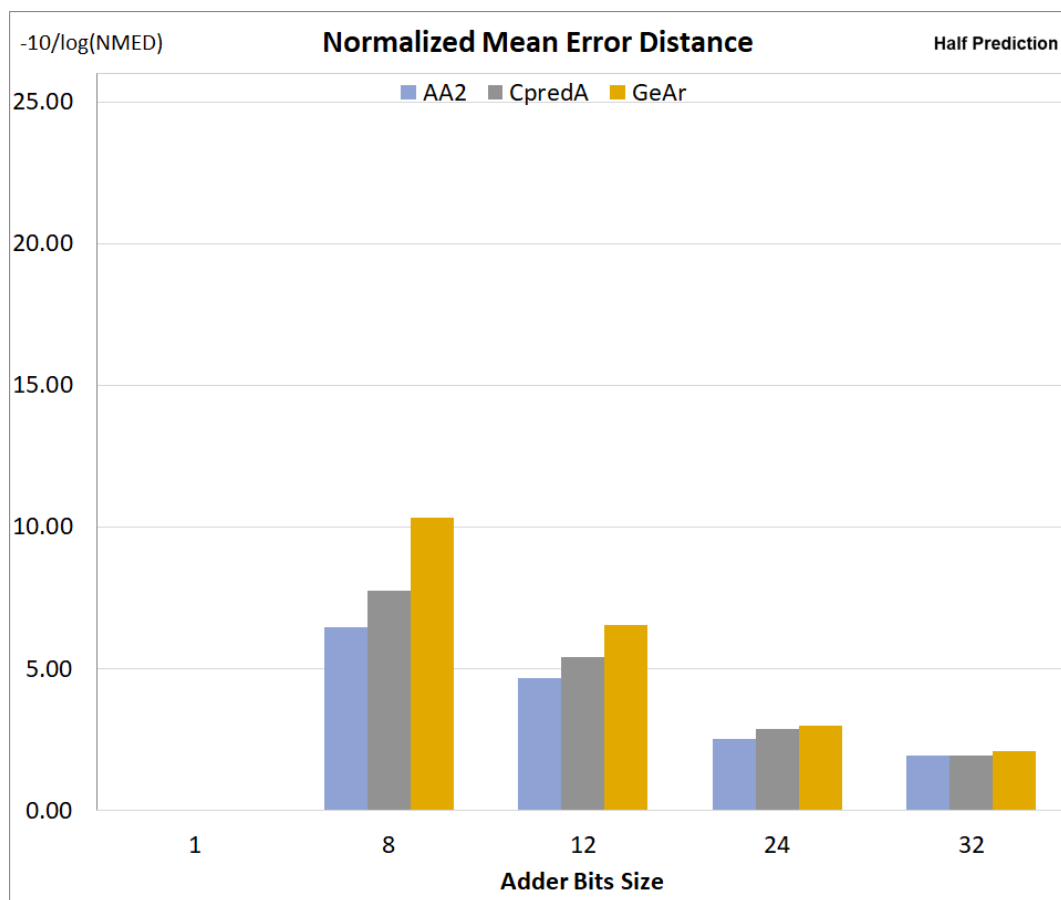


Figure 42: NMED of adders in half prediction configuration

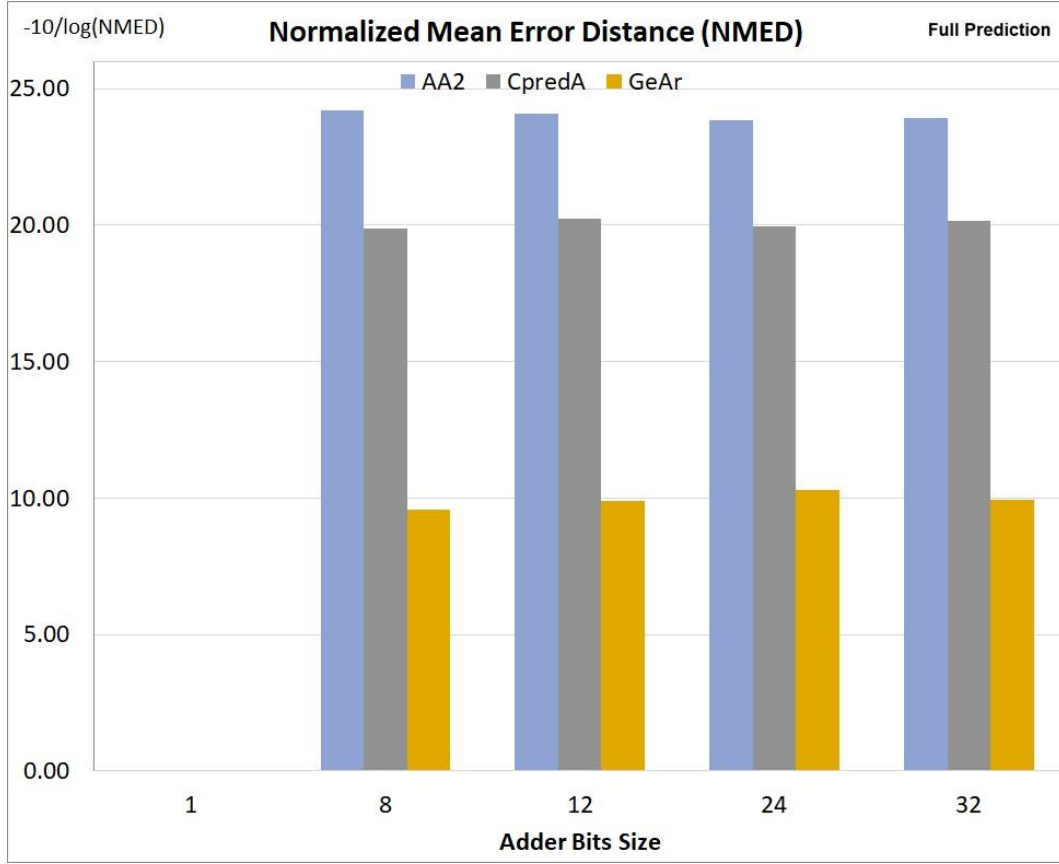


Figure 43: NMED of adders in full prediction configuration

In the Half Prediction configuration, the lower half of the final result is approximated while the upper half is accurate. In this configuration, all adders show relatively small NMED, which decreases as the operand size increases. When the operand size increases, the error value becomes only a small portion of the accurate result. In Full prediction configuration, NMED is constant for all adder widths, with GeAr having the most negligible value. GeAr shows the lowest error measure among other adders since each sub-adder result is calculated accurately using some last bits, 4-bits in the case of GeAr R2 P2.

## 4.2 Evaluation of Filtering Errors

The previous step evaluated the approximate adders to explore their potential in serving the research purpose of enhancing the MEA signals processing system. An essential effort before building the system is to check if using these adders in the first

processing stage, which is the filtering process, will produce an acceptable filtered signal that can then be forwarded to the spike detection module. It is also essential to predict the implication of using approximate adders on the accuracy of the whole processing system.

Figure 44 illustrates that the FIR filter in the proposed processing system uses two adder widths, 12 and 24-bit. Every two opposite samples of the signal in the FIR are added together using the 12-bit adders. The results of the previous additions are multiplied with corresponding coefficients and then added using the 24-bit adders.

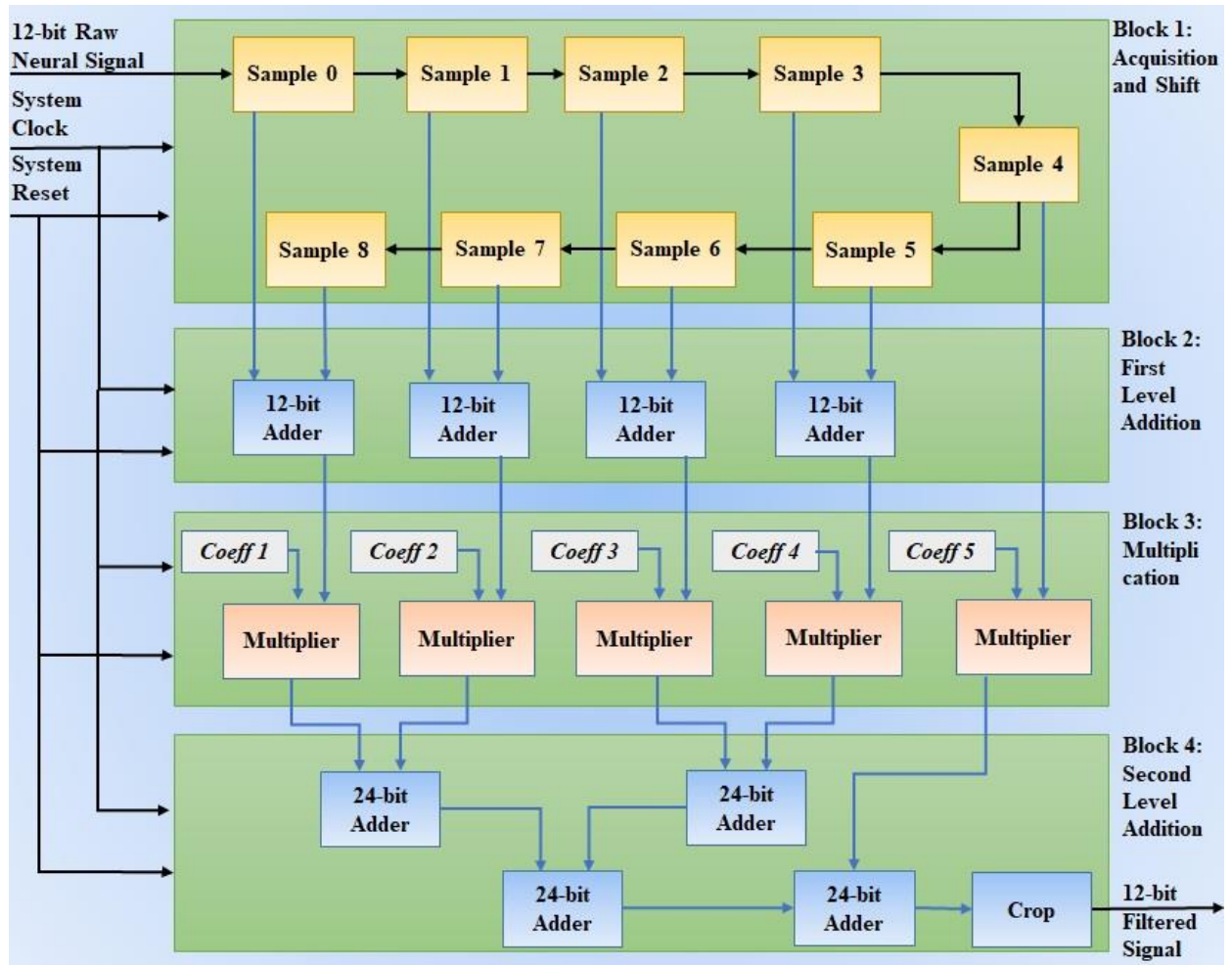


Figure 44: FIR Filter in Proposed Processing System

The FIR filter is tested in this step with the approximate adders CPredA, AA2, and GeAr at different approximation levels, as shown in Table 7. The tests are conducted on a real neural signal downloaded from 3Brain.com (3Brain, 2023), containing 35,276 samples. The filtered output signal is then compared with the filtered signal obtained

using FIR with accurate adders. NMED is used to evaluate the error value in the FIR output signal with each type of adder at different approximation levels.

Table 7: Approximation Levels and Adders' Configurations Used in Filtering Tests

	<b>Adder Configuration</b>	
<b>Level</b>	<b>12-bit Adder</b>	<b>24-bit Adder</b>
1	Half prediction	No prediction (accurate adder)
2	Full prediction	No prediction (accurate adder)
3	No prediction (accurate adder)	Half prediction
4	No prediction (accurate adder)	Full prediction
5	Half prediction	Half prediction
6	Full prediction	Half prediction
7	Half prediction	Full prediction
8	Full prediction	Full prediction

We began at level 1 by approximating the 12-bit adders used in the second FIR block to half prediction configuration while using accurate 24-bit adders in the fourth FIR block. The level of approximation was then increased until we reached full prediction in both 12 and 24-bit adders.

When the 12-bit adders are in half prediction configuration, they predict the six lower bits of the sum while they predict twelve bits of the sum in full prediction configuration. Similarly, the 24-bit adders predict twelve lower bits of the sum in half prediction while fully predicting the twenty-four bits of the sum in full prediction configuration. Figure 45 shows a sample of the filtering process output at level 1.

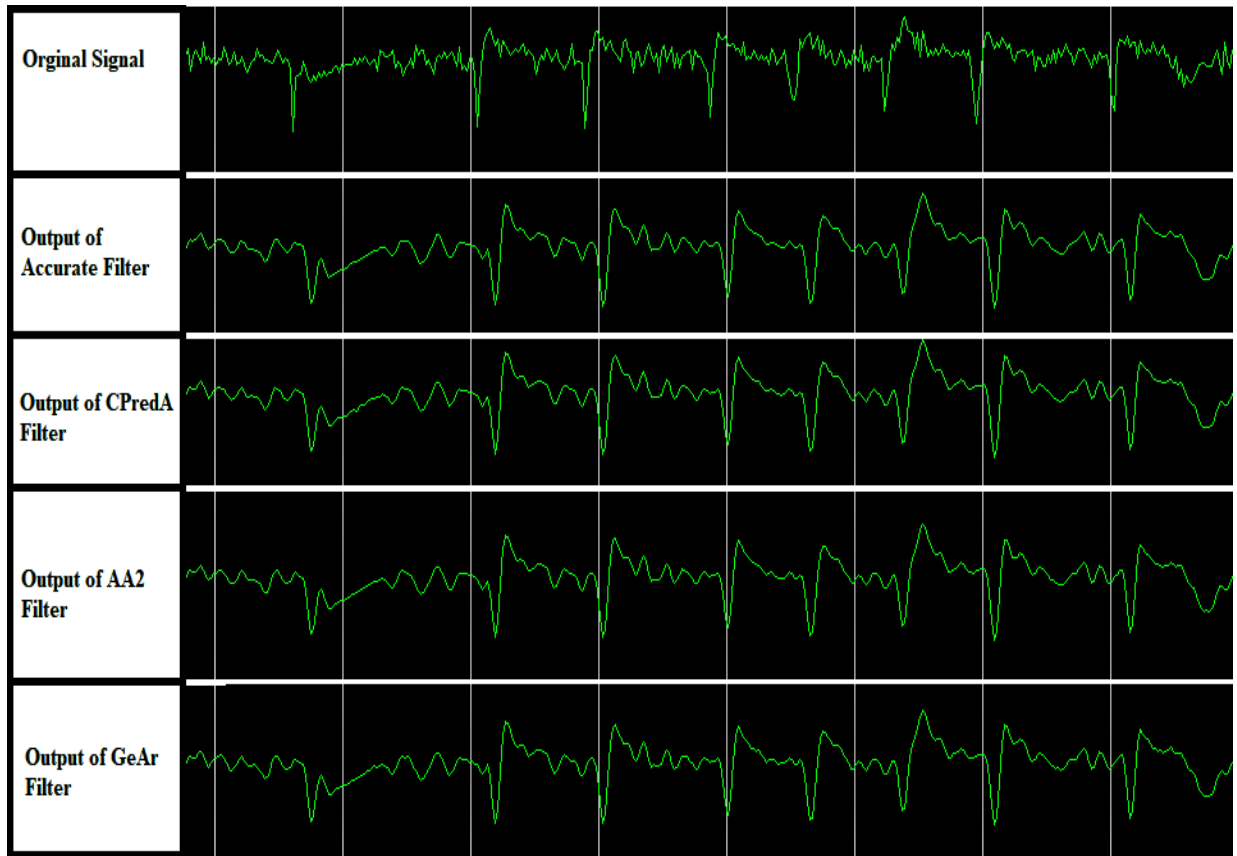


Figure 45: Sample of the filtering process output at approximation level 1

The input neural signal is applied to the input of the filter. Then the output signal is obtained and compared to the signal obtained using accurate adders. NMED for each adder-based FIR is then calculated at each level of approximation.

