



Newcastle University, UK
School of Engineering
Microsystems Group

RUN-TIME CONFIGURABLE APPROXIMATE MULTIPLIER DESIGN

Ibrahim Aref Haddadi

A thesis presented for the degree of Doctor of Philosophy
August 14, 2023

Copyright Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures. The copyright of this thesis rests with the author. Copies (by any means) either in full or of extracts, may not be made without prior written consent from the author. Copyright ©2023 Ibrahim Aref Haddadi, all rights reserved.

Signed: Ibrahim Aref Haddadi

Date: August 14, 2023

Certificate of Approval

We confirm that, to the best of our knowledge, this thesis is from the student's own work and effort, and all other sources of information used have been acknowledged. This thesis has been submitted with my approval.

Dr. Rishad Shafik
Prof. Alex Yakovlev
Dr. Fei Xia



— *Until we meet again* —

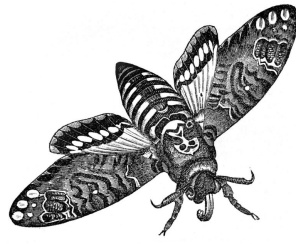
In loving memory of my Brother "Slam" (1988-2022)

In loving memory of my beautiful Grandma "Lala" (1930-2022)

*To the soul of my Father "Aref" who was and will always be my role model
(1962-2020)*

In loving memory of my special Uncle "Abdouslam" (1965-2015)

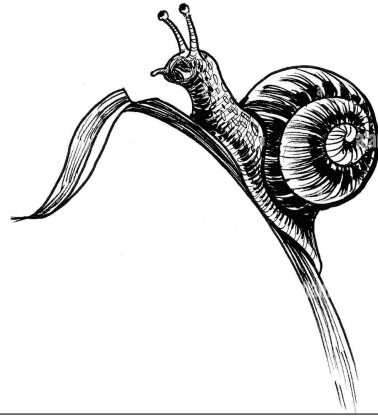
— Ibrahim —



To my beloved Mother "Rana"



To my Wonderful Wife "Eman". Thank you my darling for being with me abroad far from our country (The Kingdom of Saudi Arabia (KSA)) and for the unconditional support and love to fulfil such success.



To my lovely kids "Kyan" and "Aref"

Acknowledgements

I would like to express my deep gratitude to my supervisors Dr. Rishad Shafik, Prof. Alex Yakovlev and Dr. Fei Xia for their support and guidance through my PhD journey. Rishad and I have shared many philosophical discussions and he has guided me to become the researcher I am today. Alex has pushed me to share my research widely and has helped me to discover the commercial world. Dr. Fei Xia has been there to give practical advice in all aspects of electronic design and writing. They have always been a source of motivation and my inspirational as a researcher. I am grateful to the College of Computer Science and Engineering (CCSE), TAIBAH University, Kingdom of Saudi Arabia for funding my PhD study and offering this place to finish my study abroad.

I would like also to express my gratefulness and appreciation to my colleagues and friends in school of Engineering, especially in MicroSystems Research Group. We have discussed many topics over the years, and they were very helpful. Special thanks to my colleagues Dr. Turki Alnuayri and my friend Dr. Issa Qiqieh for their help and support. I hope they continue to be successful with their research and future careers.

Abstract

The complexity of arithmetic continues to be an issue in the design of high-performance and energy-efficient hardware. The problem is further exacerbated in systems powered by variable power levels can limit their computation capabilities. Multipliers constitute a major component of these applications with complex logic design and a large gate count compared to other arithmetic units. As such, there is significant interest in designing new approaches to low-complexity multipliers.

Recently, approximate arithmetic, in particular approximate adders and multipliers, have shown notable advantages to benefit from a wide spectrum of naturally imprecise-tolerant applications, such as image processing, pattern recognition, and machine learning (ML). The concept of approximate arithmetic involves replacing system components of normal degrees of complexity with less complex components, which may provide reduced accuracy. Compared to the adder, the multiplier is a crucial component of these applications with complex logic design and a large gate count.

This thesis investigates the possibility and profitability to trade accuracy for energy at run-time by using configurable approximate arithmetic hardware. In the first approach, a configurable adaptive approximation method for multiplication is proposed. The extra overheads associated with in the configuration circuits prove to be negligible compared to the multiplier's costs. Central to the proposed approach is a significance-driven logic compression (SDLC) multiplier architecture that can dynamically adjust the level of approximation depending on the run-time power/accuracy constraints. The architecture can be configured to operate in the exact mode (no approximation) or in progressively higher approximation modes (i.e. 2 to 4-bit SDLC). In the second approach, a novel ML hardware design method centred around multiply-accumulate (MAC) units is presented. Core to the configurable MAC design is a configurable multiplier. In the third approach, a configurable modified activation function is proposed to minimize the prediction error of the configurable MAC design.

To evaluate and validate the trade-offs, the three approaches (configurable multiplier, MAC unit and modified activation function) are designed in System-Verilog and synthesized using Synopsys Design Compiler, employing a UMC 90nm digital complementary metal-oxide semiconductor (CMOS) technology as well as on Field Programmable Gate Arrays (FPGAs), and then compared with other available methods. These improvements come at the expense of errors introduced into the circuit and investigated. The efficacy of the first approach (configurable multiplier) technique is evaluated with a real life image processing application, which consists of additions and multiplications using the proposed three multiplier configurations (Exact, 2- and 4-bit SDLC). The analysis considers the Gaussian blur filter since it is widely used in image processing application, typically to reduce image noise and artifacts by acting as a low-pass filter.

Additionally, the second and third approaches are evaluated as the key processing blocks in a multi-layer perceptron (MLP) network in order to validate the dynamic tunability between accuracy and power consumption. As case studies, the MLP is trained using well-known machine learning (ML) datasets. The configurable multiplier design (first approach) can be suitably used for energy-efficient multiplier designs, where quality requirements can be relaxed. The second and the third approaches (configurable MAC unit and activation function) can also be used within the power-adaptive neuron modules with a minimal loss in output quality compared to those used in previous studies.

Publications

Journal and magazines publications:

- **Ibrahim Haddadi**; Rishad Shafik, Fei Xia, Alex Yakovlev, *Run-time Configurable Approximate Multiplier Design for Neural Network Applications*, (in review), submitted to MDPI :Computer Science and Engineering SI: Reconfigurable Computing – Hardware/Software Co-design.

This paper presents the design and analysis of the proposed run-time configurable adaptive MAC unit and implement the proposed MAC architecture in an ANN and validate it by exercising different data-sets to evaluate the performance-energy-quality trade-offs. This work appears in Chapter 4.

- **Ibrahim Haddadi**; Rishad Shafik, Fei Xia, Alex Yakovlev, *Neural Network Design with a Configurable Modified Activation Function*, (to be submitted).

This paper presents the design and analysis of a new configurable modified activation function, which supports run-time configuration multiplier. This work appears in Chapter 5.

Conference publications:

- **Ibrahim Haddadi**; Issa Qiqieh, Rishad Shafik, Fei Xia, Muhammad Al-Hayanni and Alex Yakovlev, *Run-time Configurable Approximate Multiplier using Significance-Driven Logic Compression*, Accepted for publication in Proceedings of the 39th IEEE International Conference on Computer Design (ICCD-2021), October 24-27. 2021.

This paper presents the design and analysis of the proposed run-time configurable adaptive approximation method for multiplication. This work appears in Chapter3.

- Adrian Wheeldon, Rishad Shafik, Alex Yakovlev, Jonathan Edwards, **Ibrahim Haddadi** and Ole-Christoffer Granmo, Tsetlin Machine: A New Paradigm for Pervasive AI, SCONA Workshop at Design, Automation and Test in Europe, DATE, 2020.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Energy-Efficient Computing	2
1.3	Approximate Computing	3
1.4	Multipliers in Applications	4
1.5	MAC Unit	5
1.6	Neural Networks	7
1.7	Thesis Hypotheses and Questions	8
1.8	Thesis Scope and Contributions	8
1.9	Thesis Overview	9
2	Background	11
2.1	Approximate Computing	12
2.2	Approximate Circuit Design	13
2.2.1	Imprecise Hardware Design	13
2.2.2	Non-Boolean Circuits	14
2.2.3	The Fundamental Principle of the Approximate Circuits	15
2.2.4	Exploiting Imprecision-Resilience	17
2.2.5	Taxonomy of Approximate Circuits	17
2.2.6	Significance-Driven Logic Compression (SDLC)	19
2.3	Artificial Neural Networks	21
2.3.1	A Biological Neuron	21
2.3.2	Models of a Neuron	21
2.3.3	Network Architectures	22
2.4	Learning Processes	27
2.5	Learning Algorithms	28
2.5.1	Feed-forward Operations	29
2.5.2	Activation Functions	29
2.5.3	Back-propagation Algorithm	30

3	Configurable Approximate Multiplier	32
3.1	Introduction	33
3.1.1	Organization of the Chapter	33
3.2	Existing SDLC Method and Motivation	34
3.3	Proposed Configurable Approximation Hardware	36
3.3.1	Configurable Multiplier Architecture	36
3.3.2	Hardware Knobs for Run-Time Configuration	38
3.4	Error Analysis	42
3.5	Comparative Evaluations	44
3.5.1	Area, Delay & Power Trade-offs in ASIC Implementations	44
3.5.2	Area, Delay & Power Trade-offs in FPGA Implementations	46
3.6	Case Studies	46
3.6.1	Energy-Aware Configuration Algorithm (EACA)	47
3.6.2	Gaussian Blur Filter	48
3.6.3	Energy-Aware Approximation	50
3.7	Conclusions	51
	4 Neural Network Design with Run-time Configurable Approximate MAC	53
4.1	Introduction	54
4.1.1	Contributions	54
4.1.2	Chapter Organisation	55
4.2	SDLC Method and Motivation	55
4.3	Run-time Configurable Approximate Neuron Design	56
4.3.1	Configurable Neuron Architecture	56
4.3.2	MAC unit reconfigurable circuits	58
4.4	Implementation of ANNs	59
4.4.1	Area Reduction & Inference Accuracy in FPGA Implemen- tations	64
4.5	Case Study	64
4.6	Conclusion	66
5	NN Design with Modified Activation Function	67
5.1	Introduction	68
5.1.1	Contributions	68
5.2	Methods of Activation Functions	69
5.3	Proposed Adaptive Approximation Method	69

5.4	Proposed Configurable Modified Activation Function	73
5.5	Experimental Results	75
5.5.1	Area, Delay & Power Trade-offs in ASIC Implementations . .	75
5.5.2	Area, Delay & Power Trade-offs in FPGA Implementations . .	77
5.5.3	Neuron Module Learning	77
5.6	Case Studies	80
5.6.1	Data Classification Results using MLP	80
5.6.2	Inference Accuracy Results for Different ANN Hardware Configurations with Different Data-sets and Activation Func- tions	84
5.6.3	Modified Activation Function Area Overhead in ASIC Im- plementations	87
5.6.4	Energy-Aware Variation Scenarios	87
5.6.5	Configurable Neuron Architecture Scenarios	89
5.7	Conclusions	92
6	Conclusions and Future Work	94
6.1	Summary	95
6.2	Critical Review and Future Work	96

List of Figures

1.1	Global internet users [6].	3
1.2	Basic design of MAC unit	6
1.3	Fully parallel feed-forward deep neural network architecture[45].	7
2.1	Error-Resilient Paradigms [54].	12
2.2	Taxonomy of imprecise computation in hardware [55].	13
2.3	Example of a (2×2) multiplier: (a) exact, and (b) approximate [37].	16
2.4	: Taxonomy of approximate circuits [75].	18
2.5	Three different sizes of logic clusters are used to compress partial products based on their progressive bit-significance in an (8 x 8) multiplier : Dot diagram showing the impact of increasing the depth of the logic clusters in the case of (8×8) multiplier: (a) clustering a group of bits within 2 successive rows in the partial product bit-matrix after bitwise multiplication; (b) generating a reduced set of product terms after targeting the depth of 2-row logic compression; (c) ordered matrix after applying commutative remapping of the bit sequence resulting from the SDLC approach; (d), (e) and (f) the same process when applying 3-bit logic compression; (g), (h) and (i) the same process when applying 4-bit logic compression. The dotted rectangles at the right indicate the heights of the critical columns which are further reduced compared to the accurate accumulation tree [25].	20
2.6	A single neuron structure [84].	22
2.7	Neural networks Types.	23
2.8	Feedforward network with a single layer of neurons [85].	23
2.9	Fully connected feedforward network with one hidden layer and one output layer [86].	24
2.10	Architectural graph of a MLP with four hidden layers [87].	25

2.11	Recurrent network with no self-feedback loops and no hidden neurons [88].	26
2.12	Supervised learning [96].	27
2.13	Reinforcement Learning [95].	28
3.1	Two different sizes of logic clusters are used to compress partial products based on their progressive bit-significance in an (8 x 8) parallel multiplier architecture. (a) clustering a group of bits within two successive rows in the partial product bit-matrix after bitwise multiplication; (b) generating a reduced set of product terms after targeting the depth of 2-row logic compression; (c) ordered matrix after applying commutative remapping of the bit sequence resulting from the SDLC approach; (d), (e), and (f) the same process when applying 4-bit depth of logic clusters. The d-bit indicates the depth of logic cluster.	35
3.2	Process chart explaining the difference between the main stages in (a) exact multiplication and (b) the proposed multiplication using the SDLC approach.	37
3.3	The reduction stages of an (8 × 8) Wallace tree multiplication illustrate the accumulation method for the PPM formed from exact and two different sizes of logic clusters (shown in Figure 3.1) (a) four reduction stages are required in the case of the (8 × 8) traditional Wallace tree multiplier(WTM); (b) two reduction stages are required by means of the Wallace accumulation method to reduce the PPM generated by the 2-bit SDLC(see Figure3.1 (c); (c) no reduction stages for the 4-bit SDLC as the height of the PPM is only two rows(see Figure 3.1 (f)), and (d) configurable (8 × 8) Wallace tree multiplication includes the common similarities and variations shown in (a), (b) and (c).	41
3.4	Diagrammatic sketch of the proposed hardware architecture of the configurable (8 × 8) multiplier with exact, 2-bit and 4-bit SDLC modes.	42
3.5	Flowchart diagram showing the main steps for evaluating the proposed design using Synopsys Design Compile.	45
3.6	Flowchart diagram showing the main steps for evaluating the impact of the proposed multiplier on the final quality of image processed by Gaussian blur filter.	49

3.7	Output quality and energy consumption for Gaussian blur filtering using the three different configurations of the proposed (8×8) multiplier.	50
3.8	Different scenarios for the EACA model operate at run-time under highly variable energy conditions while sustaining execution.	51
4.1	Neuron structure-serial processing including the proposed hardware architecture of the configurable MAC unit.	57
4.2	Representation of the reconfigurable circuits supporting three different modes.	58
4.3	Main steps for training and testing a Neural Network.	60
4.4	Back-propagation learning rule results for various selections of ANN hardware configurations using Noisy XOR and Binary IRIS data-sets.	61
4.5	Inference accuracy results for various selections of ANN hardware architectures using various data-sets.	62
4.6	Different run-time scenarios for the EACA model to operate under highly variable energy conditions. (a) only <i>exact</i> configuration allowed and (b) all configurations allowed.	65
5.1	The reduction stages of an (8×8) Wallace tree multiplication illustrate the accumulation method for the PPM formed from the exact and logic clusters of two different sizes (a) four reduction stages are required in case of (8×8) traditional Wallace tree multiplier(WTM); (b) no reduction stages for the 4-bit SDLC as the height of the PPM is only two rows, and (c) configurable (8×8) Wallace tree multiplication.	71
5.2	Neuron (structure-serial) processing including the proposed hardware architecture of the configurable MAC unit.	72
5.3	Simulation process of error metrics calculation.	75
5.4	A schematic representation of a single neuron architecture with the activation function is shown (a) and the proposed modified activation function is shown in (b).	76
5.5	Flowchart diagram demonstrating the main steps for evaluating the impact of the proposed MAC unit with modified activation function and on a perception based Classifier.	79

5.6	The test set perceptron classification using; (a) exact multiplier and exact activation function ; (b) 4-bit SDLC multiplier in [102] and exact activation function; (c) proposed 4-bit SDLC multiplier and a configurable activation function (blue and red points represent two classes 0 and 1, black dots mismatch classification points and the axes show the random inputs between 0 to 255 for 8x8 multiplier . . .	79
5.7	Backpropagation learning rule results for various selections of ANN hardware configurations using different MAC configurations with the modified Sigmoid activation function in several well-known data-sets.	81
5.8	Backpropagation learning rule results for various selections of ANN hardware configurations using different MAC configurations with the modified ReLU activation function in several well-known data-sets.	82
5.9	Inference accuracy results for various selections of ANN hardware architectures using various data-sets with (a)- Sigmoid and (b)- ReLU activation functions.	84
5.10	Backpropagation learning rule results for various selections of ANN hardware configurations using different MAC configurations with the ReLU activation function in MNIST data-set. Different runtime scenarios for the EACA model to operate under highly variable energy conditions. (a) Only Exact configuration allowed. (b) Approximate (4-bit SDLC) configurations allowed (c) All configurations allowed.	88
5.11	Backpropagation learning rule results for various selections of ANN hardware configurations using different MAC configurations with fixed and modified ReLU activation function in the MNIST data-set.	92