Table 8 presents the results of these tests. It shows the approximation level and adder configuration for the 12-bit adders used in block 2 and the 24-bit adders used in block 4 of the FIR filter. NMED is also presented with three adder types (CPredA, AA2, and GeAr).

Table 8: Approximation Levels and Filters' NMED

Level	Adder Configuration		NMED of the Filter for Each Adder Used		
	12-bit Adder	24-bit Adder	CPredA	AA2	GeAr
1	Half prediction	No prediction (accurate adder)	0.0095232	0.00783	0.018379
2	Full prediction	No prediction (accurate adder)	0.0698913	0.10507	0.101823
3	No prediction (accurate adder)	Half prediction	0.025617	0.01108	0.052142
4	No prediction (accurate adder)	Full prediction	0.0893383	0.12505	0.12394
5	Half prediction	Half prediction	0.0348504	0.01362	0.070972
6	Full prediction	Half prediction	0.078738	0.11564	0.106501
7	Half prediction	Full prediction	0.0921594	0.12907	0.120065
8	Full prediction	Full prediction	0.1270446	0.14362	0.128553

Figure 46-48 illustrate NMED values in the filter output for every adder at each approximation level.

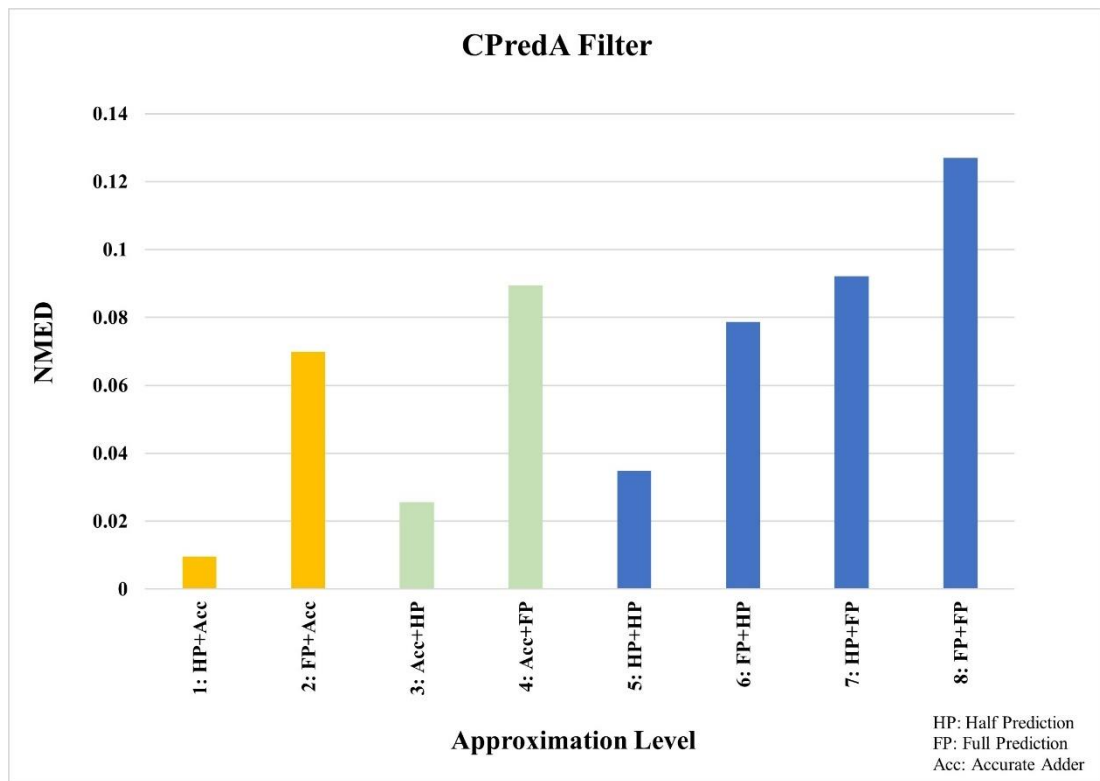


Figure 46: NMED for the FIR filter using CPredA adder

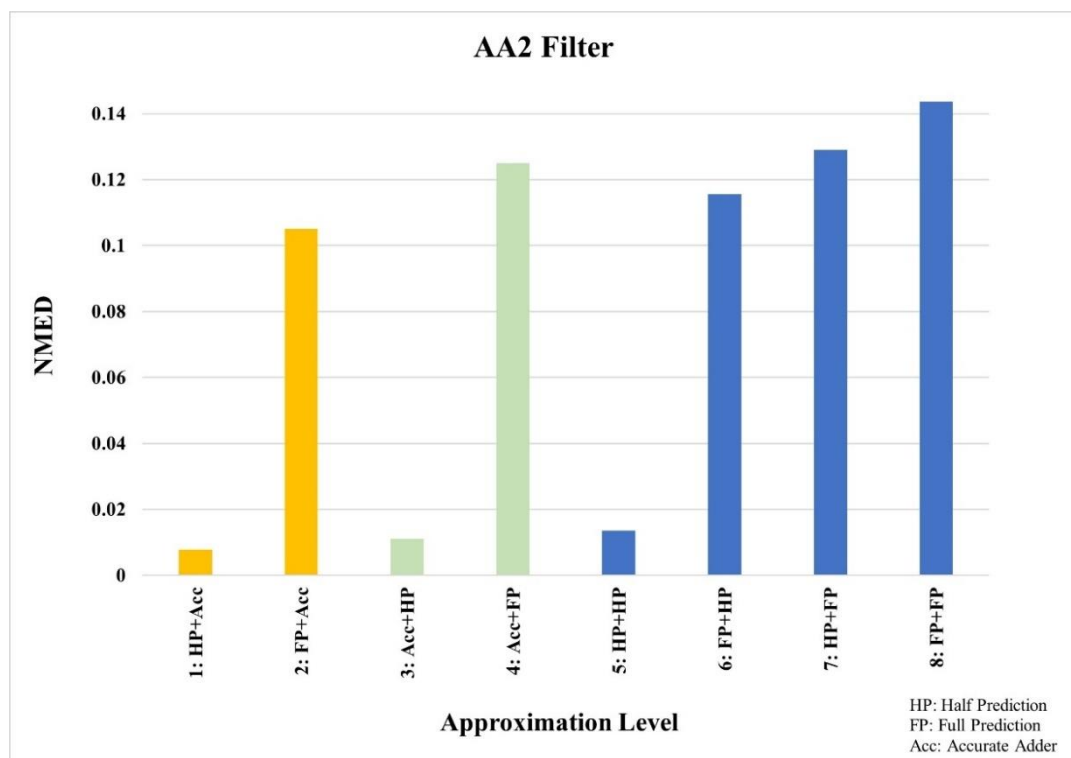


Figure 47: NMED for the FIR filter using AA2 adder



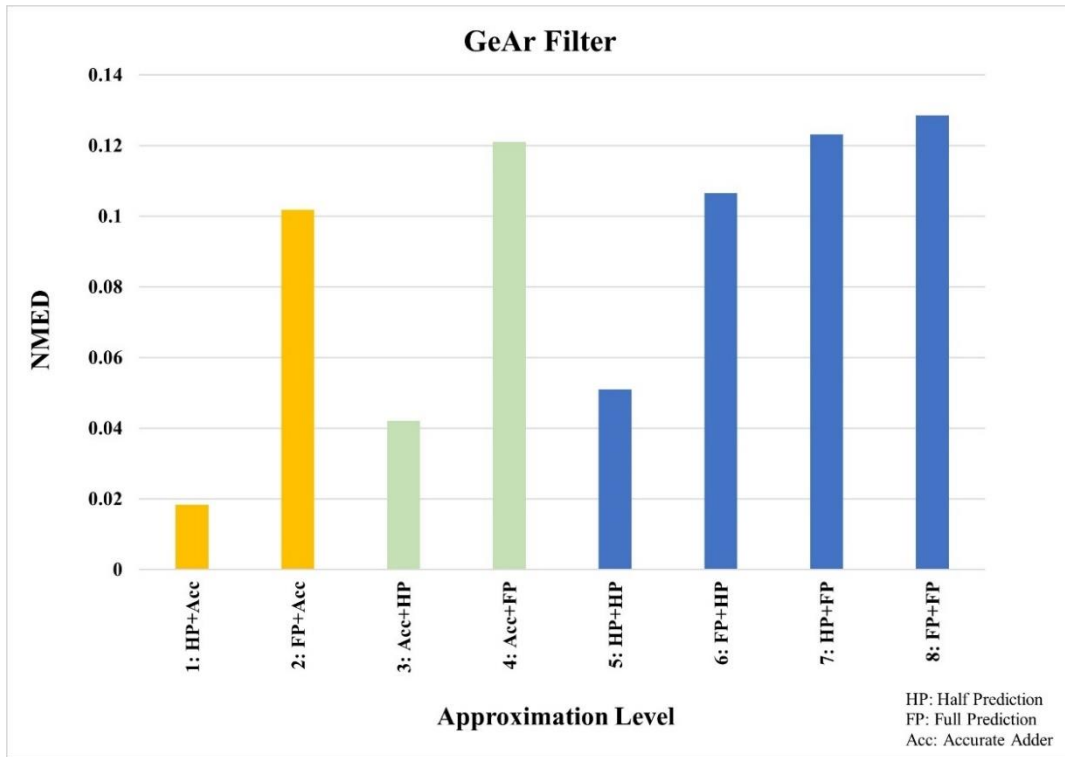


Figure 48: NMED for the FIR filter using GeAr adder

The previous three figures show that when we increase the approximation level of the 12 or 24-bit adders, the accumulated error in the output signal of the filter rises.

All filter versions started with very low error rates at levels 1 and 2. Therefore, we expect high detection of spikes by the processing system when we set its filter at this level 1. In contrast, the error becomes more significant when the 12-bit adders are all configured to predict their results entirely at level 2. Hence, lower detection of spikes is expected at this level.

At level 3, accurate 12-bit adders are used with 24-bit adders set to half prediction configuration. Then, the approximation level of the 24-bit adders is increased full prediction at level 4. The error in the filter output is still low at level 3, but more than at level 1 since the 24-bit adders are predicting twelve bits of their results instead of six in the 12-bit adders. The accuracy of the processing system is expected to be high at level 3 and worse at level 4 than at level 2.

At levels 5, 6, 7, and 8, the half prediction and full prediction configurations are combined to investigate each combination's effect on the filter output. The filtering error

started low at level 5 when both 12 and 24-bit adders were set to the half prediction configuration. Hence, high accuracy is expected for the whole processing system. The error increased significantly at levels 6, 7, and 8 as the approximation level increased. In other words, the number became greater than any previous level, indicating that the accuracy of the entire system will be very low at these three levels.

### **4.3 Evaluation of the Approximate Processing System**

This section presents the results of applying approximate computing to neural signal processing.

The proposed processing system consists of identical parallel sets of filtering and spike detection modules. Each set filters the signals acquired from a single MEA channel with its FIR filter that uses 12 and 24-bit approximate adders in its calculations. The filtered sample is forwarded to the spike detector, which compares it with a threshold value. A spike is detected when the sample value is below the threshold. Spike output is pulled high to indicate the presence of a spike, and the spike counter is incremented.

This research proposes to use the approximate computing paradigm to minimize the time required by each set to process the signal received from its channel. Similarly, this research aims to reduce or maintain the power consumption of the processing set and minimize the circuit area expressed by the utilization of FPGA resources. Reducing the area of each set will allow the system to handle more MEA channels with more programmed sets. On the other hand, a faster processing set will enhance the whole system's performance since all processing sets run in parallel.

Two systems, accurate and approximate, are built identically with the Verilog language and implemented on the Xilinx Vivado Design Tool with ZedBoard for the Xilinx Zynq-7000 specified as the target board. Accurate full adders are used in the accurate system, while approximate adders are used in the three approximate system versions. Each version is built based on one of the approximate adders CPredA, GeAr, and AA2 then tested in different approximation modes.

Different approximation levels are available to study the proposed system performance in terms of accuracy, latency, area, and power. As shown in section 4.2,

some approximation levels produce insignificant errors while others produce higher rates. As a result, system accuracy tests are initially conducted in order to select the most valuable levels and then use them for further research.

The accuracy of detecting spikes in all approximate system versions is measured at the same filtering and threshold setting as the accurate system. Other parameters such as delay time, area, and power are measured after the implementation step of each proposed system version, which includes placing and routing, then compared with the accurate system parameters.

All tests are done on real neural data signals downloaded from the 3Brain website (3Brain, 2023). Data signals used involve around 106,000 samples fed into accurate and approximate systems. The number of samples is sufficient to overcome any abnormalities in some samples and obtain consistent results.

#### *4.3.1 Testing System Accuracy*

As a first step, the data samples are processed offline using the official BrainWave X software from 3Brain to determine the number of spikes measured and then used as a reference in testing the implemented accurate and approximate systems. The implemented accurate system is then set to detect the number of spikes detected offline. All system settings such as the threshold value and filter specifications are kept unchanged while testing the proposed approximate system in all its versions and approximation levels.

All approximation levels are applied to select the useful ones which can be used to study the proposed system performance in the next stage. The first accuracy test of the approximate systems is conducted when the 12-bit adders are in the half prediction configuration, and 24-bit accurate adders are used. The approximation level is then increased, and the accuracy of the system in detecting spikes is measured, as presented in Table 9.

Table 9: System Accuracy Results at Different Approximation Levels

Level	Adder Configuration		System Accuracy		
	12-bit Adder	24-bit Adder	CPredA	AA2	GeAr
1	Half prediction	No prediction (accurate adder)	100.00%	100.00%	100.00%
2	Full prediction	No prediction (accurate adder)	83.00%	25.32%	57.03%
3	No prediction (accurate adder)	Half prediction	100.00%	100.00%	100.00%
4	No prediction (accurate adder)	Full prediction	0.00%	0.00%	0.00%
5	Half prediction	Half prediction	100.00%	100.00%	100.00%
6	Full prediction	Half prediction	69.30%	23.61%	54.00%
7	Half prediction	Full prediction	0.00%	0.00%	0.00%
8	Full prediction	Full prediction	0.00%	0.00%	0.00%

Test results demonstrate that the proposed system accuracy is very significant at levels 1, 3, and 5. They show that the maximum system accuracy is achieved if the approximation level does not exceed the half prediction in the 12 and 24-bit adders. These three levels produce the lowest error in the filtering process, as shown in Table 8 in the previous section.

Levels 2 and 6 show that the system accuracy drops to lower values when the approximation level of the 12-bit adders is set to full prediction while keeping the 24-bit adders approximation level at half prediction and below. As a result, at these approximation levels, the filter output signals are more prone to errors, causing the spikes to be undetected or incorrectly detected by the system.

Levels 4, 7, and 8 produce the worst accuracy results when applied to the system. Fully approximating the 24-bit adders of the system, which work as filters' accumulators, will result in a highly disturbed signal even if the 12-bit adders are not approximated. The filters' output signals have the highest error rates at these levels, as shown in Table 8 before. These three levels are excluded from future tests.

It is also noticeable from Table 8 that the accuracy decreases when the approximation level of the 12-bit adders is changed from half prediction to full prediction while keeping the 24-bit adders in half prediction configuration. Thus, two more levels between levels 5 and 6 are added to study further the behavior and performance of the system in this area. Eight bits of the 12-bit adders are predicted in the first extra level, and ten bits are predicted in the second extra level.

#### 4.3.2 Testing System Performance

Useful levels, such as 1, 2, 3, 5, 6, and the two extra levels, are used to study approximate system versions in seven different modes, as shown in **Error! Reference source not found.**

Table 10: Approximate System Modes

Mode	Adders Configurations	
	12-bit Adder	24-bit Adder
1	Half prediction	No prediction (accurate adder)
2	Full prediction	No prediction (accurate adder)
3	No prediction (accurate adder)	Half prediction
4	Half prediction	Half prediction
5	8 bits predicted	Half prediction
6	10 bits predicted	Half prediction
7	Full prediction	Half prediction

The neural signals are applied to each system version in these modes. Performance parameters, such as accuracy, delay, area, and power, are measured after synthesizing and then implementing the system on Xilinx Vivado software, including placing and routing steps.

The design delay time is not reported directly in Vivado tool. Hence, we measured it by using a wrapper module and giving successively tighter timing constraints until the design fails the implementation step. The last timing constraint that succeeded is picked as the design delay time. The system area is obtained from the chip utilization report after the implementation step which includes placing and routing. All parameters are then compared with the accurate system. The following sections present the performance measurements and analysis of the proposed approximate system in all approximation modes.

4.3.2.1 Mode 1

In this mode, three approximate systems are developed using half prediction 12-bit CPredA, AA2, or GeAr adders while the 24-bit adders remain accurate.

Figure 49-51 show the test results of accuracy, delay time, and area for the three approximate system versions and accurate system results for comparison.

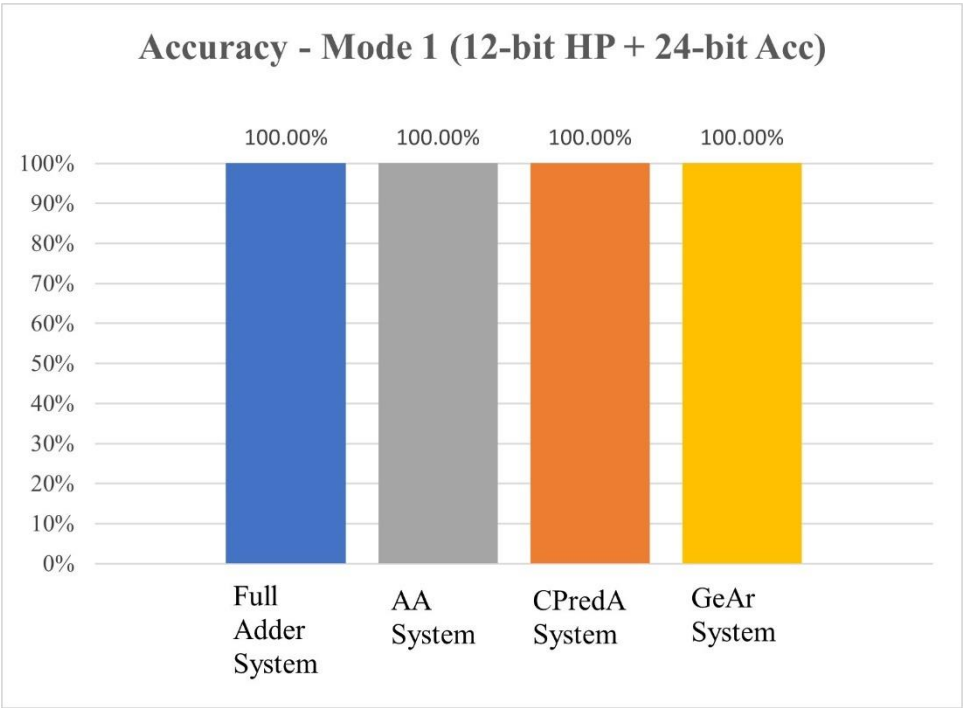


Figure 49: Accuracy test results for processing systems in Mode 1

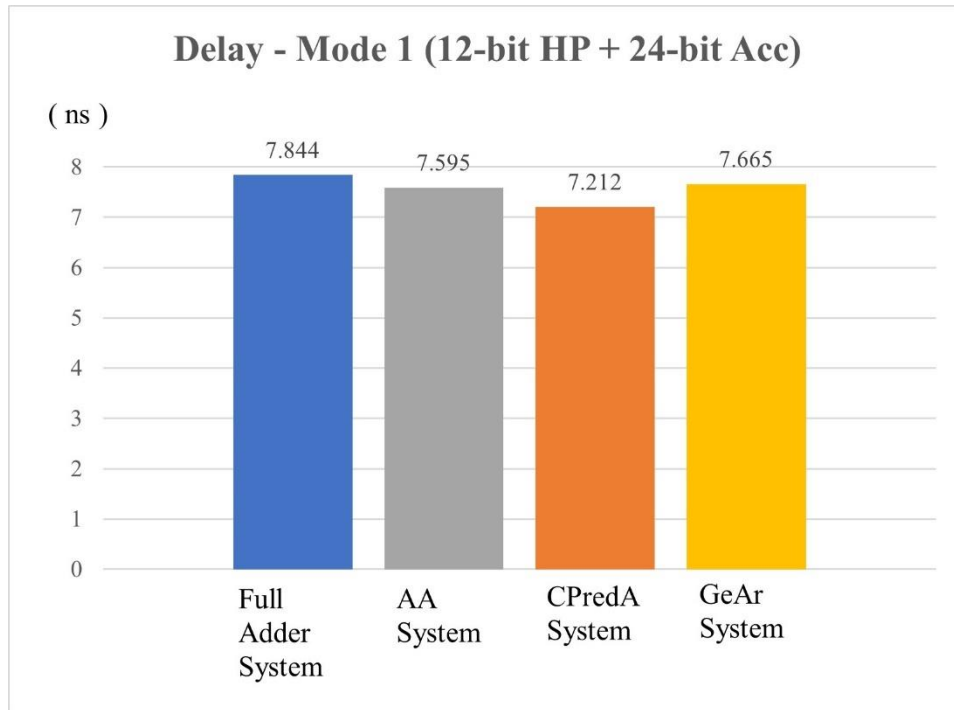


Figure 50: Delay test results of processing systems in Mode 1

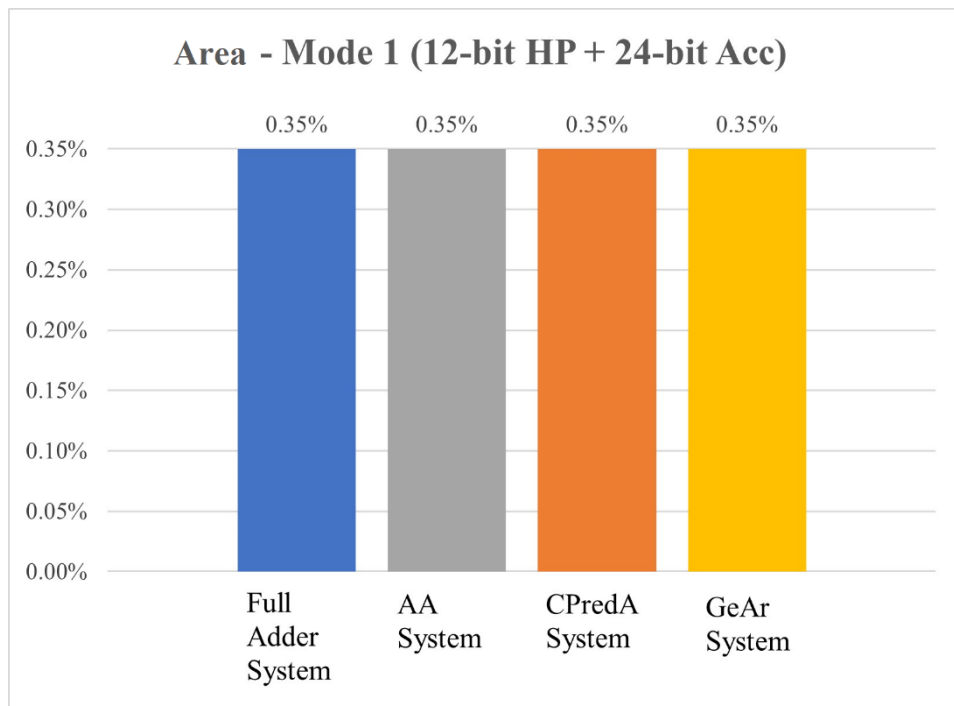


Figure 51: Area test results of processing systems in Mode 1

The results of this mode show that some enhancement in system speed can be achieved at a low level of approximation without losing the system's accuracy. The error

produced in the filter output signal at this approximation level is insignificant as discussed in section 4.2 which does not affect the spike detection accuracy of the system.

The highest reduction achieved in the system delay is 8% at this level when using CPredA adder, with no significant reduction in the system area. Six bits of the 12-bit adder result are produced in parallel in CPredA system. The CPredA circuit architecture produces the sum and the carry outputs in parallel without creating a carry chain, reducing the time required for calculations. The AA adder circuit does not break the carry chain, instead it calculates the output carry depending on the input operands bits and the previous carry. It then inverts the output carry bit to produce the sum bit value. This architecture forces the AA adder to produce its sum and carry bits sequentially which causes more delay. GeAr in half prediction mode divides the 12-bit result into three segments (6 + 2 + 4) bits which are produced in parallel using ordinary full adders. The results of the three segments are then combined to form the full 12-bit result. This brings the delay of the GeAr closer to the full adder system.

#### *4.3.2.2 Mode 2*

The approximation level in the 12-bit adders is increased in this mode All three versions of the approximate systems developed by fully predicting the outputs of the 12-bit adders and precisely calculating the 24-bit adders' outputs.

Figure 52-54 show the test results of this mode.