List of Tables

1.1	Summary of approximate multiplier design approaches [32].	5
2.1	Truth table for the exact and approximate (2×2) multipliers used to obtain comparative error analysis in Figure 2.3, with changed the entry highlighted [37].	16
2.2	List of Abbreviations [100].	31
3.1	The number of full and half adders used by the accumulation stages and CPA for different exact, approximate multipliers and the Proposed configurable (8×8) Wallace tree.	38
3.2	MRED and NMED for different approximate configurations in an (8×8) configurable multiplier.	43
3.3	comparing existing multiplier designs and the proposed configurable design in terms of power(P), area (A) delay(DL) and Power-delay product (PDP).	45
3.4	Comparing non-functional metrics with Kumar et al [42].	46
4.1	Numbers of input, layers and required neurons/MACs for different data-sets.	59
4.2	Area (A), power(P), delay(DL) and Power-delay product (PDP) results for different MAC configurations.	63
4.3	Comparing with Ullah, S et al [148] in terms of Area reduction and Inference accuracy.	64
5.1	The number of full and half adders used by the accumulation stages and CPA for different exact, approximate multiplier and the Proposed configurable (8×8) Wallace tree.	72
5.2	Normalised Mean Error Distance (NMED) for (8×8) multiplier size with a different d-bit.	73

5.3	Error Distance (ED), Error Rate (ER), Mean Relative Error Distance (MRED), Normalised Mean Error Distance (NMED), are different metrics used to evaluate a multiplier.	74
5.4	comparing existing multiplier designs and the proposed configurable design in terms of power(P), area (A) delay(DL) and Power-delay product (PDP).	77
5.5	Comparing non-functional metrics with other approaches.	77
5.6	ER results for different designs (a),(b) and (c).	80
5.7	Numbers of input, layers and required neurons for different data-sets.	83
5.8	Area overhead results for the modified Sigmoid activation function ($Modified_{AF}$) including the additional number of adders in hidden layers (h) and output layer (y) compared to the fixed ($Fixed_{AF}$) activation function.	86
5.9	Area (A), Power(P), Delay(DL), Power-delay product (PDP), Inference Accuracy (IA) and Area overhead(Aoverh) results for different MAC configurations using fixed/modified ReLU activation function in the hidden layers (h) and output layer (y).	91

Acronyms

AI artificial intelligence.

ANN artificial neural network.

ASIC application-specific integrated circuit.

CLA carry look-ahead adder.

CMOS complementary metal-oxide-semiconductor.

CNN convolutional neural network.

CPA carry-propagate addition.

DNN deep neural networks.

EACA energy-aware configuration algorithm.

ED error distanc.

FPGA field programmable gate array.

MAC multiply–accumulate.

ML machine learning.

MLP multi-layer perceptron.

MNIST modified national institute of standards and technology.

MRBM modified radix2 booth multiplier.

PCMOS probabilistic complementary metal-oxide-semiconductor.

PPM partial product matrix.

PSNR peak signal to noise ratio.

QoR quality of the result.

RED relative error distance.

RNN recurrent neural network.

RTL register-transfer level.

SC stochastic computing.

SDLC significance-driven logic compression.

SOC system-on-chip.

VOS voltage over-scaling.

WTM wallace tree multiplier.

Chapter 1

Introduction

In a wide range of computing applications, an emerging design paradigm called approximate computing has recently evolved to exploit imprecision resilience and achieve further improvements. This chapter presents the motivation and defines the key concepts of the research reported in this thesis. It emphasises the importance for approximate computing in the domain of arithmetic multiplier design to improve energy and performance efficiency. Subsequently, the primary contributions of this research, together with how the thesis is arranged is discussed.

1.1 Motivation

Designing high-performance and energy-efficient hardware still has difficulties due to the complexity of arithmetic. The issue is even worse in systems powered by variable power level since changing power levels can restrict their ability to perform computations. In the past, researchers have worked on low-level modifications that focus on common arithmetic circuits and have investigated and designed circuits such as multipliers [1], [2] or made application-specific modifications [3]. However, these techniques have not been implemented in systems that are designed to operate under varying power/energy conditions, such as those used to harvest power from ambient sources that may exhibit variations in spatial and temporal dimensions, by two or more orders of magnitude [4]. This presents new challenges in operating devices powered by such energy sources, specifically those with varying power envelopes. Despite this, the device must make forward progress with its computations and deliver sufficient computational capacity under extreme power conditions [5]. To meet the requirements of these applications, the current study investigates the factors that should be considered in designing these types of new devices, here by using the advantages of approximate computing, ML techniques and a system to allocate an accelerator and optimise hardware according to power constraints. The motivation of this work is introduced as follows.

1.2 Energy-Efficient Computing

The ever-evolving digital world's present processing capability cannot keep up with the demands for computational performance brought on by the large amounts of global data. Furthermore, there are already more than 4.95 billion internet users worldwide (as of January, 2022) [6]. Mobile devices were used by the vast majority of users to access their preferred platforms see Figure 1.1. Additionally, the exponential rise in the production of digital data worldwide [7] is linked to the evolution of the global electricity demand for data centres [8]. This makes it possible to discover substitute computer systems that offer the required processing performance while consuming considerably less energy. The exponential scaling of integrated circuits and many-core system-on-chip (SOC) technologies have helped to meet the demand for better computational performance over the past three decades [9]. However, the rapid development of these technologies coincides with reports that Moore's Law, which states that the number of transistors on a microchip doubles every two years, no longer applies [10]. As a result, technology scaling won't likely be a driv-

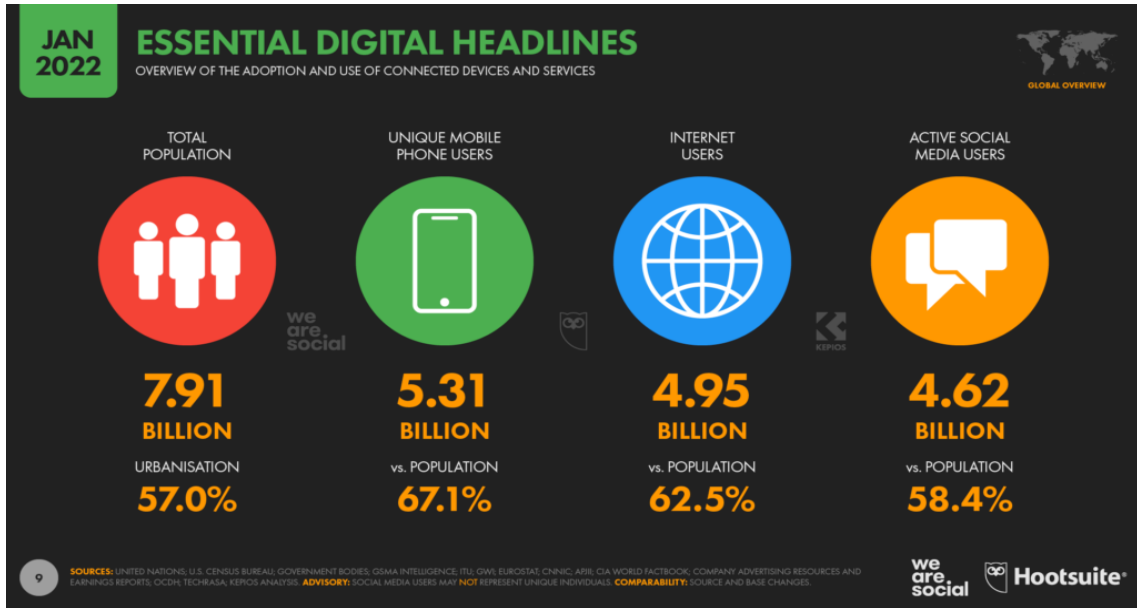


Figure 1.1: Global internet users [6].

ing force behind computing in the near future [11]–[14] as the downscaling of the complementary metal-oxide semiconductor (CMOS) is entirely strained to its limits. Furthermore, at different abstraction levels, the per-transistor performance power efficiency is not keeping up with established power-reduction strategies [15]–[17]. With time, this leads to the so-called Dark Silicon era [18], in which it could only be possible to turn on a small portion of the on-chip computer resources in order to maintain safe thermal and power density restrictions. In order to increase functionality of computing platforms across the board, from servers and data centres to mobile and deeply embedded devices, there is an actual need to investigate new computing paradigms. Examples of existing energy efficiency design methods include precision scaling [19], approximate logic designs [5] and new analog or mixed-signal circuit designs [20]. The approximate computing paradigm is a viable strategy for achieving energy efficient [2], [21]–[25]. The following section provides a thorough introduction to approximation computing and multiplier design.

1.3 Approximate Computing

There is a constant need for increased computing performance at reduced energy cost. It is unlikely that improvements from manufacturing processes alone, such as technology nodes, voltage scaling or many-core system-on-chip, will be able to cope with this challenge. Thus, there is a genuine need to develop disruptive design

approaches to achieve transformational energy reductions. Approximate computing systems design is a promising approach to this end [2], [23].

The basic idea of approximate arithmetic, which is approximate computing, involves replacing system components operating at normal levels of complexity with less complex components in order to achieve a number of goals such as lower power consumption, high performance, accuracy and a very low area. However, in terms of accuracy, error-resilience can be defined as a technique used to deliver acceptable results. Moreover, to reach this point, two factors need to be focused on, namely perceptual resilience in relation to error and algorithmic resilience [22], [24].

Over the years, a number of approximate computing approaches have been proposed. These approaches aim to reduce the complexity of the circuits and systems in terms of their computation latency and energy consumption. Approximations can be introduced at all design levels, starting from the circuit [26] via the logic [27] and the design [28] to programming language [29] and algorithms [30], [31]. The common design techniques in approximate circuit designs including functional, timing (voltage over-scaling (VOS) and over-clocking) approximations and systematic design methodologies with a summary of the related work, are described in further detail in the next chapter.

1.4 Multipliers in Applications

In many applications, multipliers are the most significant part due to two main reasons. The first reason is that multipliers in modern types of microprocessors are identified as having comprising complex logic design owing to their function unit for data processing. The second reason is that computation-intensive applications require a massive multiplication operation in order to generate results. Furthermore, these two factors have led to a focus on approximate fixed-point multipliers and research into their design, due to its impact on overall system power consumption as well as performance [32].

Table 1.1: Summary of approximate multiplier design approaches [32].

References	Scheme	Limitations
[33], [34]	Aggressive voltage scaling: reduce the current value of the input voltage underneath its nominal value.	Unexpected errors generated by time, generally impacting the most significant bit.
[35], [36]	Truncation: elimination from the least significant columns of partial products.	The resulting errors are increased as more columns are eliminated.
[37], [38]	Modular re-design: inaccurate small blocks of multipliers.	The critical path may not be reduced by this technique.

[32] sums up aspects of as well as the limitations of studies in the area of approximate computing listed in Table 1.1. [33], [34] worked on timing behaviour and how to adjust it by using specific voltage scaling techniques. Moreover, working lower than minimal required voltage to decrease the required energy consumption will affect with the loss of accuracy because of the number of the errors that will appear. While, [35], [36] focus on reducing the area and energy by using functional modifications which can handle logic reduction techniques using Boolean equivalence.

As a consequence of reducing the number of columns, energy reduction is accomplished. Conversely, error levels will rise. Likewise, modules can be redesigned to use low complexity combinational logic, which is known as different effective method [37], [38]. Using this technique can help to design multipliers with greater energy efficiency by using small approximate blocks. As a result of using this technique, the number of errors will raise the cost of increasing the size of the multiplier. In one proposed design of an energy-efficient approximate multiplier, an approach was used called Significance-Driven Logic Compression (SDLC) [25]. This is an algorithm that produces partial product rows depending on their bit significance [25]. To minimize the number of product rows, the algorithm is followed by combative remapping.

1.5 MAC Unit

multiply-accumulate (MAC) unit is normally made of three parts which are a multiplier, adder and accumulator (pipeline registers). So, an n-bit type of MAC consists of an 8-bit multiplier, a 17-bit adder, and an 18-bit accumulator. Figure

1.2 shows the design of the normal MAC unit [39].

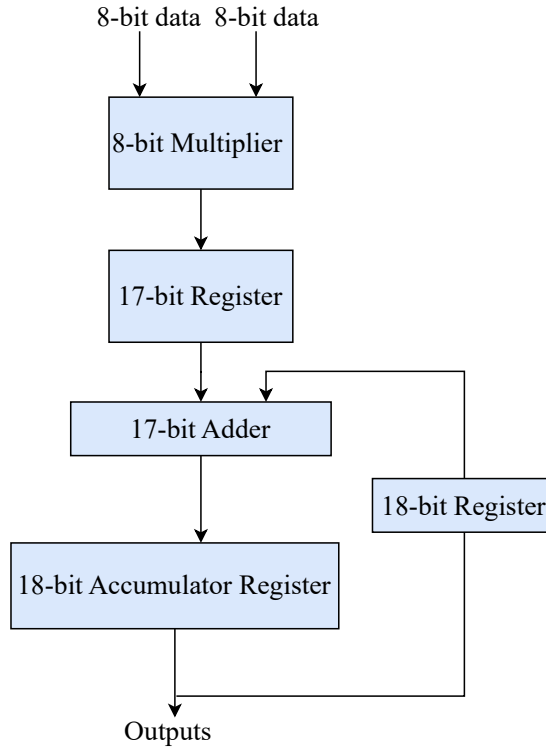


Figure 1.2: Basic design of MAC unit [39].

Many researchers have been working on the design of the MAC unit, notably on the architectures required to design advanced MAC unit modules that optimize for power, area and timing parameters. A considerable amount of literature is available on designing MAC units due to their significance in different applications. For example, [40], [41] and [42] target exact designs without considering approximation for power reduction. Existing work on an approximate MAC unit is limited compared to other functional units. For example, Dutt et al. [43] proposed an approximate radix-2 hybrid redundant MAC unit based on a redundant number system. The proposed design exploits an approximate hybrid redundant adder as the basic building block for both addition and multiplication operations in the MAC unit. However, a significant energy gain mandates approximating 40 out of the 64 bits in relation to the results, which significantly degrades the output quality significantly. An approximate MAC unit based on partial products compression and elimination, was proposed in [44].

The parallel information processing structure known as an artificial neural net-

work (ANN) comprises of processing units. Therefore, it is necessary to build an effective processing unit that also offers greater performance in the computation time. The MAC unit (multiplication and accumulation) and the activation unit make up the processing unit. The following section describes the usage of the MAC unit as arithmetic operations of the modular electronic neurons.

1.6 Neural Networks

An artificial neural network (ANN) is characterised by a large number of simple processing neurons like processing elements or nodes, which are the fundamental components of an ANN [46]. Two different processing elements in the networks are processing units and topology. Processing units are typically, MAC and an activation unit [47]. Established on the processing unit the performance of the network is increased. Two topologies are in the ANNs, one comprises feed forward networks, whilst the other is feedback or recurrent networks. Feed forward network consists of single layer, multilayer perception and radial basis function. Recently, the research on artificial neural network algorithms and ML has increased to the level of human beings in specific fields. In particular, researches pertaining to hardware accelerators [48], [49] such as deep neural networks (DNN) have been extensively adopted in many computer vision areas such as image processing, medical imaging, activity recognition and robotics. This is because of the performance of the parallel process calculators that have been improved to handle a large number of computations and

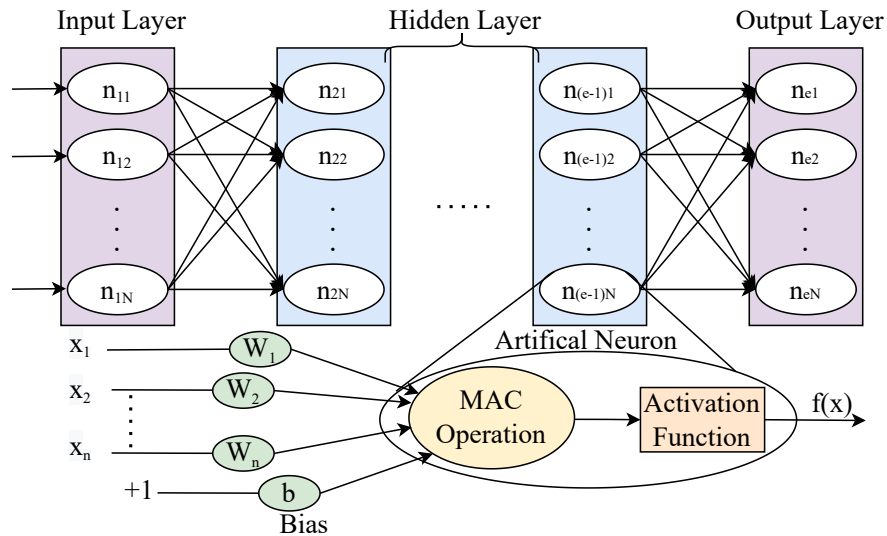


Figure 1.3: Fully parallel feed-forward deep neural network architecture[45].

weights for ANN [50]. Typically, DNNs conduct the challenging multiplication to update synaptic weights using high-precision values, for example 24-bit, 32-bit or even 64-bit. The key factor requiring the extremely complex multiply-accumulate (MAC) circuits with significant energy consumption is the high-precision multiplication in DNNs [51](see Figure 1.3). Neural networks (NNs), often referred to as "connectionist models" or "concurrent connections," have witnessed initial interest. Indeed, the study of connectionism, another name for NN, has come back into fashion in recent years, and there is now a substantial amount of global research being conducted in this field. Many facets of computing are expected to undergo a renaissance thanks to NN technology. The phrase "neural networks" conjures up countless vivid images. It implies devices that resemble neurons and might be loaded with overtones of science fantasy associated with the Frankenstein legend. In brief, a NN is a linked collection of discrete processing "nodes" or cells, whose operation is somewhat analogous to that of a biological neuron.