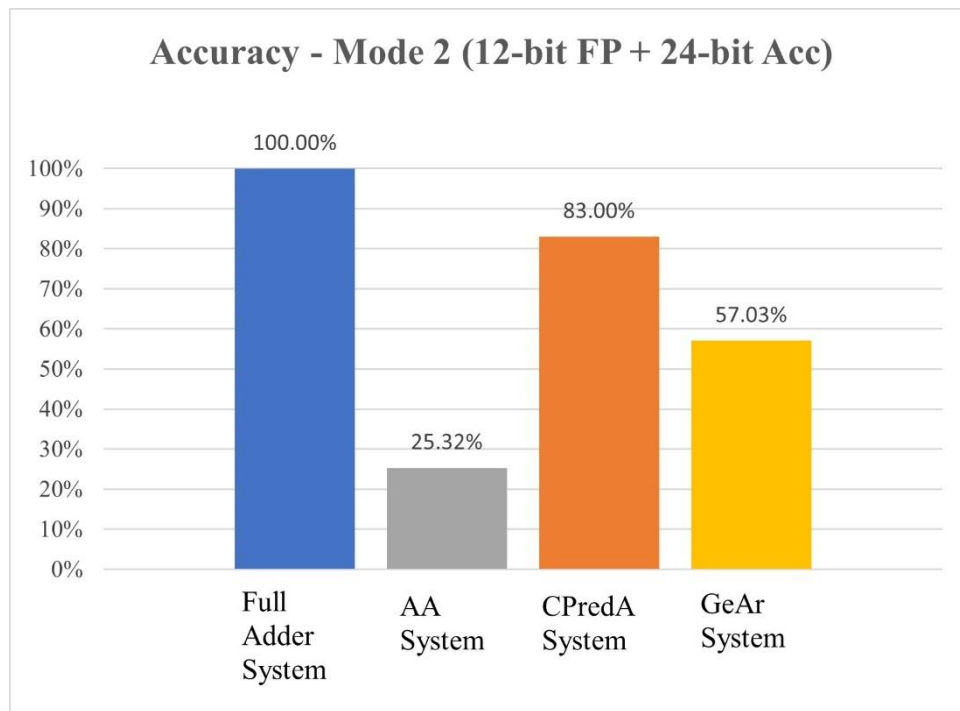


Figure 52: Accuracy test results for processing systems in Mode 2

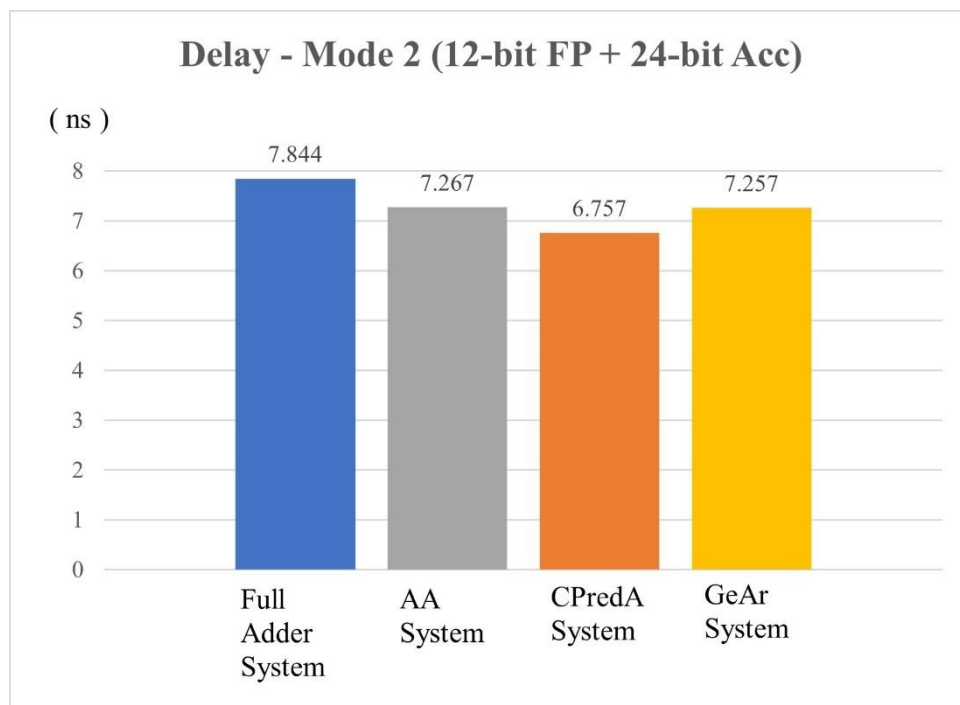


Figure 53: Delay test results of processing systems in Mode 2

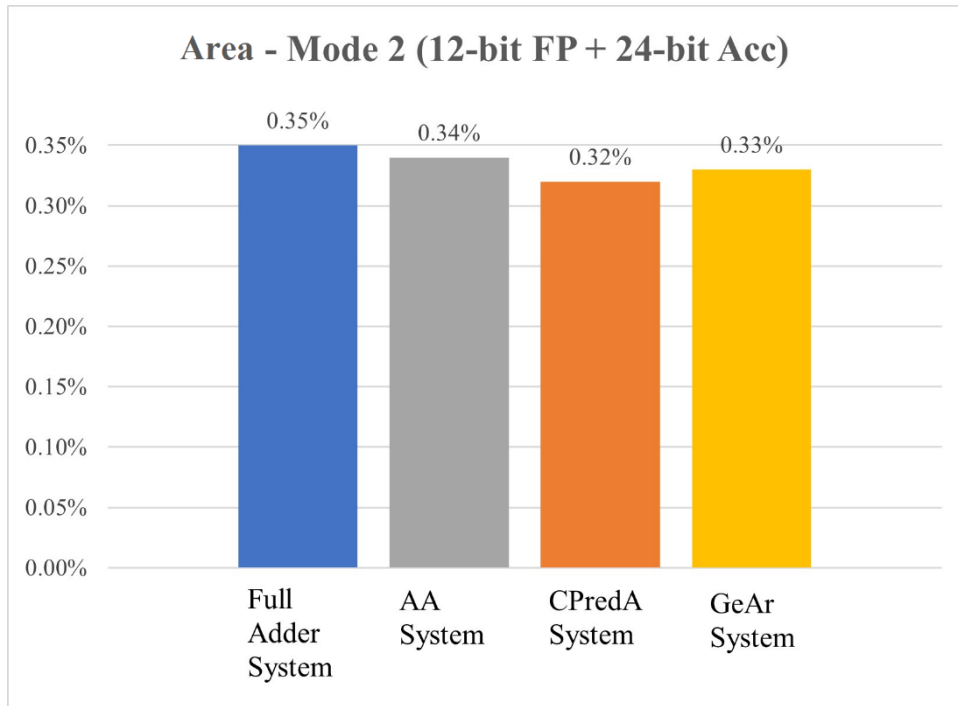


Figure 54: Area test results of processing systems in Mode 2

Accuracy drops in this mode due to the high approximation level of the 12-bit adders. The results of these adders are fully predicted, producing significant error rates even if the 24-bit adders are accurate. The approximate system can achieve the highest accuracy in this mode by using the CPredA circuit which predicts the output carry only within its architecture. The CPredA system in this mode shows the highest accuracy among others since the errors produced in the filter output when using CPredA are the minimum compared to AA2 and GeAr adders, as shown in section 4.2. GeAr system accuracy is also better than AA system since its addition result is a combined version of the accurate one through segmentation.

Generally, the delay and area enhancements in this mode are low in comparison with the accuracy losses.

#### 4.3.2.3 Mode 3

No approximation is applied to the 12-bit adders in this mode. In contrast, the 24-bit adders are set to predict half of their outputs.

Figure 55-57 show the accuracy, delay, and area tests' results in this mode.

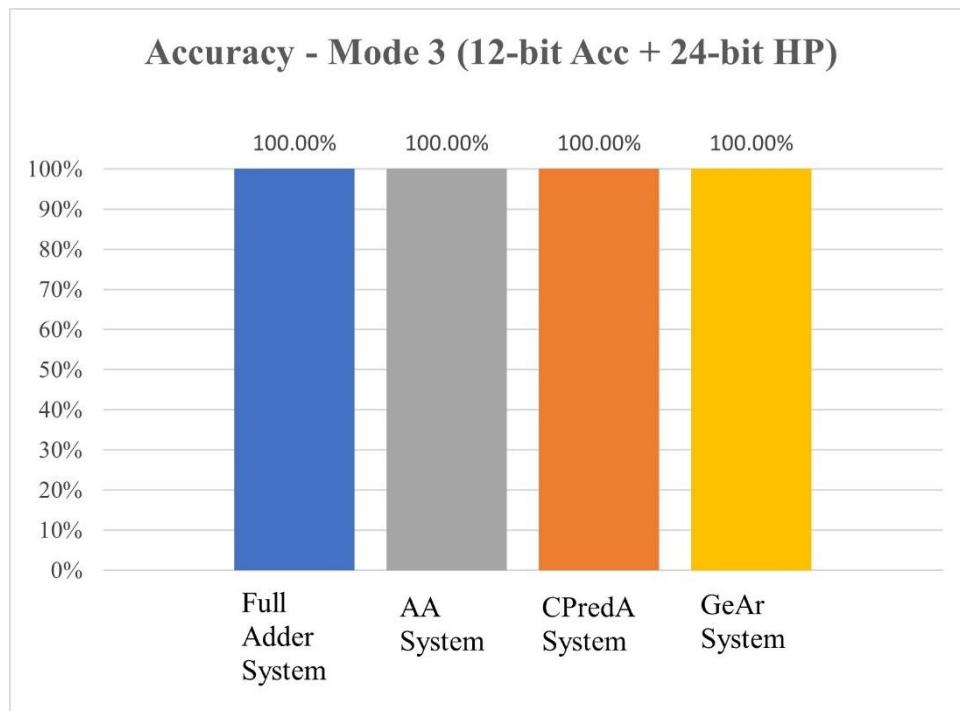


Figure 55: Accuracy test results for processing systems in Mode 3

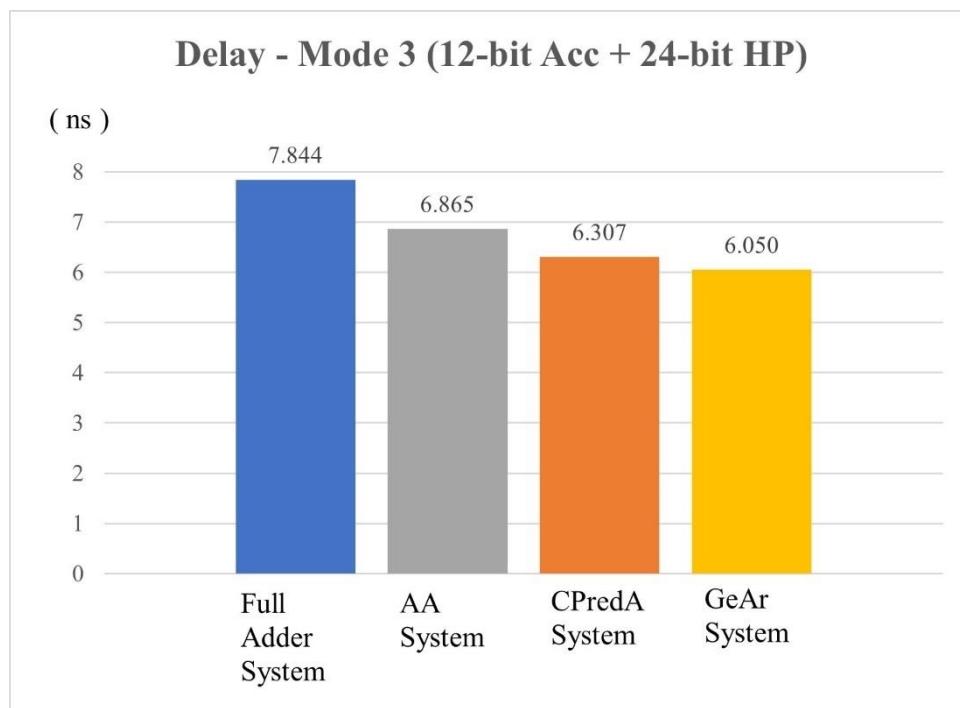


Figure 56: Delay test results of processing systems in Mode 3

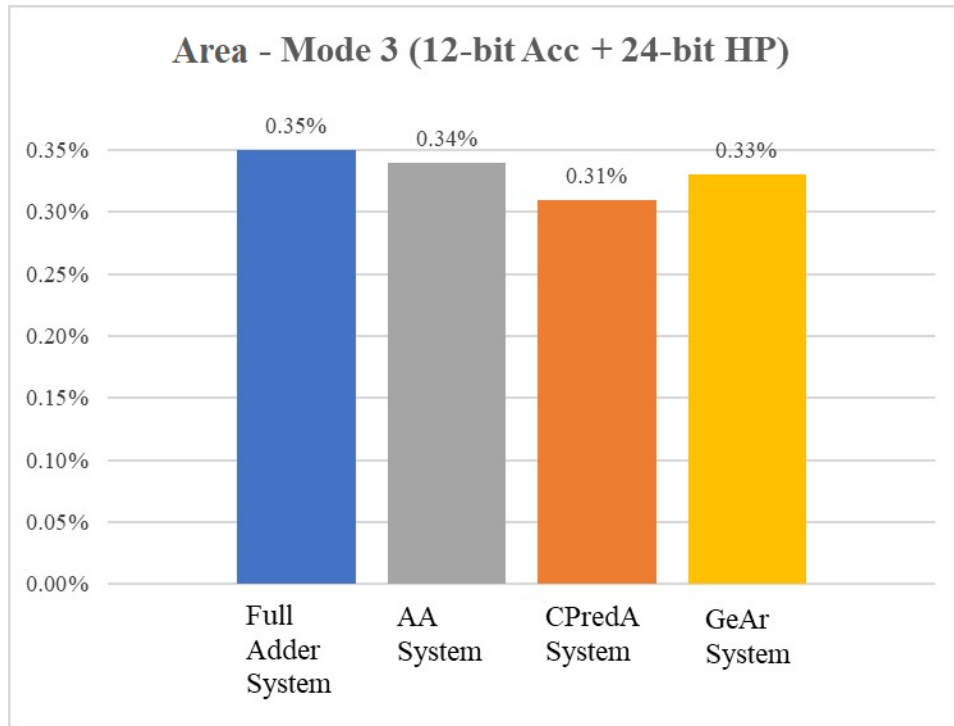


Figure 57: Area test results of processing systems in Mode 3

The accuracy of the three system versions is very high due to the low error rates produced at this level, and better system enhancements have been achieved compared to the previous modes. The system delay is reduced by 23% when using the GeAr adder, and the system area is also reduced by 11% when the CPredA adder is implemented. The AA system still shows the highest delay due to its circuit architecture where the carry output production is delayed due to the carry chain, then inverted to produce the sum.

#### 4.3.2.4 Mode 4

Both 12 and 24-bit adders are set to half prediction configuration in this mode. In half prediction configuration the lower half of the result is predicted, and the upper half is calculated accurately. Figure 58 shows a sample of the system output in this mode. It shows the original and filtered signals and the spikes detected through accurate and approximate systems. Both filtering and detection are almost identical in the accurate and approximate systems.

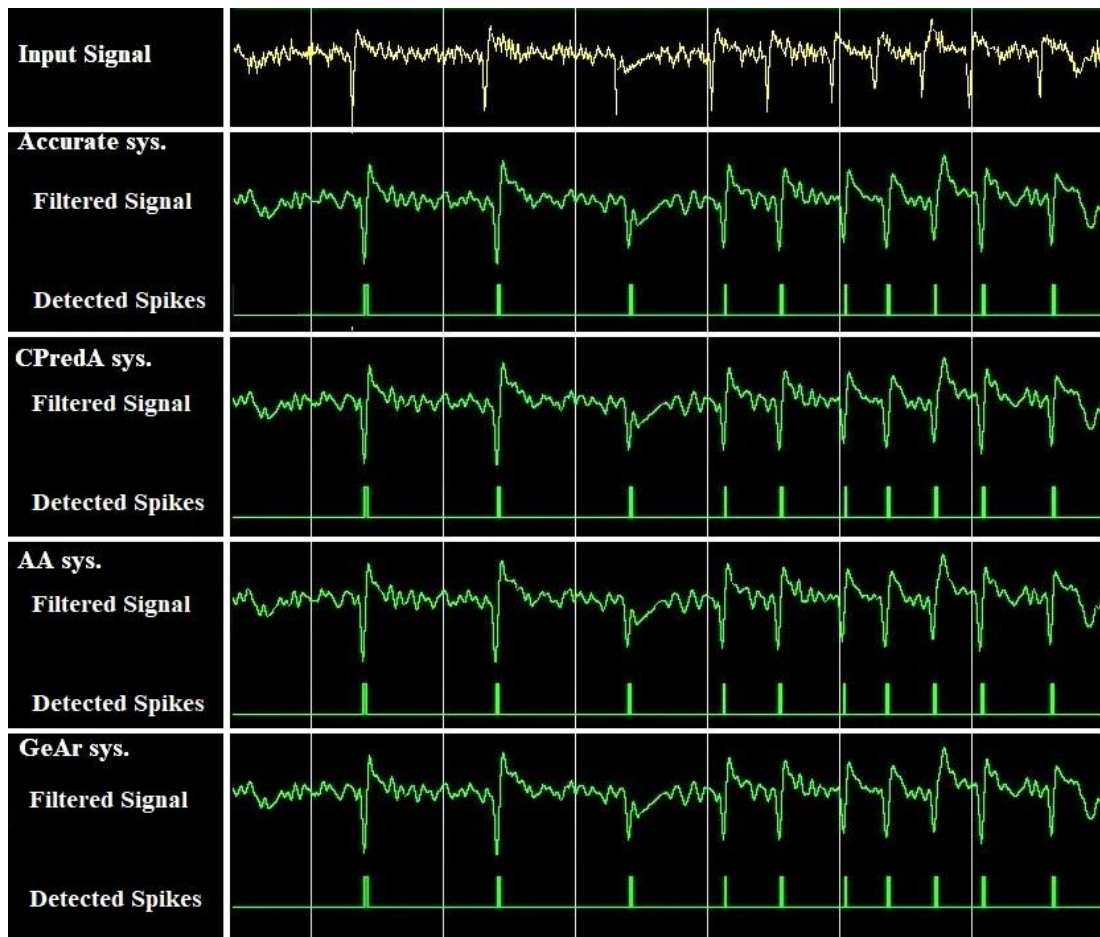


Figure 58: Sample test of signal filtering and spike detection in Mode 4

Figure 59-61 show the test results of accuracy, delay, and area for the three approximate systems and accurate system results for comparison.