1.7 Thesis Hypotheses and Questions

Hypotheses are in the context of configurable approximate arithmetic hardware, must address the following points:

- (a)- Is it possible and profitable to trade accuracy for energy at run-time by using configurable approximate multiplier design ?
- (b)- Can the target design work against the power consumption of the error correction (training) ?
- (c)- Can every approximate neuron (includes approximate multiplier and activation function) be used functionally the same as or approximately the same as exact neuron (includes exact multiplier and activation function) ?

1.8 Thesis Scope and Contributions

This thesis attempts to address the above fundamental questions by exploring promising research directions in the development of energy-efficient configurable approximate arithmetic hardware. It presents novel design approaches for improving energy and performance efficiency of configurable approximate multiplier with variable-accuracy implementations. From the perspective of levels of abstraction, the research work in this thesis focuses on modifying the behavioural description of the

multiplier design from gate level and register-transfer level (RTL). The aim is to develop a run-time configurable adaptive approximation method for multiplication that is capable of managing the energy and performance trade-offs. Using this architecture (configurable multiplier), a configurable-approximation MAC unit is implemented that is the fundamental arithmetic component in Neural Networks based ML systems. Additionally, a configurable modified activation function was proposed to minimize the prediction error of the configurable-approximation MAC unit. Points below are emphasize the thesis contribution's architecture as follows:

- propose a new multiplier architecture using variable approximation, which supports run-time configuration (Chapter 3, Thesis Questions 1.7-(a));
- propose a new MAC design using variable approximation, which supports run-time configuration by implementing the proposed MAC design in an ANN and validating it by way of applying different data-sets to evaluate the performance-energy-quality trade-off (Chapter 4, Thesis Questions 1.7-(b));
- propose a new neuron design with configurable modified activation function (Chapter 5, Thesis Questions 1.7-(c)).

1.9 Thesis Overview

This thesis is organized into six chapters. The following is a summary of the work carried out in the thesis:

Chapter 2 outlines in a logical manner widely used and recently reported methods for energy-efficient imprecise hardware. It covers approximate circuit design approaches and describes the purposes of approximate computing. A survey of the most recent approximative multiplier designs is then presented. In order to highlight their similarities and distinctions, different design methodologies are reviewed and categorised using a multi-dimensional taxonomy. Chapter 3 proposes a run-time configurable adaptive approximation method for multiplication that is capable of managing the energy and performance tradeoffs — ideally suited in these systems. Central to the approach is a significance-driven logic compression (SDLC) multiplier architecture that can dynamically adjust the level of approximation depending on the run-time power/accuracy constraints. Results compared to other available approaches are also shown. This chapter exhibits a case study of image processing application using Gaussian blur filter to evaluate the efficacy of the proposed technique with a real life application. Chapter 4 proposes a new MAC architecture taking advantage of the ability of the configurable adaptive approximate multiplier approach, which supports run-time configuration. This chapter presents the assessment of

the capabilities of the configurable-approximation MAC units as the key processing block in an artificial neural network (ANN). Additionally, the chapter provides the implementation of the proposed MAC architecture in an ANN and validates it by exercising different data-sets to evaluate the performance-energy-quality trade-offs. Chapter 5 presents a new neuron design with configurable modified activation function. A configurable modified activation function was proposed to minimize the prediction error from the approximate configuration. The methods of approximation of the activation functions are developed and modified using piecewise linear and approximation. This chapter exhibits two case studies; a machine learning application using perceptron classifier and backpropagation learning rule results for various selections of ANN hardware configurations using different MAC configurations with the Sigmoid activation function on several well-known data-sets. Chapter 6 outlines the contributions and main aspects of this thesis, demonstrating a critical analysis of the research as well as possible future directions.

Chapter 2

Background

During the last decade, there has been an increasing interest in approximate computing as one of the most promising energy-efficient computing paradigms. Approximate computing exploits the flexibility provided by inherent application resilience in hardware or software implementations for more performance and energy gains. Approximate computing research combines insights from hardware engineering, architecture, system design, programming languages and even application domains similar to image processing and ML. However, approximate computing based on software techniques has been considered out of the scope for this thesis. This chapter highlights the basic concepts behind approximate circuits to understand the motivation and the choices made in the context of this work.

2.1 Approximate Computing

This particular section will cover the design of the most important component of the MAC unit, specifically the multiplier. Additionally, this section reveals several multiplier designs using approximate computing methods which can improve the traditional design strategies concerning power reduction. The basic idea of approximate arithmetic, which is approximate computing, involves replacing system components operating at normal complexity levels with less complex components in order to achieve a number of goals, such as lower power consumption, high performance, very low area and accuracy. However, in terms of accuracy, Error-resilience can be defined as a technique used to deliver acceptable results. Moreover, to reach this point, two factors need to be focused on, namely perceptual resilience to error and algorithmic resilience [22], [52]. Figure 2.1 shows the different types of error-resilience paradigms. Moreover, stochastic computing (SC) is the second type of error-resilience paradigm known as a low-cost alternative in the field of conventional binary computing. Additionally, this type has several other advantages. For example, in terms of representing information as well as processing, it applies the characteristics of digitised probabilities. A further advantage is that arithmetic units in SC are very low in complexity.

Despite the fact that the advantages connected with SC were considered to be extremely significant in the past, there were disadvantages which meant that it was seen as impractical due to low a performance in computation as well as low accuracy [53]. According to Intel, probabilistic computing is not a new study area and working on algorithms as regard deep learning and high-performance computing can generate new and valuable knowledge. As a consequence, research during the next few years is likely to achieve significant improvements in AI systems including in hardware design; for instance, in reliability, security, serviceability and high-performance computing [54].

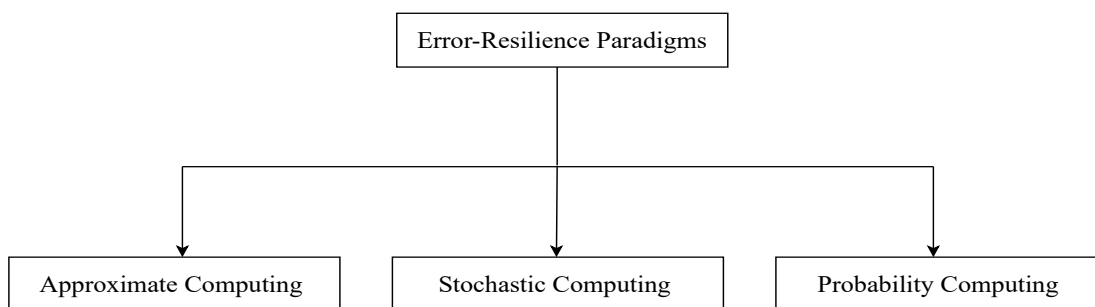


Figure 2.1: Error-Resilient Paradigms [54].

2.2 Approximate Circuit Design

In the literature, there are various research efforts devoted to addressing different designs relating to imprecise computing in hardware. In this section, the hardware that does not produce the exact output at all times by imprecise hardware is denoted. The reason for this concept is to ensure it includes a variety of existing designs that can lead to error in the output, such as in approximate, probabilistic and non-Boolean circuits. The following subsection enables the reader to understand the scope of approximate circuit design by distinguishing the work on it from related but conceptually distinct efforts in probabilistic and non-Boolean circuits.