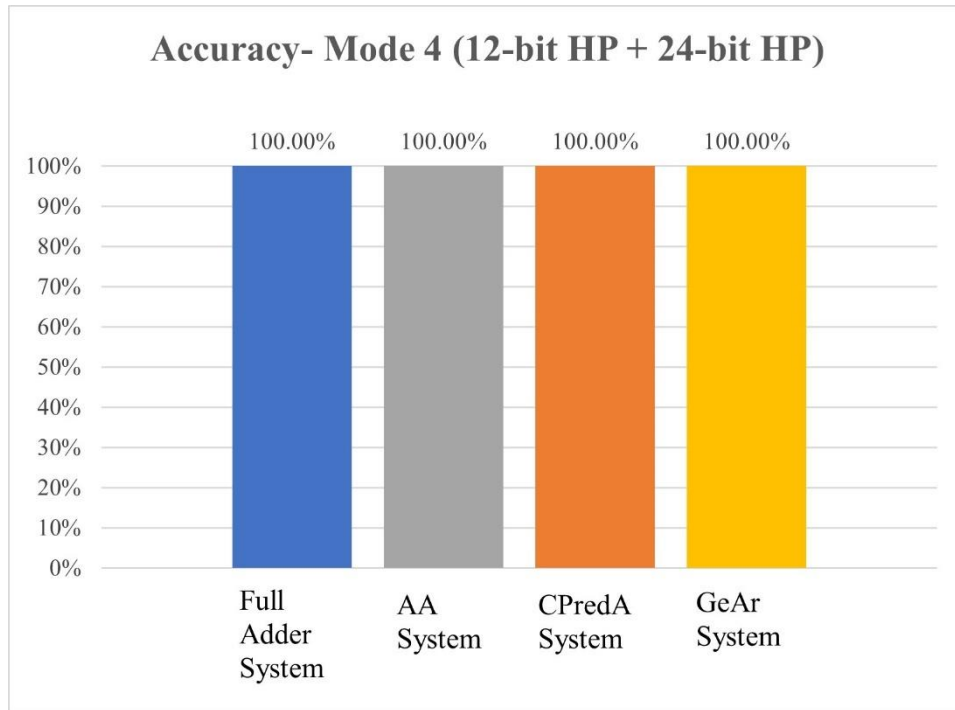


Figure 59: Accuracy test results of processing systems in Mode 4

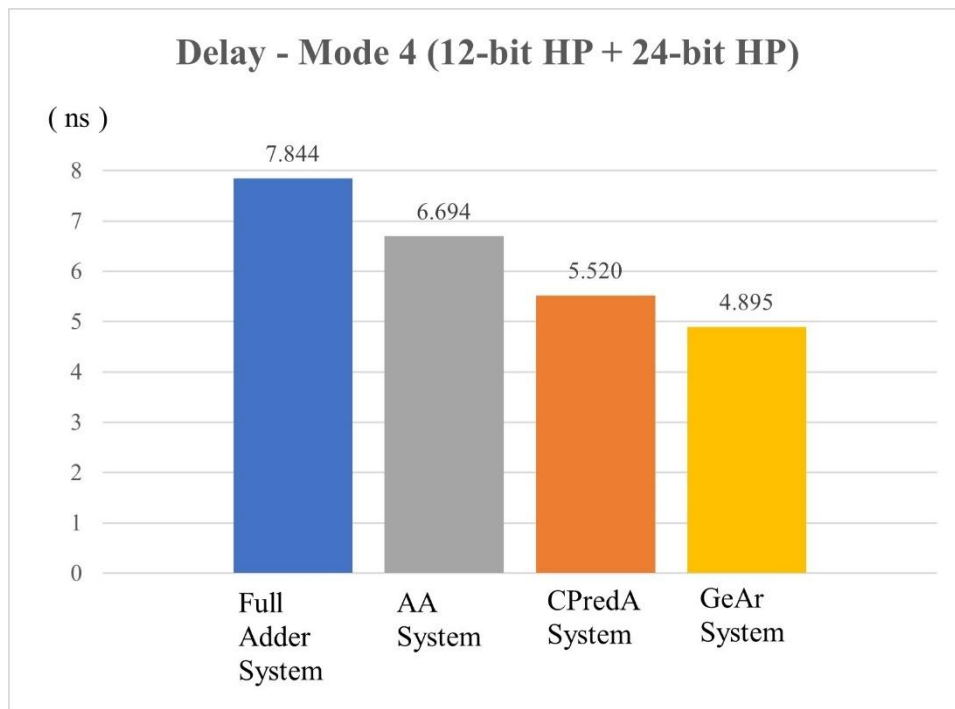


Figure 60: Delay time test results of processing systems in Mode 4

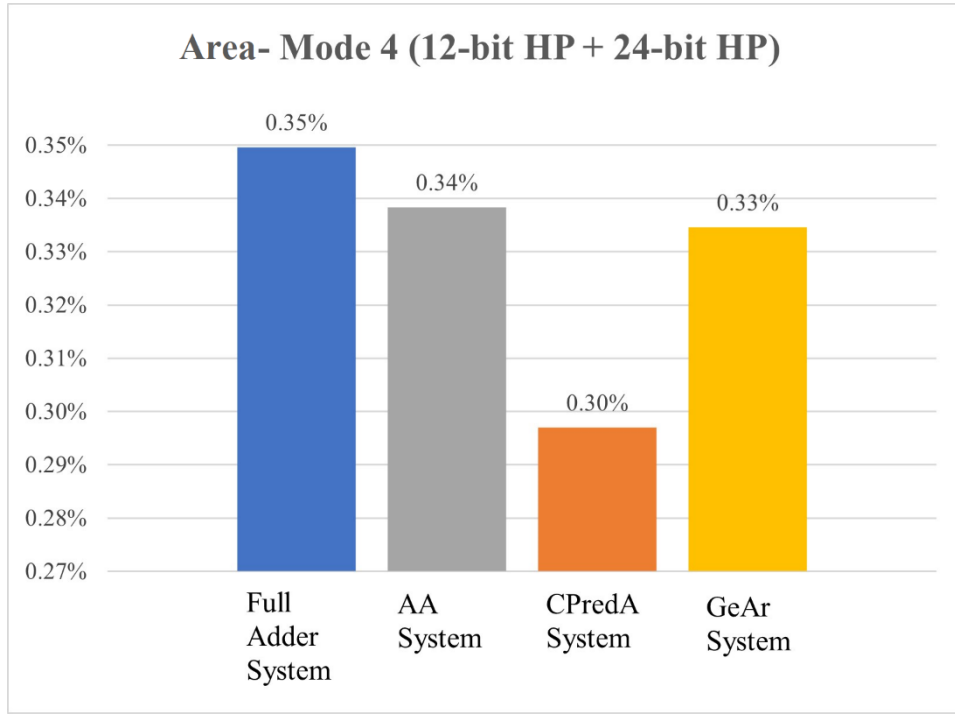


Figure 61: Area test results of processing systems in Mode 4

The error rates are very low even when all adders are predicting half of their results; hence the accuracy is very high in all system versions. On the other hand, the system delay has reduced significantly in the approximate implementations, where we can achieve around 37.6% reduction in time by using GeAr and a 29.6% when using CPredA which are producing half of their results in parallel in this mode. The area of the system is reduced from 0.35% in the accurate system to 0.30% in the CPredA system, therefore, 14.3% reduction in area is achieved since it has a simple circuit compared to the full adder. AA system still has a higher delay than other approximate systems due to the carry chain delay and its way in producing the sum after producing the carry.

All parameters have been enhanced significantly without losing the spike detection accuracy.

#### 4.3.2.5 Mode 5

The approximation level is slightly increased in the 12-bit adders. In this mode, 8 bits of the 12-bit adder results are predicted while the 24-bit adder results are kept half-predicted. Figure 62-64 show the testing results of this mode.

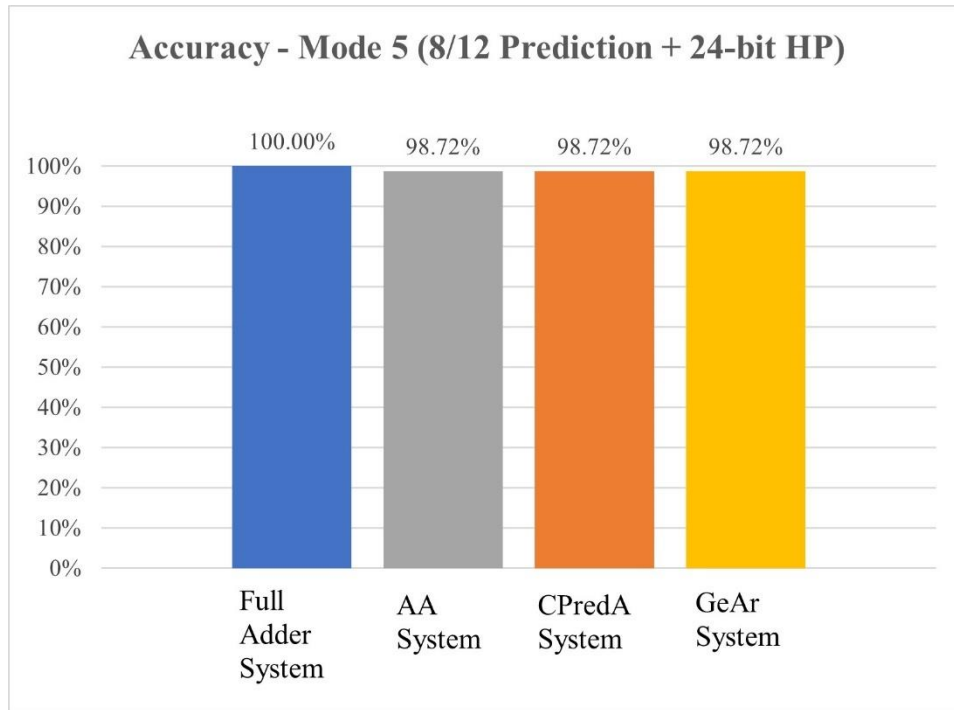


Figure 62: Accuracy test results of processing systems in Mode 5

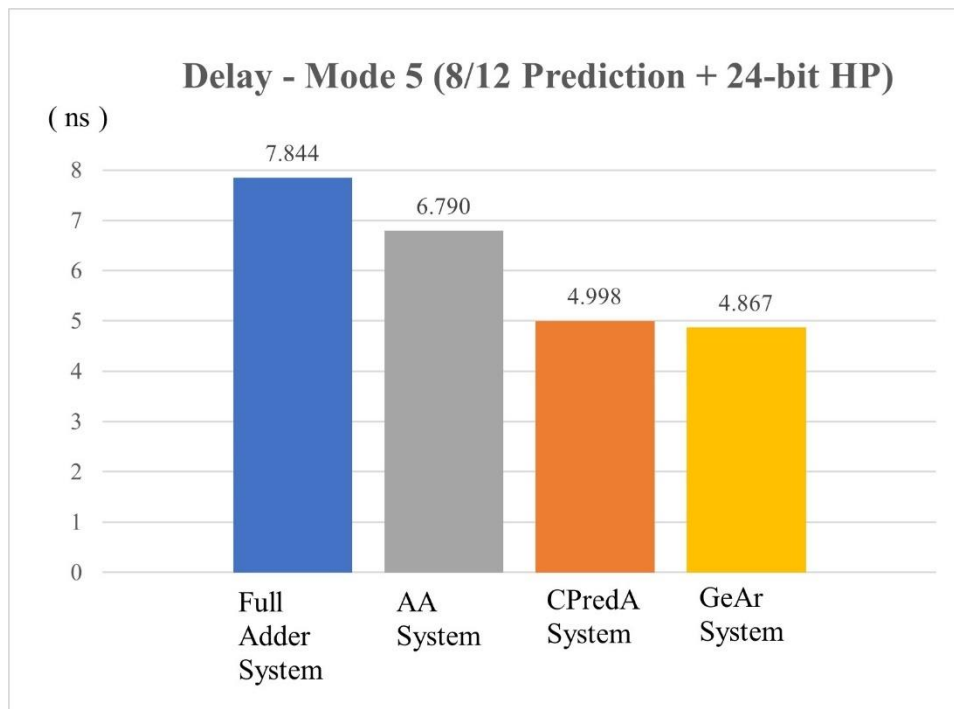


Figure 63: Delay time test results of processing systems in Mode 5



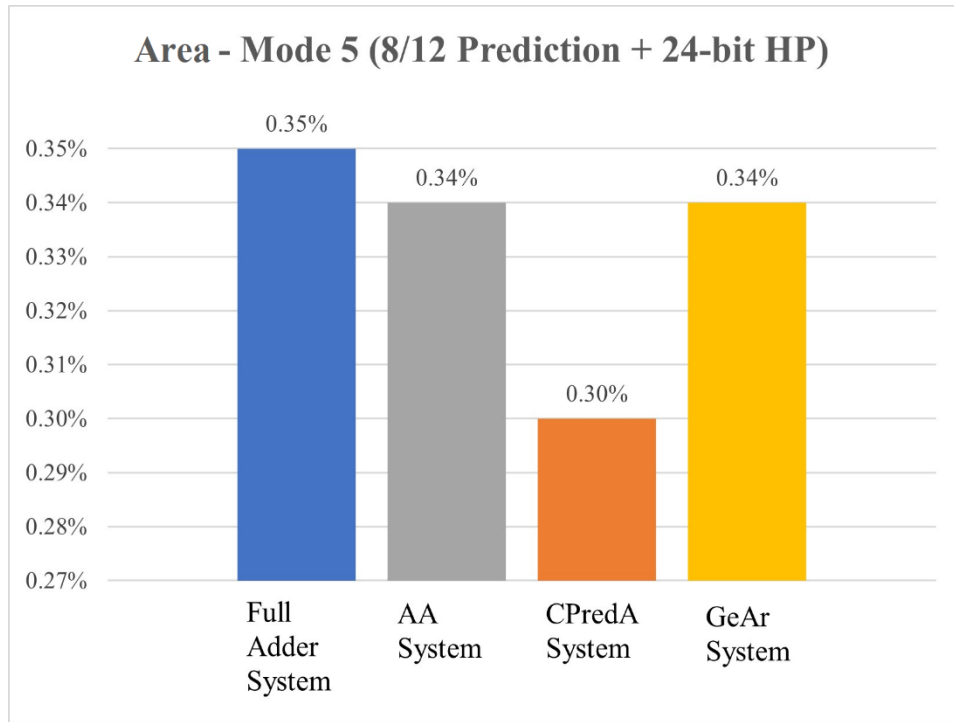


Figure 64: Area test results of processing systems in Mode 5

In mode 5, more system enhancement is obtained with satisfactory accuracy that is still high and acceptable. It reaches 98.7% in all approximate systems. The system delay is reduced further in this mode compared to the previous modes. It is reduced by 38% in the case of the GeAr system and 36.3% in the case of CPredA. However, the area of the systems is almost the same as the previous mode.