2.2.1 Imprecise Hardware Design

This section presents imprecise hardware, including probabilistic, non-Boolean and approximate designs, to understand the area of the approximate circuits from other related efforts in the literature. Figure 2.2 illustrates a taxonomy of related terms in this area. Therefore, these terms are briefly discussed along with some explanatory instances, as follows:

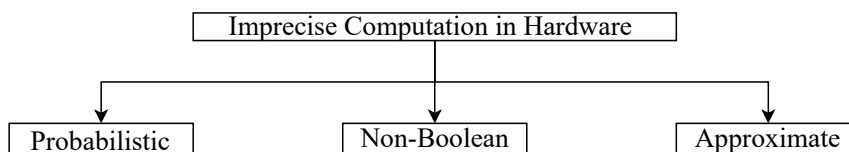


Figure 2.2: Taxonomy of imprecise computation in hardware [55].

Stochastic circuits use three computational techniques: probabilistic complementary metal-oxide-semiconductor (PCMOS), quantum, and stochastic circuits, depending on the circuit, which is inherently stochastic in operation. While PCMOS and quantum circuits and signals are stochastic, probabilistic circuits use deterministic means to represent and process probabilistic data [55], [56]. These computer techniques are briefly described below.

- **Probabilistic CMOS**

Probabilistic CMOS is a particular form of complementary metal-oxide-semiconductor (CMOS) termed PCMOS that was invented in anticipation of competing with current low power CMOS technology. This is motivated by hardware-specific "bug" behaviour when to smaller technology nodes. This is because it is susceptible to process variability and noise. Consider that CMOS devices have an exponential relationship between accuracy probability (P) and switching energy

(E). To this end, PCMOS devices can use noise as a resource to achieve low power, high-performance computations [14], [57], [58].

- **Quantum Circuits**

Quantum circuits show different kinds of intrinsically probabilistic calculations that perform efficient algorithms for stubborn problems such as classical computer science [59], [60]. B. Fast factorization of numbers [61], [62], such as fast number factorization [63]. According to the laws of quantum physics, quantum circuits are in multiple states and offer tremendous processing power due to their ability to perform one possible sequential task simultaneously and quantum computing data processes in the form of quanta.

- **Stochastic Circuits**

Stochastic circuits are a class of arithmetic circuits in which numbers are represented by random values in a series "stream" or a "bundle" of parallel [64]. The main attraction of stochastic hardware is the ability to perform complex operations with simple circuits. However, there are certain drawbacks to such calculations. For example, meeting higher accuracy requirements exponentially increases the computational time of a stochastic circuit. This is especially true if the correlation between the signals in the circuit is not guaranteed to be low enough [65].

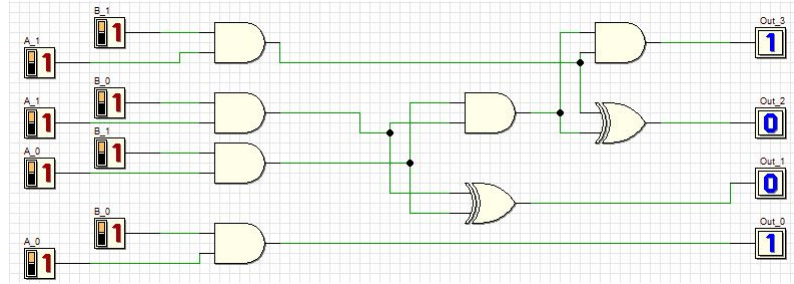
2.2.2 Non-Boolean Circuits

Non-Boolean computing refers to the class of computers that use analogue devices to store/process data. For example, the bit is the truth value (0 or 1) [66]–[68]. Non-Boolean circuits can exceed these definitions. This has been proven in various arithmetic tasks. For instance, the simulation of biological systems using analogue transistors [69], has proven to be much more efficient than digital computing. However, similar calculations use continuous values, so the process cannot be reliably repeated with exact equivalence [70]. In addition, existing analogue computers must be manually programmed, i.e., is a complex process that is very time-consuming for large, application-specific simulations [71]. However, there is still the opportunity to investigate the potential of these technologies to perform useful non-Boolean calculations.

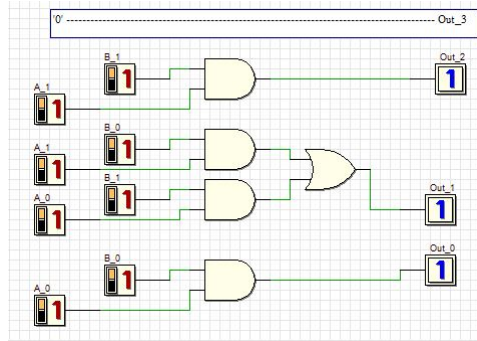
2.2.3 The Fundamental Principle of the Approximate Circuits

A class of computing known as approximate circuits uses deterministic designs to provide outputs with an acceptable level of accuracy in the pursuit of enhanced performance and energy-efficiency. However, it often utilises statistical properties to trade quality for energy/power reduction [22]. For instance, Figure 2.3 delineates the Exact and Approximate circuits for (2×2) multiplier suggested in [72]. The error statistics of the approximate circuit are displayed in the last row of Table 2.1. The primary goal is to introduce error into the multiplier by manipulating its logic function.

The fundamental principle of the above imprecise hardware designs (i.e., probabilistic, non-Boolean and approximate), is to trade minor accuracy loss in the output so as to improve performance and energy efficiency in emerging computing systems. It is feasible to address the result of (2×2) multiplication using just three bits instead of four. Consequently, the approximate circuit has an incorrect output out of the sixteen possible inputs (with a magnitude of $9 - 7 = 2$ and a probability of $1/16$ (assuming a uniform input distribution)). However, the approximate multiplier in Figure 2.3 - (b) has nearly half the area of the Exact (a), with a shorter critical path and is also less interconnected. Thus, the approximate version offers the potential for significant dynamic power reduction for the same frequency of operation, due to smaller switching capacitance. Nevertheless, a taxonomy of approximate circuits is further discussed in the following sections, presenting their potential to benefit many emerging applications using different design approaches. In the following subsection, the intrinsic attributes of imprecision-resilience in a wide range of applications are defined [37].



(a)



(b)

Figure 2.3: Example of a (2×2) multiplier: (a) exact, and (b) approximate [37].

Table 2.1: Truth table for the exact and approximate (2×2) multipliers used to obtain comparative error analysis in Figure 2.3, with changed the entry highlighted [37].

Input				Outputs		Error	
B1	B0	A1	A0	Exact	Approximate	Free	Distance
0	0	0	0	0000	0000	YES	0
0	0	0	1	0000	0000	YES	0
0	0	1	0	0000	0000	YES	0
0	0	1	1	0000	0000	YES	0
0	1	0	0	0000	0000	YES	0
0	1	0	1	0100	0100	YES	0
0	1	1	0	0010	0010	YES	0
0	1	1	1	0011	0011	YES	0
1	0	0	0	0000	0000	YES	0
1	0	0	1	0010	0010	YES	0
1	0	1	0	0100	0100	YES	0
1	0	1	1	0110	0110	YES	0
1	1	0	0	0000	0000	YES	0
1	1	0	1	0011	0011	YES	0
1	1	1	0	0110	0110	YES	0
1	1	1	1	1001	0111	NO	2

2.2.4 Exploiting Imprecision-Resilience

By eliminating the necessity for operations to be absolutely accurate or completely deterministic, approximate computing takes advantage of the inherent resilience in a wide range of hardware implementations. Thoughts on how to use faults to shorten operation times and save energy have just begun to surface. Because of measurable trade-offs and application-based error-resilience, errors are increasingly being included into hardware design as purposeful features. It is possible to define error-resilience as the ability of a system to generate usable outcomes even when job computations are performed inaccurately. A deeper understanding of inherent resilience applications is necessary for approximation computing design [22]. Utilizing perceptual restriction can enhance performance and energy efficiency (i.e., errors are not recognisable because of human perception capabilities, e.g., in multimedia applications). The functional equivalence between the specification and implementation is not necessary for these applications to function properly. These applications can be generally categorised into four groups and are frequently found in hardware circuits for mobile, embedded, and server systems [22], [59], [73]:

- Applications using analogue inputs that analyse noisy real-world data from sensors, such voice recognition, image processing,
- Multimedia, image rendering, and sound synthesis are examples of applications using analogue output that are designed for human perception,
- applications like ML and web search that lack a clear solution, and
- programmes that iteratively process large volumes of redundant data, with the number of iterations determining the quality of the result (QoR). Due to this redundancy, an algorithm can frequently be lossy and still be adequate. In the aforementioned applications, approximate circuits take use of imprecision-resilience and significantly reduce delay and energy usage. The next discussion will cover various explicit design techniques that can be used to accomplish this.