#### 4.3.2.6 Mode 6

In this mode, 10 bits of the 12-bit adder results are predicted while 24-bit adder results are kept half-predicted. Figure 65-67 show the test results of this mode.

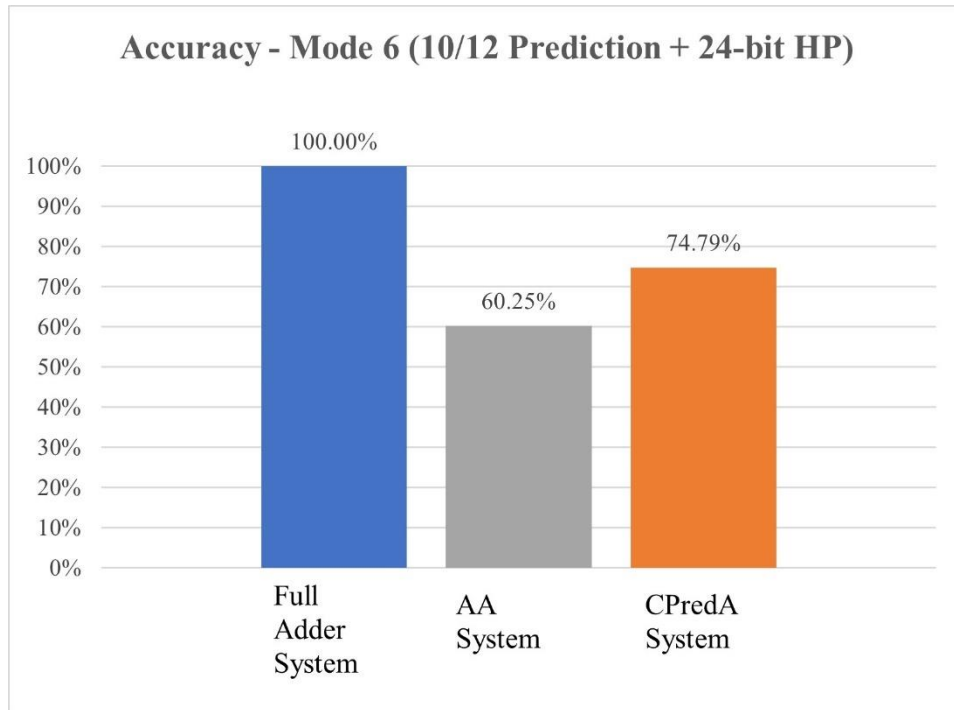


Figure 65: Accuracy test results of processing systems in Mode 6

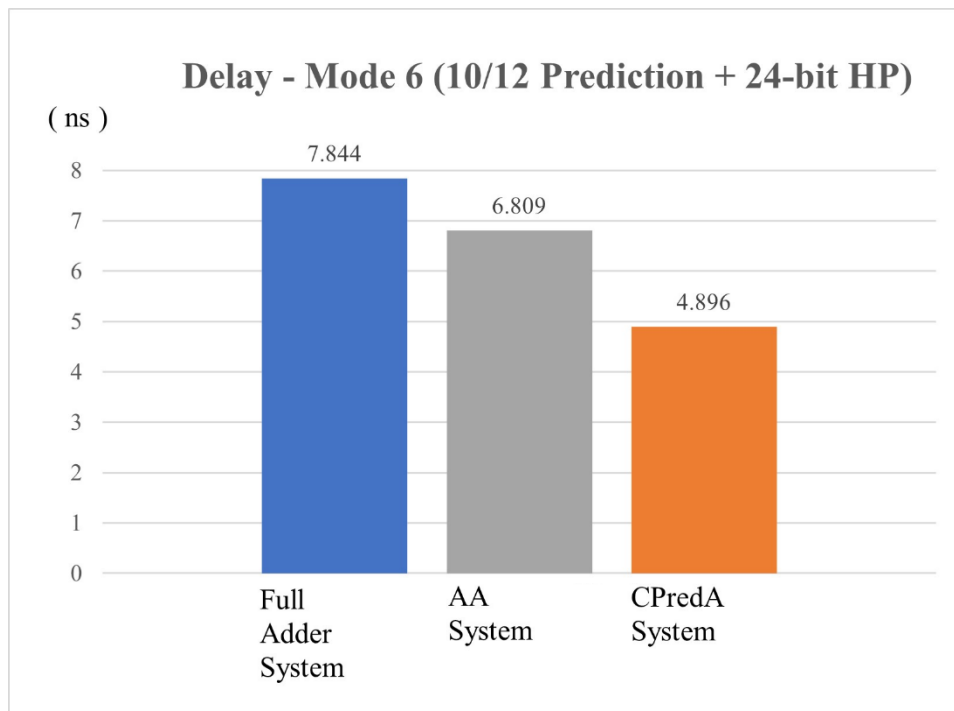


Figure 66: Delay time test results of processing systems in Mode 6

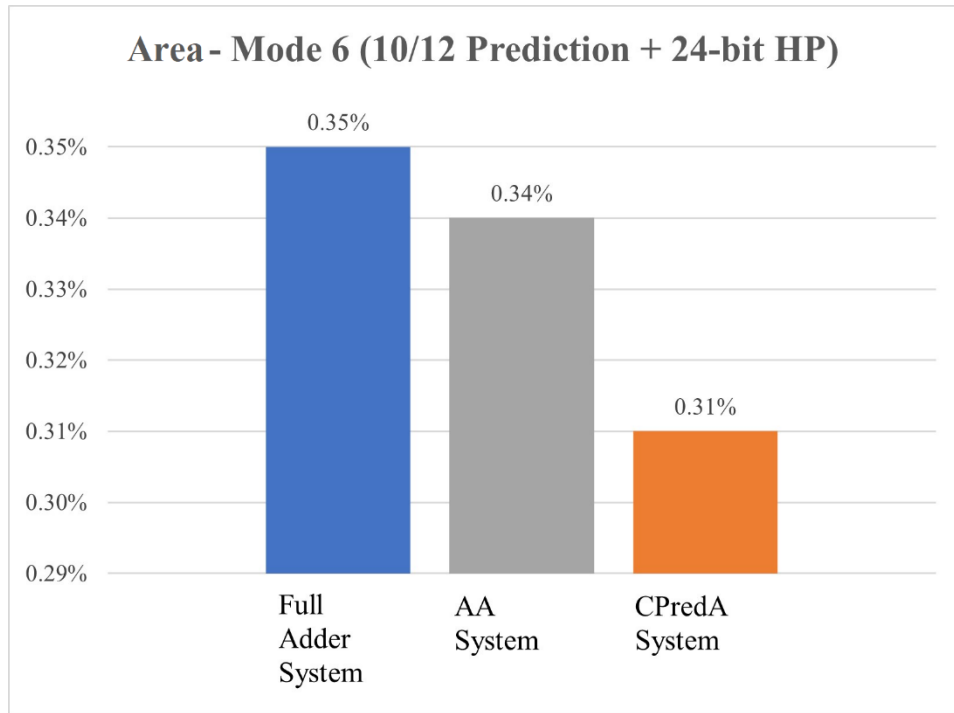


Figure 67: Area test results of processing systems in Mode 6

GeAr is not applicable in this mode since only the last two bits of the result will be left to be accurately calculated, which are less than the 4 bits required for prediction in each segment. In this mode, the accuracy has dropped to 75% in the CPredA system and more to 60% in AA2 systems. System speed has been enhanced more in the CPredA system to reach 37.6%, while the enhancement in area is less than in the previous modes since more chip resources are required to reach the low delay time.

#### 4.3.2.7 Mode 7

In this mode, 12-bit adder results are fully predicted, while the 24-bit adders are kept in half prediction mode. Figure 68-70 display the results of the tests.

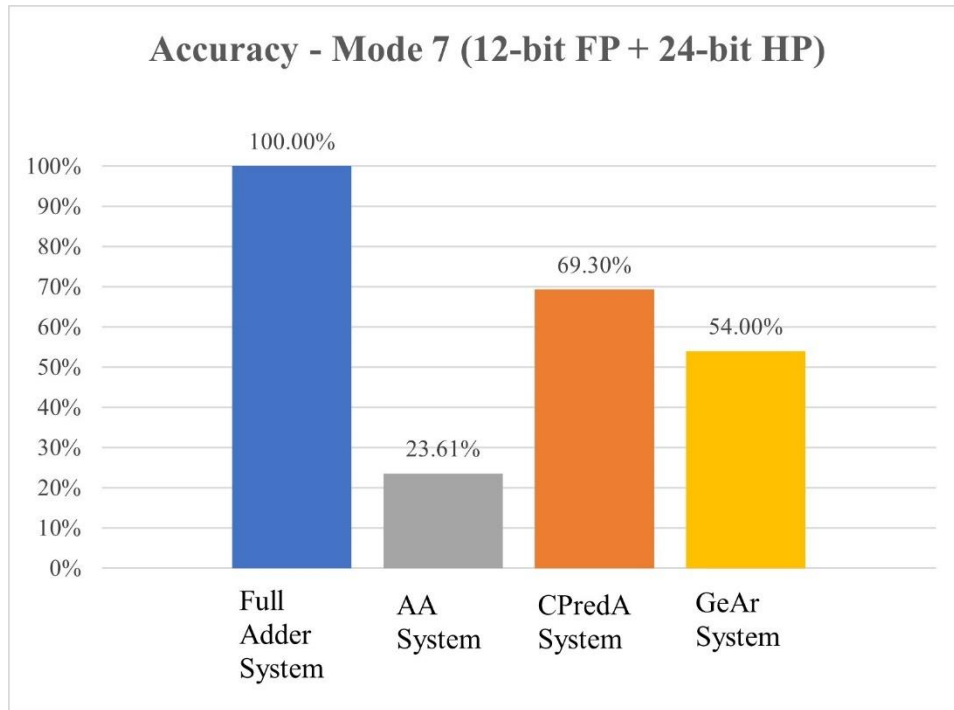


Figure 68: Accuracy test results of processing systems in Mode 7

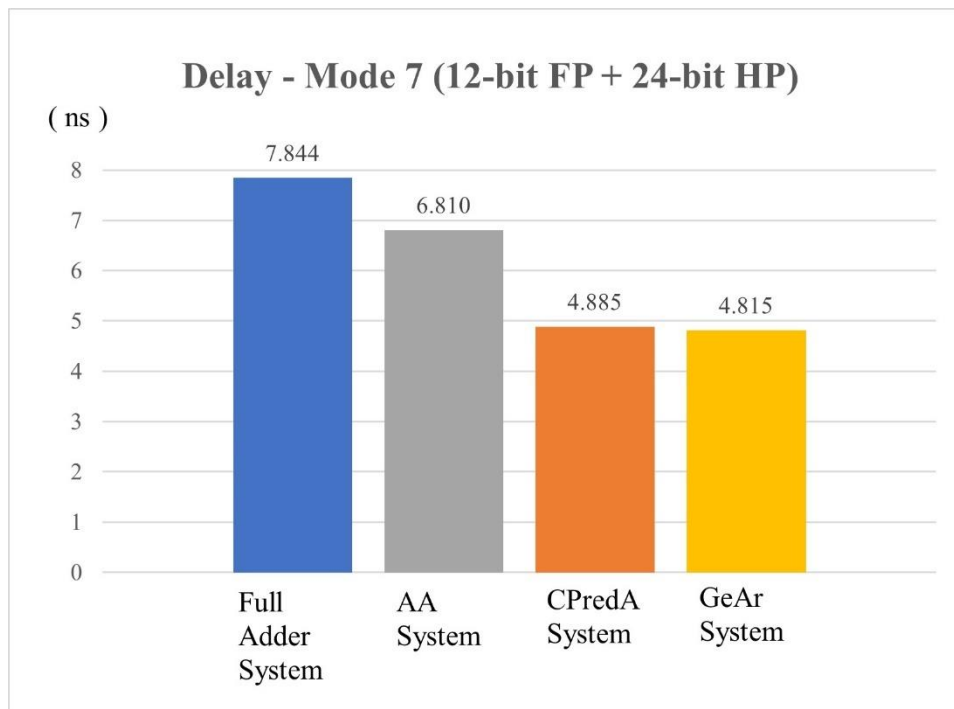


Figure 69: Delay time test results of processing systems in Mode 7

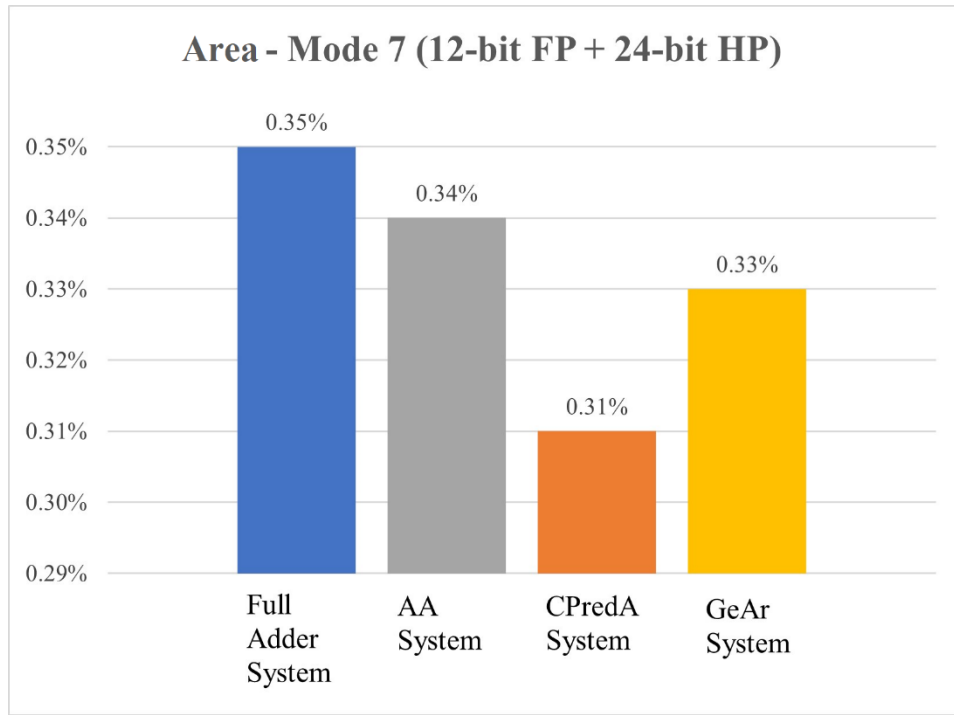


Figure 70: Area test results of processing systems in Mode 7

The maximum improvement in speed is gained in this mode by trading off the accuracy. The system speed enhancement reaches 38.6% in the GeAr system and 37.7% in the CPredA system compared to the full accuracy system. In contrast, accuracy has dropped to almost 70% in the CPredA system and to lower values in the GeAr and AA2 systems.