2.2.5 Taxonomy of Approximate Circuits

Applying the concept of approximate computing in hardware consists of two basic design approaches [2], [23], [74]:

- Timing-induced approximation (voltage over-scaling (VOS) and over-clocking), and

- Functional approximation.

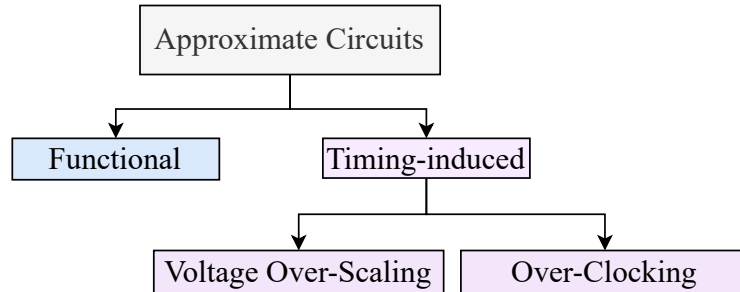


Figure 2.4: : Taxonomy of approximate circuits [75].

Figure 2.4 demonstrates a taxonomy of design approaches in approximate circuits. The voltage over-scaling [75], [76] and over-clocking [77], [78] approaches use ordinary circuits that work perfectly fine under usual circumstances.

The principal design approach can reduce power consumption by scaling the supply voltage below the expected value, which can generate the occurrence of occasional violations. Typically, in CMOS technology, the power consumption associated with the task is dominated by dynamic power dissipation $P_{dynamic}$, which is given by [79]:

$$P_{dynamic} = C_{eff} \cdot V_{dd}^2 \cdot f \quad (2.1)$$

Where (V_{dd}) is the supply voltage, (C_{eff}) is the effective switched capacitance, and f is the clock frequency. In this manner, scaling down the supply voltage leads to an overall quadratic reduction within the energy to total a task. Nevertheless, aggressive scaling of supply voltage result in timing errors while maintaining a fixed performance. The literature has shown different methods to reduce the effect of such caused errors in the systems, including adding error detection and correction circuits, which often operate within standard supply voltage [80].

Over-clocking can accomplish superior performance by enhancing the circuit’s working frequency over the highest frequency. This bring about the occurrence of timing errors but better overall performance [77], [78]. Generally, the circuit will have a maximum “stable” speed for any voltage that still operates correctly. However, while working on a constant voltage, a designer may trade the manufacturer’s safety margin by setting the device to run in the higher end of the margin.

Unlike voltage/frequency over-scaling, functional approximation does not use the original circuit but a specially designed one instead. This circuit is redesigned with-

out the fully Boolean logic implementation described in the specification. For instance, a standard method for implementing functional approximation is to omit the less significant bits of the result by removing related logic. This is expected to reduce the circuit complexity (i.e., critical path) and, therefore, achieve more performance/energy efficiency than the accurate version, with some error in the less significant bits of output. However, the approaches mentioned above used by approximate circuits are further discussed from the perspective of multiplier design given that it is the central theme of this thesis. The following subsection introduces a brief overview as regards applying the concept of approximate computing in arithmetic circuits.

2.2.6 Significance-Driven Logic Compression (SDLC)

Recently, Qiqieh *et al.* [25] proposed different levels of logic compression depending on bit significance, namely significance-driven logic compression (SDLC). Their investigation highlighted energy-accuracy trade-offs. The partial product terms are combined and compressed progressively to reduce the carry propagation chain length. Figure 2.5 shows an illustration of the approach using a dot notation in three steps. Three compression techniques are depicted using different logic cluster sizes in a 8-bit multiplier circuit. The first step aims to form a cluster of several rows in the partial product terms depending on the number of logic compression: 2-,3- or 4-bit from the groups after the multiplication operation. The second step is accumulation, which refers to the generation of a reduced partial product after applying the logic compression type. The third step subsequently applies a commutative remapping of the bit sequence resulting from the SDLC approach.

Figure 2.5 demonstrates the dotted rectangles in the third step indicating the critical column's height, which is reduced by half compared to the accurate accumulation tree in step 1. After applying these three steps, the accuracy of the results varies depending on the size of significant bits, seeing as more compression (i.e., clustering of rows) is performed for low significance bits and progressively less compression is performed for higher significance bits. By doing so, the product is generally approximate with minimum accuracy loss. However, due to shorter carry chain paths the energy is reduced drastically.

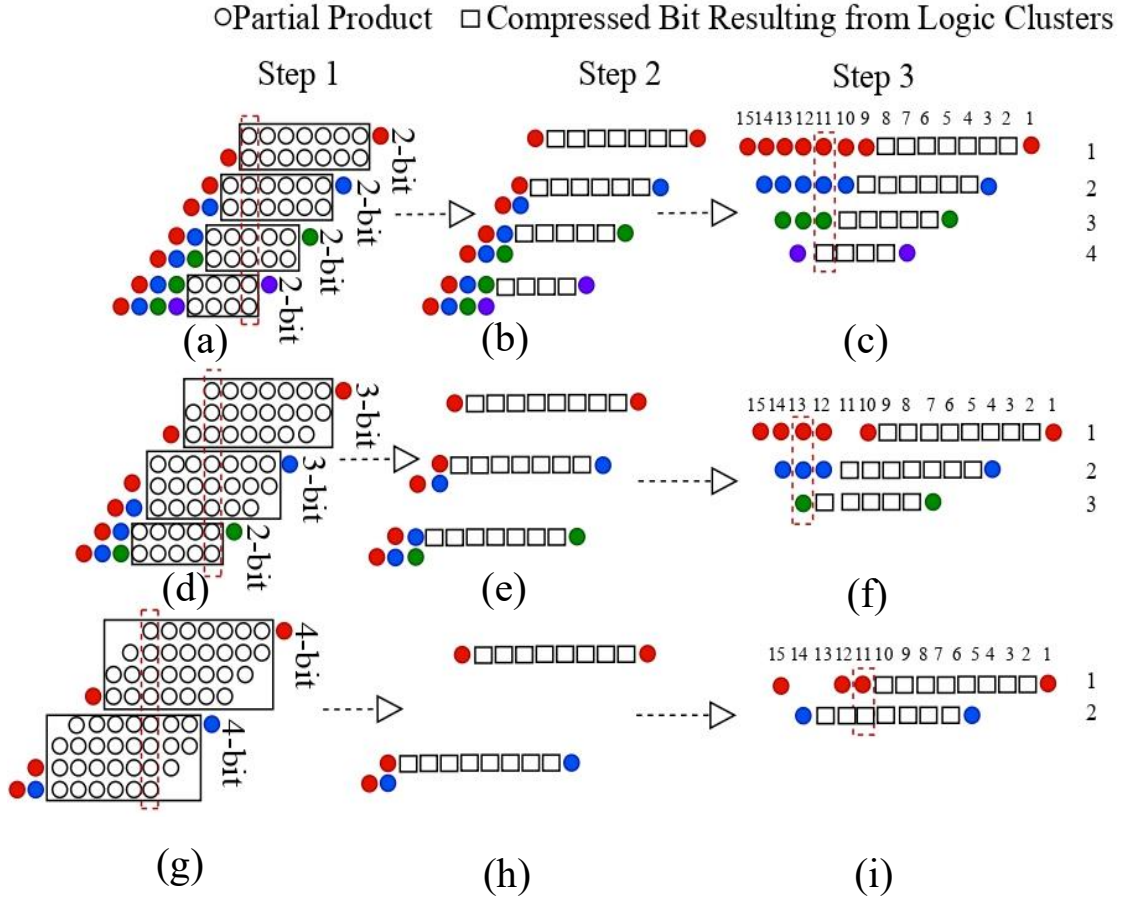


Figure 2.5: Three different sizes of logic clusters are used to compress partial products based on their progressive bit-significance in an (8×8) multiplier : Dot diagram showing the impact of increasing the depth of the logic clusters in the case of (8×8) multiplier: (a) clustering a group of bits within 2 successive rows in the partial product bit-matrix after bitwise multiplication; (b) generating a reduced set of product terms after targeting the depth of 2-row logic compression; (c) ordered matrix after applying commutative remapping of the bit sequence resulting from the SDLC approach; (d), (e) and (f) the same process when applying 3-bit logic compression; (g), (h) and (i) the same process when applying 4-bit logic compression. The dotted rectangles at the right indicate the heights of the critical columns which are further reduced compared to the accurate accumulation tree [25].