Table 11 summarizes the evaluation results for accurate and approximate systems in all testing modes. The power measurements of all approximate system versions in all modes are either slightly less or the same as the accurate system. Therefore, at the same power consumption or less, more improvements in speed and area are gained as the level of approximation increases with trading off the accuracy.

Table 11: Evaluation results for all systems

Mode	Adder Used	Accuracy %	Delay (ns)	Reduction in Time %	Area %	Reduction in Area %	Power (mW)
	Full Adder	100.00	7.844		0.35		0.127
<b>Mode 1</b> <b>HP + Acc</b>	AA	100.00	7.595	3.17	0.35	0.00	0.126
	CPredA	100.00	7.212	8.06	0.35	0.00	0.127
	GeAr	100.00	7.665	2.28	0.35	0.00	0.127
<b>Mode 2</b> <b>FP + Acc</b>	AA	25.32	7.267	7.36	0.34	2.86	0.126
	CPredA	83.00	6.757	13.86	0.32	8.57	0.127
	GeAr	57.03	7.257	7.48	0.33	5.71	0.127
<b>Mode 3</b> <b>Acc + HP</b>	AA	100.00	6.865	12.48	0.34	2.86	0.126
	CPredA	100.00	6.307	19.59	0.31	11.43	0.127
	GeAr	100.00	6.050	22.87	0.33	5.71	0.127
<b>Mode 4:</b> <b>HP + HP</b>	AA	100.00	6.694	14.66	0.34	2.86	0.126
	CPredA	100.00	5.520	29.63	0.30	14.29	0.127
	GeAr	100.00	4.895	37.60	0.33	5.71	0.127
<b>Mode 5:</b> <b>8/12 + HP</b>	AA	98.72	6.790	13.44	0.34	2.86	0.126
	CPredA	98.72	4.998	36.28	0.30	14.29	0.127
	GeAr	98.72	4.867	37.95	0.34	2.86	0.127
<b>Mode 6:</b> <b>10/12 + HP</b>	AA	60.25	6.809	13.19	0.34	2.86	0.127
	CPredA	74.79	4.896	37.58	0.31	11.43	0.127
	GeAr	N/A	N/A	N/A	N/A	N/A	N/A
<b>Mode 7:</b> <b>FP + HP</b>	AA	23.61	6.810	13.18	0.34	2.86	0.127
	CPredA	69.30	4.885	37.72	0.31	11.43	0.127
	GeAr	54.00	4.815	38.62	0.33	5.71	0.127

Table 12 shows the statistical analysis of all modes. The table shows that the Average (AVG) and the Standard Deviation (STD) of the output samples in the modes which have high detection accuracy are close to the values of the accurate system which uses the full adder in its calculations. On the other hand, the average and standard deviation values differ significantly as the detection accuracy goes lower in modes 2, 6, and 7.

Table 12: Statistical analysis of all modes

		Accurate System	Approximate Systems Modes						
			Mode 1	Mode2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
CPredA	Accuracy %	-	100	83	100	100	98.72	74.79	69.30
	STD	88.26	88.54	101.46	88.93	89.21	93.31	102.85	101.92
	AVG	-12.34	-13.71	-50.15	-14.91	-15.48	-24.11	-65.94	-65.74
AA2	Accuracy %	-	100	25.32	100	100	98.72	60.25	23.61
	STD	88.26	88.58	56.19	88.84	89.05	93.61	222.87	57.57
	AVG	-12.34	-13.39	-4.81	-10.88	-13.13	-14.35	-32.46	-4.29
GeAr	Accuracy %	-	100	57.03	100	100	98.72	NA	54.00
	STD	88.26	88.96	128.25	89.15	90.09	99.12	NA	128.51
	AVG	-12.34	-13.72	-65.94	-14.35	-14.83	-18.53	NA	-96.76

Processing biological signals with approximate computing shows practical improvements in processing time and circuit area. This depends on the type of adder and the level of approximation. Approximate adders with simplified circuits which break the carry chains and produce all outputs in parallel, like the CPredA adder, deliver generally significant enhancements in terms of reduced system delay and circuit area. In contrast, adders that use segmentation, such as GeAr, can improve delay, but not the circuit area. In this type of approximate adders, the full adder is divided into smaller segments that produce sub results in parallel. However, the total area of these segments is almost the same as the original full adder. Approximate adders with simplified circuits which does not break the carry chain, like AA adder, provide the minimum enhancements in delay and circuit area due to the time required to produce the results through the carry chain.

Half approximation of either 12 or 24-bit adders brings some improvements in the system performance, as shown in modes 1 and 2. Furthermore, fully approximating the adders will cause a significant disturbance in the processed signal, leading to many missed or false detections.

Mode 4 significantly enhances the speed and area without losing the system's accuracy. This mode can be used in scenarios where speed, area, and accuracy are all required. In this mode, all adders are configured to half approximation, i.e., the low significant bits of the result are approximated while the important high bits are calculated accurately. Therefore, the samples processed by the FIR and fed to the spike detector contain some errors, but they are still above the threshold required for spike detection.

More-speed requiring scenarios can use Mode 5 by sacrificing some of the system accuracy within acceptable margins.

Modes 6 and 7 can provide even higher speed for applications with low accuracy and area enhancement requirements. They do not provide reliable accuracy but they provide higher processing speed for those speed-demanding applications with low accuracy requirement. For instance, they can be used in a system that detects the existence of spikes in a neural signal but not their exact number. The accuracy of these modes can be enhanced if required by changing the threshold value, decreasing the approximation level of the 24-bit adders, or using different type of approximate adders with lower error distance.

To prove the concept of this work within the time constraints, a limited number of approximate adders are tested. Different approximate adders and multipliers may be tested in future works, and the proposed system may be connected to physical MEA devices for further investigations and testing.

This research can be further improved with the following steps:

1. Apply more approximate computing algorithms to implement more versions of the approximate system and benchmark them with the existing ones.
2. Apply the approximate computing algorithms to the system multipliers.
3. Improve the accuracy of the system at high levels of approximation by integrating error correction units in the approximate systems.

Furthermore, the proposed system can be applied to process different biological signals online and offline such as Electroencephalography (EEG), Electrocorticography



(ECoG), Local Field Potential (LFP), and Electromyography (EMG) by changing the type of the FIR filters. FIR filter can act as a low pass filter, high pass filter, or band pass filter by changing the values of its coefficients. Cutoff frequencies can be changed also in the same way according to the targeted biological signal.

## Chapter 5: Conclusion

Due to the significant advances in technology in the field of neuroscience, MEAs with thousands of electrodes are now available. These MEAs can instantly monitor and record the activity of thousands of neurons in parallel, requiring processing systems with low latency, reduced circuit area, and low power consumption. In neuroscience studies, the processing time of the MEA system is a critical factor, as the system must acquire data, process it, and generate feedback stimuli with the lowest possible latency. Additionally, the circuit area of the processing system is also an important consideration, as the system is composed of parallel processing sets.

Programmable devices, such as microcontrollers and FPGAs, have limited resources, so their usage must be optimized efficiently. Reducing the circuit area of the processing sets allows for building more sets on the same chip and handling the data channels in parallel. Furthermore, reducing the power consumption of the processing units enhances the portability of the system and decreases its overall power consumption.

In this thesis, a novel neural processing and spike detection system is proposed that reduces processing latency, circuit area, and power consumption by using the approximate computing paradigm in its calculations. The most computationally intensive parts of the system use approximate adders to achieve the minimum possible latency in processing signals and detecting spikes. The design of the FIR filters also considers additional enhancements, such as the minimum required filter order, parallelism, and symmetry.

Three versions of the approximate processing systems were implemented and tested at different approximation levels. They were based on three approximate adders: CPredA, GeAr, and AA2. All systems were built using Verilog hardware description language and implemented on an FPGA using the Xilinx Zynq-7000 All Programmable SoC on the ZedBoard as the target platform.

The results of using approximate computing in processing biological signals show effective enhancements in processing time and circuit area, depending on the type of adder and the level of approximation used. For example, CPredA, which has a simplified

circuit area and carry calculation, offers a reduction of up to 29.6% in processing time and 14.3% in circuit area without sacrificing system accuracy or increasing power consumption, even at half approximation mode. On the other hand, GeAr, which produces results through parallel segments, can offer a processing speed enhancement of 37.6% at half approximation. The research also presents different levels of approximation that can be used in neural processing systems to meet various demands for speed, accuracy, and area requirements. Further improvements in speed and area can be achieved by trading off spike detection accuracy and running the system in high approximation modes, where most adder results are predicted. These modes can be used in applications where the existence of spikes is important, but not their exact number.

As future work related to this thesis, different approximate adders and multipliers may be tested, and the proposed system may be connected to physical MEA devices for further testing and conclusions.

## References

- 3Brain. (2023). Retrieved April 01, 2023, from 3Brain Downloads:  
<https://www.3brain.com/resources/downloads#product-location>
- Ades, C., Abd, M. A., Du, E., Wei, J., Tognoli, E., & Engeberg, E. D. (2022). Robotically Embodied Biological Neural Networks to Investigate Haptic Restoration with Neuroprosthetic Hands. *2022 IEEE Haptics Symposium (HAPTICS)*, 1-7. <https://doi.org/10.1109/HAPTICS52432.2022.9765605>
- Andina, J. J., Arnanz, E. d., & Valdes, M. D. (2020). *FPGAs: Fundamentals, Advanced Features, and Applications* (pp. 22-25). CRC Press.  
<https://doi.org/10.1201/9781315162133>
- Angotzi, G. N., Boi, F., Zordan, S., Bonfanti, A., & Vato, A. (2014). A programmable closed-loop recording and stimulating wireless system for behaving small laboratory animals. *Scientific Reports*, 4(1), 5963.  
<https://doi.org/10.1038/srep05963>
- Bavishi, S., Rosenthal, J., & Bockbrader, M. (2019). Neuroprosthetics. *In Rehabilitation After Traumatic Brain Injury* (pp. 241-253). Elsevier.  
<https://doi.org/10.1016/B978-0-323-54456-6.00017-7>
- Betts, J. G., Young, K. A., Wise, J. A., Johnson, E., Poe, B., Kruse, D. H., Korol, O., Johnson, J. E., Womble, M., & DeSaix, P. (2013). *Anatomy and Physiology* (pp. 523-526). OpenStax. Retrieved March 29, 2023, from  
<https://openstax.org/books/anatomy-and-physiology/pages/1-introduction>
- Bhargav, A., & Huynh, P. (2021). Design and Analysis of Low-Power and High Speed Approximate Adders Using CNFETs. *Sensors*, 21(24), 8203.  
<https://doi.org/10.3390/s21248203>
- Bhaskara, S., Sakorikar, T., Chatterjee, S., Girishan, K. S., & Pandya, H. J. (2022). Recent advancements in Micro-engineered devices for surface and deep brain animal studies: A review. *Sensing and Bio-Sensing Research*, 36, 100483.  
<https://doi.org/10.1016/j.sbsr.2022.100483>
- Biffi, E., Ghezzi, D., Pedrocchi, A., & Ferrigno, G. (2010). Development and Validation of a Spike Detection and Classification Algorithm Aimed at Implementation on Hardware Devices. *Computational Intelligence and Neuroscience*, 2010.  
<https://doi.org/10.1155/2010/659050>

- Bosio, A., Virazel, A., Girard, P., & Barbareschi, M. (2017). Approximate computing: Design & test for integrated circuits. *2017 18th IEEE Latin American Test Symposium (LATS)*, 1-1. <https://doi.org/10.1109/LATW.2017.7906737>
- Chowdhury, M. H., Elyahoodayan, S., Song, D., & Cheung, R. C. (2020). An FPGA-Based Neuron Activity Extraction Unit for a Wireless Neural Interface. *Electronics*, 9(11), 1834. <https://doi.org/10.3390/electronics9111834>
- Cong, P., Karande, P., Landes, J., Corey, R., Stanslaski, S., Santa, W., Jensen, R., Pape, F., Moran, D., & Denison, T. (2014). A 32-channel modular bi-directional neural interface system with embedded DSP for closed-loop operation. *ESSCIRC 2014 - 40th European Solid State Circuits Conference (ESSCIRC)*, 99-102. <https://doi.org/10.1109/ESSCIRC.2014.6942031>
- Doliwa, S., Erbsloh, A., Seidl, K., & Iossifidis, I. (2022). Development of an Analog Front-End for Brain-Computer Interfaces. *2022 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, 309-312. <https://doi.org/10.1109/PRIME55000.2022.9816757>
- Dragas, J., Viswam, V., Shadmani, A., Chen, Y., Bounik, R., Stettler, A., Radivojevic, M., Geissler, S., Obien, M. E. J., Muller, J., & Hierlemann, A. (2017). In Vitro Multi-Functional Microelectrode Array Featuring 59760 Electrodes, 2048 Electrophysiology Channels, Stimulation, Impedance Measurement, and Neurotransmitter Detection Channels. *IEEE Journal of Solid-State Circuits*, 52(6), 1576-1590. <https://doi.org/10.1109/JSSC.2017.2686580>
- Gorantla, A., & Deepa, P. (2020). Design of approximate adders and multipliers for error tolerant image processing. *Microprocessors and Microsystems*, 72, 102940. <https://doi.org/10.1109/iSES.2018.00029>
- Gross, G. W., Rieske, E., Kreutzberg, G. W., & Meyer, A. (1977). A new fixed-array multi-microelectrode system designed for long-term monitoring of extracellular single unit neuronal activity in vitro. *Neuroscience Letters*, 6(3), 101-105. [https://doi.org/10.1016/0304-3940\(77\)90003-9](https://doi.org/10.1016/0304-3940(77)90003-9)
- Huang, P., Wang, C., Liu, W., Qiao, F., & Lombardi, F. (2021). A Hardware/Software Co-Design Methodology for Adaptive Approximate Computing in Clustering and ANN Learning. *IEEE Open Journal of the Computer Society*, 2, 38-52. <https://doi.org/10.1109/OJCS.2021.3051643>
- Huang, W.-C., Lei, W.-L., & Peng, C.-W. (2023). Fabrication of Biodegradable Soft Tissue-Mimicked Microelectrode Arrays for Implanted Neural Interfacing. *IEEE 36th International Conference on Micro Electro Mechanical Systems (MEMS)*, 392-395. <https://doi.org/10.1109/MEMS49605.2023.10052266>

- Kim, G., Kim, K., Lee, E., An, T., Choi, W., Lim, G., & Shin, J. (2018). Recent Progress on Microelectrodes in Neural Interfaces. *Materials*, 11(10), 1995. <https://doi.org/10.3390/ma11101995>
- Kozai, T. D. (2018). The History and Horizons of Microscale Neural Interfaces. *Micromachines*, 9(9), 445. <https://doi.org/10.3390/mi9090445>
- Lee, H.-S., Park, H., & Lee, H.-M. (2020). A Multi-Channel Neural Recording System with Adaptive Electrode Selection for High-Density Neural Interface. *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, 4306-4309. <https://doi.org/10.1109/EMBC44109.2020.9175670>
- Liang, J., Han, J., & Lombardi, F. (2013). New Metrics for the Reliability of Approximate and Probabilistic Adders. *IEEE Transactions on Computers*, 62(9), 1760-1771. <https://doi.org/10.1109/TC.2012.146>
- Liu, X., Zhang, M., Richardson, A. G., Lucas, T. H., & Spiegel, J. V. (2017). Design of a closed-loop, bidirectional brain machine interface system with energy efficient neural feature extraction and pid control. *IEEE Transactions on Biomedical Circuits and Systems*, 11(4), 729-742. <https://doi.org/10.1109/TBCAS.2016.2622738>
- Liu, Z., Sun, Z., Shi, G., Wu, J., & Xie, X. (2018). A Novel Algorithm for Online Spike Detection. *MATEC Web of Conferences*, 173, 02017. <https://doi.org/10.1051/mateconf/201817302017>
- Liu, Z., Tang, J., Gao, B., Li, X., Yao, P., Lin, Y., Liu, D., Hong, B., Qian, H., & Wu, H. (2020). Multichannel parallel processing of neural signals in memristor arrays. *Science Advances*, 6(41), 47-97. <https://doi.org/10.1126/sciadv.abc4797>
- Masadeh, M., Hasan, O., & Tahar, S. (2018). Comparative Study of Approximate Multipliers. *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 415-418. <https://doi.org/10.1145/3194554.3194626>
- Massobrio, P., Tessadori, J., Chiappalone, M., & Ghirardi, a. M. (2015). In Vitro Studies of Neuronal Networks and Synaptic Plasticity in Invertebrates and in Mammals Using Multielectrode Arrays. *Neural Plasticity*, 2015, 1-18. <https://doi.org/10.1155/2015/196195>
- Matyukha, V., Voloshchuk, S., & Mosin, S. (2022). A Configurable IP Core for Calculating the Integer Square Root for Serial and Parallel Implementations in FPGA. *Electronics*, 11(15), 2335. <https://doi.org/10.3390/electronics11152335>

- Muller, J., Bakkum, D., & Hierlemann, A. (2013). Sub-millisecond closed-loop feedback stimulation between arbitrary sets of individual neurons. *Frontiers Neural Circuits*, 6. <https://doi.org/10.3389/fncir.2012.00121>
- Newman, J. P., Zeller-Townson, R., Fong, M.-F., Arcot Desai, S., Gross, R. E., & Potter, S. M. (2013). Closed-loop, multichannel experimentation using the open-source neurorighter electrophysiology platform. *Frontiers Neural Circuits*, 6. <https://doi.org/10.3389/fncir.2012.00098>
- Obien, M. E., Gong, W., Frey, U., & Bakkum, D. J. (2017). CMOS-Based High-Density Microelectrode Arrays: Technology and Applications. In *Emerging Trends in Neuro Engineering and Neural Computation* (pp. 3-39). Springer. [https://doi.org/10.1007/978-981-10-3957-7\\_1](https://doi.org/10.1007/978-981-10-3957-7_1)
- O'Loughlin, D., Coffey, A., Callaly, F., Lyons, D., & Morgan, F. (2014). Xilinx Vivado High Level Synthesis: Case studies. *25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CICT 2014)*, 352-356. <https://doi.org/10.1049/cp.2014.0713>
- Özcan, Z., Yıldırım, Ö., & Kayıkcıoğlu, T. (2021). BroomyCell: An Introductory Educational Software for Neuroscience. *2021 29th Signal Processing and Communications Applications Conference (SIU)*, 1-4. <https://doi.org/10.1109/SIU53274.2021.9477851>
- Pal, R. (2017). Comparison of the design of FIR and IIR filters for a given specification and removal of phase distortion from IIR filters. *2017 International Conference on Advances in Computing, Communication and Control (ICAC3)*, 1-3. <https://doi.org/10.1109/ICAC3.2017.8318772>
- Park, J., Kim, G., & Jung, S.-D. (2017). A 128-channel FPGA based realtime spike-sorting bidirectional closed-loop neural interface system. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(12), 2227-2238. <https://doi.org/10.1109/TNSRE.2017.2697415>
- Perepelitsyn, A., & Kulanov, V. (2022). Technologies of FPGA-based projects Development Under Ever-changing Conditions, Platform Constraints, and Time-to-Market Pressure. *2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 1-5. <https://doi.org/10.1109/DESSERT58054.2022.10018828>
- Pine, J. (2006). A History of MEA Development. In *Advances in Network Electrophysiology: Using Multi-Electrode Arrays* (pp. 3-23). Springer US. [https://doi.org/10.1007/0-387-25858-2\\_1](https://doi.org/10.1007/0-387-25858-2_1)

- Priyadharshni, M., & Kumaravel, S. (2019). A Comparative Exploration About Approximate Full Adders for Error Tolerant Applications. *International Symposium on VLSI Design and Test*, 61-74. [https://doi.org/10.1007/978-981-13-5950-7\\_6](https://doi.org/10.1007/978-981-13-5950-7_6)
- Raghuram, S., & Shashank, N. (2022). Approximate Adders for Deep Neural Network Accelerators. *2022 35th International Conference on VLSI Design*, 210-215. <https://doi.org/10.1109/VLSID2022.2022.00049>
- Rey, H. G., Pedreira, C., & Quiroga, R. Q. (2015). Past, present and future of spike sorting techniques. *Brain Research Bulletin*, 119, 106–117. <https://doi.org/10.1016/j.brainresbull.2015.04.007>
- Saggese, G., & Strollo, A. (2022). Low-Power Energy-Based Spike Detector ASIC for Implantable Multichannel BMIs. *Electronics*, 11(18), 2943. <https://doi.org/10.3390/electronics11182943>
- Saggese, G., Tambaro, M., Vallicelli, E. A., Strollo, A. G., Vassanelli, S., Baschiroto, A., & Matteis, M. D. (2021). Comparison of Sneo-Based Neural Spike Detection Algorithms for Implantable Multi-Transistor Array Biosensors. *Electronics*, 10(4), 410. <https://doi.org/10.3390/electronics10040410>
- Sato, T., Yang, T., & Ukezono, T. (2019). Trading Accuracy for Power with a Configurable Approximate Adder. *IEICE Transactions on Electronics*, E102.C(4), 260-268. <https://doi.org/10.1587/transele.2018CDP0001>
- Seu, G. P., Angotzi, G. N., Boi, F., Raffo, L., Berdondini, L., & Meloni, P. (2018). Exploiting All Programmable SoCs in Neural Signal Analysis: A Closed-Loop Control for Large-Scale CMOS Multielectrode Arrays. *IEEE Transactions on Biomedical Circuits and Systems*, 12(4), 839-850. <https://doi.org/10.1109/TBCAS.2018.2830659>
- Shi, J., & Fang, Y. (2018). Flexible and Implantable Microelectrodes for Chronically Stable Neural Interfaces. *Advanced Materials*, 31(45), 1804895. <https://doi.org/10.1002/adma.201804895>
- Shulyzki, R., Abdelhalim, K., Bagheri, A., Salam, M. T., Florez, C. M., Velazquez, J. L. P., Carlen, P. L., & Genov, R. (2015). 320-Channel Active Probe for High-Resolution Neuromonitoring and Responsive Neurostimulation. *IEEE Transactions on Biomedical Circuits and Systems*, 9(1), 34-49. <https://doi.org/10.1109/TBCAS.2014.2312552>



- Slepova, L. O., & Berkhova, E. M. (2019). Development of a System Providing the Interaction of the Transducer with the Nervous System for Sensory Prosthetics. *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 1233-1236.  
<https://doi.org/10.1109/EIConRus.2019.8657210>
- Strollo, A. (2021). A Low Power 1024-Channels Spike Detector Using Latch-Based RAM for Real-Time Brain Silicon Interfaces. *Electronics*, 10(24), 3068.  
<https://doi.org/10.3390/electronics10243068>
- Strumwasser, F. (1958). Long-Term Recording from Single Neurons in Brain of Unrestrained Mammals. *Science*, 127(3296), 469-470.  
<https://doi.org/10.1126/science.127.3296.469>
- Tanskanen, J. M., Ahtiainen, A., & Hyttinen, J. A. (2020). Toward Closed-Loop Electrical Stimulation of Neuronal Systems: A Review. *Bioelectricity*, 2(4), 328-347. <https://doi.org/10.1089/bioe.2020.0028>
- Thomas, C., Springer, P., Loeb, G. B.-N., & Okun, L. (1972). A miniature microelectrode array to monitor the bioelectric activity of cultured cells. *Experimental Cell Research*, 74(1), 61-66. [https://doi.org/10.1016/0014-4827\(72\)90481-8](https://doi.org/10.1016/0014-4827(72)90481-8)
- Varier, P., Raju, G., Madhusudanan, P., Jerard, C., & Shankarappa, S. (2022). A Brief Review of In Vitro Models for Injury and Regeneration in the Peripheral Nervous System. *International Journal of Molecular Sciences*, 23(2), 816.  
<https://doi.org/10.3390/ijms23020816>
- Venkatraman, S., Elkabany, K., Long, J. D., Yao, Y., & Carmena, J. M. (2009). A system for neural recording and closed-loop intracortical microstimulation in awake rodents. *IEEE Transactions on Biomedical Engineering*, 56(1), 15-22.  
<https://doi.org/10.1109/TBME.2008.2005944>
- Verma, A. K., Brisk, P., & Ienne, P. (2008). Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design. *2008 Design, Automation and Test in Europe*, 1250-1255. <https://doi.org/10.1109/DATE.2008.4484850>
- Wallach, A., Eytan, D., Gal, A., Zrenner, C., & Marom, S. (2011). Neuronal Response Clamp. *Frontiers in Neuroengineering*, 4.  
<https://doi.org/10.3389/fneng.2011.00003>

- Wang, L., Guo, Z., Ji, B., Xi, Y., Yang, B., & Liu, J. (2021). A High-Density Drivable Microelectrodes Array for Multi-Brain Recording. *21st International Conference on Solid-State Sensors, Actuators and Microsystems (Transducers)*, 679-682. <https://doi.org/10.1109/Transducers50396.2021.9495647>
- Wang, Z., Zhang, G., Ye, J., Jiang, J., Li, F., & Wang, Y. (2022). Accurate Reliability Analysis Methods for Approximate Computing Circuits. *Tsinghua Science and Technology*, 27(4), 729-740. <https://doi.org/10.26599/TST.2020.9010032>
- Yang, Z., Jain, A., Liang, J., Han, J., & Lombardi, F. (2013). Approximate XOR/XNOR-based Adders for Inexact Computing. *2013 13th IEEE International Conference on Nanotechnology (IEEE-NANO 2013)*, 690-693. <https://doi.org/10.1109/NANO.2013.6720793>
- Ye, J., Yanagisawa, M., & Shi, Y. (2022). Scalable Hardware Efficient Architecture for Parallel FIR Filters with Symmetric Coefficients. *Electronics*, 11(20), 3272. <https://doi.org/10.3390/electronics11203272>
- Ye, R., Wang, T., Yuan, F., Kumar, R., & Xu, Q. (2013). On Reconfiguration-Oriented Approximate Adder Design and Its Application. *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 48-54. <https://doi.org/10.1109/ICCAD.2013.6691096>
- ZedBoard. (2023). Retrieved April 01, 2023, from <https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/zedboard/zedboard-board-family>
- Zhang, X., Li, Q., Chen, C., Li, Y., Zuo, F., Liu, X., Zhang, H., Wang, X., & Liu, Y. (2021). A Fully Integrated 64-Channel Recording System for Extracellular Raw Neural Signals. *Electronics*, 10(21), 2726. <https://doi.org/10.3390/electronics10212726>
- Zhang, Z., & Constandinou, T. (2021). Adaptive spike detection and hardware optimization towards autonomous, high-channel-count BMIs. *Journal of Neuroscience Methods*, 354, 109103. <https://doi.org/10.1016/j.jneumeth.2021.109103>
- Zoladz, M., Kmon, P., Rauza, J., Grybos, P., Kowalczyk, T., & Caban, B. (2013). A complete 256-channel reconfigurable system for in vitro neurobiological experiments. *2013 IEEE Biomedical Circuits and Systems (BIOCAS)*, 250-253. <https://doi.org/10.1109/BioCAS.2013.6679686>

Zrenner, C., Eytan, D., Wallach, A., Thier, P., & Marom, S. (2010). A Generic Framework for Real-Time Multi-Channel Neuronal Signal Analysis, Telemetry Control, and Sub-Millisecond Latency Feedback Generation. *Frontiers Neuroscience*, 4. <https://doi.org/10.3389/fnins.2010.00173>

## List of Publications

- Hassan, M., Awwad, F., Atef, M., & Hasan, O. (2023). Approximate Computing-Based Processing of MEA Signals on FPGA. *MDPI Electronics*, 12(4), 848. <https://doi.org/10.3390/electronics12040848>
- Memon, Q., & Hassan, M. (2020). Cluster Analysis of Patients' Clinical Information for Medical Practitioners and Insurance Companies. *International Journal of Online & Biomedical Engineering*, 16(04), 128-138. <https://doi.org/10.3991/ijoe.v16i04.13119>
- Memon, Q., & Hassan, M. (2018). Detecting Computer Vision Syndrome Using Eye Blink—An Experimental Evaluation. *Journal of Physics: Conference Series*, 1098(1), 012029. <https://doi.org/10.1088/1742-6596/1098/1/012029>

**UAEU**جامعة الإمارات العربية المتحدة  
United Arab Emirates University

## UAE UNIVERSITY DOCTORATE DISSERTATION NO. 2023:9

This dissertation proposes enhancing the capability of MEA signal processing systems by using approximate computing-based algorithms. These algorithms can be implemented in systems that process parallel MEA channels using FPGAs. Real biological signals are used to evaluate both precise and approximative systems with three approximate algorithms. The findings reveal notable improvements, especially in terms of speed and area. Processing speed enhancements reach up to 37.6%, and area enhancements reach 14.3% in some approximate system modes without sacrificing accuracy.

**Mohammad Hassan** received his PhD in Electrical Engineering from the Department of Electrical and Communication Engineering, College of Engineering at UAE University, UAE. He also received his Master of Science in Electrical Engineering from the College of Engineering, UAE University, UAE.

[www.uaeu.ac.ae](http://www.uaeu.ac.ae)