Although SDLC method provides substantial energy reductions, It lacks configurability. Put differently, the accuracy cannot be adjusted dynamically with variable compression levels. As a result, the multiplier models produced can provide only a specific range of performance, energy and power due to static design.

2.3 Artificial Neural Networks

The human brain which has inspired work on ANN computes entirely differently from the conventional digital computer. It can organise its structural constituents known as neurons to perform certain computations. For example, pattern recognition, perception and motor control are considerably quicker than the fastest digital computer in existence these days.

Consider, for instance, human vision, which is an information-processing task [81]–[83] At birth, a brain has an excellent system and the capability to develop via experience or knowledge. Specifically, the brain routinely accomplishes perceptual recognition tasks. For instance, a familiar face embedded in an unfamiliar scene can be recognised in approximately 100-200 ms, whereas for functions of much lesser complexity it may take days on a conventional computer.

Experience is created over time, with the most dramatic development, for example, the hard-wiring of the human brain during the first two years from birth, although the growth, which is termed the training stage, continues well beyond that stage. The interest in this work is primarily confined to an introductory class of neural networks that perform valuable computations through a process of learning.

A neural network is a machine designed to model how the brain performs a particular task or function of interest in its most common form. Networks are generally implemented using electronic components or simulated in software on a digital computer. To achieve good performance, neural networks use large connections of simple computational cells called "neurons" or "processing devices" to attain high performance. Therefore, the following definition of a neural network that is considered an adaptive machine can be provided. The next section introduces the components of the neural network.

2.3.1 A Biological Neuron

A biological neuron is the most basic data processing unit in the body's nervous system, the primary controlling, regulatory and communicating system. Biological neurons are made up of the following parts: Dendrites (input), Cell body, Axon (output). A biological neuron receives signals from its dendrites, processes the signal and outputs the signal from its axon based on the input signal [84].

2.3.2 Models of a Neuron

Neural networks are comprised of neurons, also known as nodes or perceptrons. Here, three basic elements of the neural model are identified:

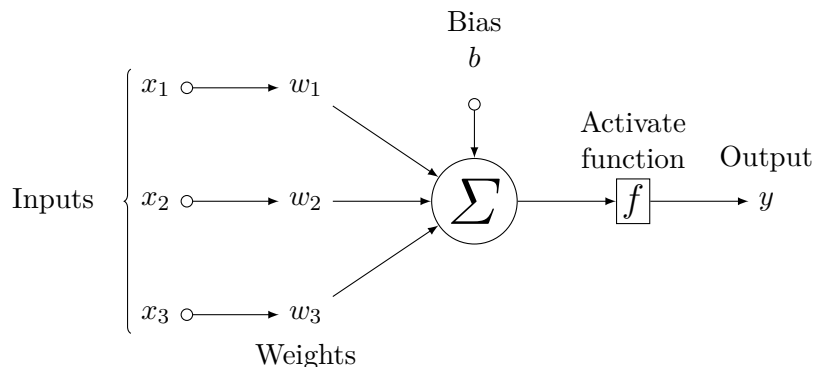


Figure 2.6: A single neuron structure [84].

- A set of synapses or connecting links, each of which is characterised by a weight or strength of its own.
- An adder for summing the input signals, weighted by the respective synapses of the neuron and
- an activation function for limiting the amplitude of the output of a neuron.

A neuron is a computational unit with one or more weighted input connections. The neurons are split between the input, hidden and output layers. A simple operation is shown in Figure 2.6, where the single neuron accepts the serial processing of weights and parallel input pairs. Each pair is multiplied together and a running total is recorded. Once all the input pairs have been processed, the final sum is passed through the activation function to produce the neuron's output.

Here, w is represents the weights, b denotes the bias, whilst x signifies the input activations of a neuron. The Equation 2.2 represents the activation function for introducing non-linearity in the network model.

$$f(x) = \sum(x_i w_j + b) \quad (2.2)$$

2.3.3 Network Architectures

Neurons in neural networks are structured tight and linked to the learning algorithms used to train the network. However, the classification of learning algorithms is considered in the next section(see section 2.3). The development of different learning algorithms is taken up in subsequent chapters of this work. This section, focus on network architectures (structures). In general, three fundamentally different classes of network architectures may be identified (see Figure2.7 :

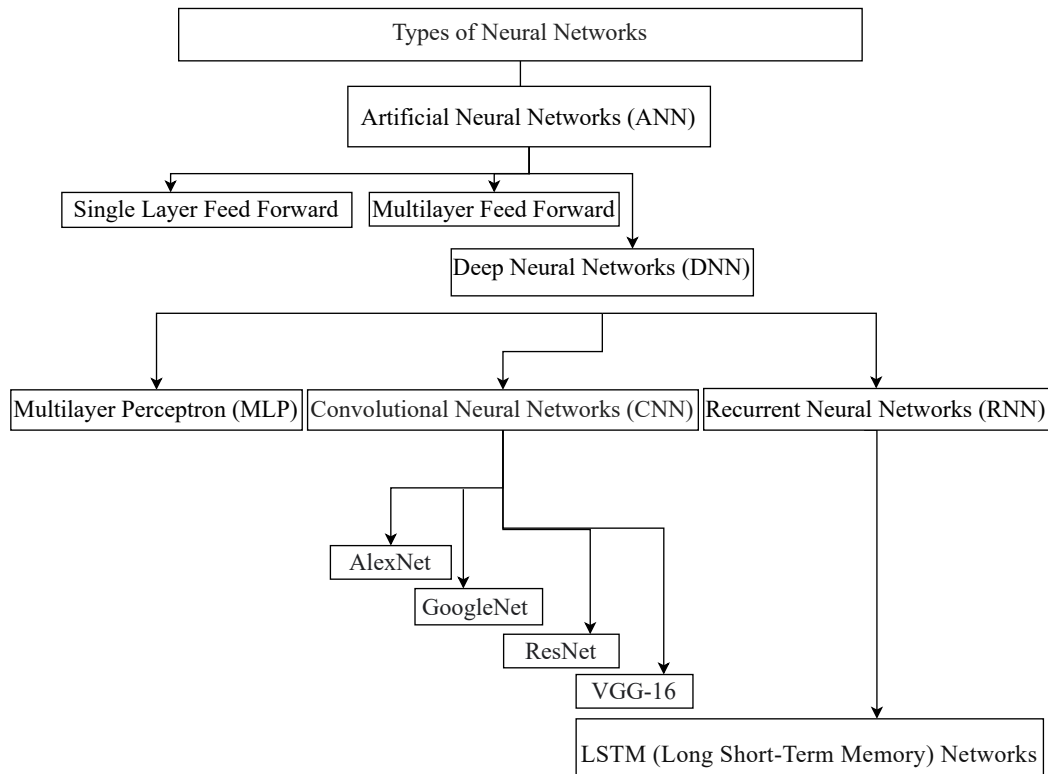


Figure 2.7: Neural networks Types.

- **Single-Layer Feedforward Networks**

In a layered neural network, neurons are organised into layers. The simplest form of a layered network is the input layer of the source node, which is projected onto the output layer (computing node) of the neuron, although not the other way round. Specifically, this network is strictly feedforward, when there are four nodes on both the input and output layers, as shown in Figure 2.8 when there are four nodes on both the input and output layers. Such networks are called

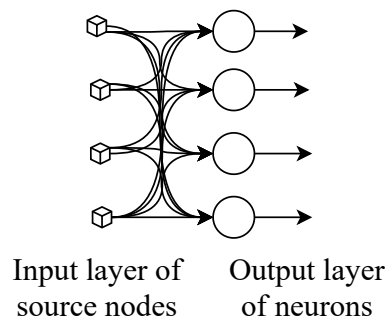


Figure 2.8: Feedforward network with a single layer of neurons [85].

single-layer networks. The term "single-layer" refers to the output layer of the computing nodes (neurons). The source node's input layer is not counted because no calculations are performed on the source node [85].

- **Multilayer Feedforward Networks**

Second-class feedforward neural networks consist of one or more hidden layers, whose compute nodes are named accordingly hidden neurons or units. The function of hidden neurons is to intervene between the external input and network output in a convenient way. Adding one or more hidden layers allows the network to extract higher-order statistics [86]. The ability of hidden neurons extracting higher-order statistics is exceptionally useful when the size of the input layer is large.

The source node of the input layer of the network provides each element of the activation pattern (input vector) that represents the input signal applied to the neurons (computational nodes) of the second layer (specifically, the first hidden layer). Second layer's output is used as input by the third layer and the rest of the network. Typically, neurons in each network layer have only the output signal from the previous layer as input. The set of output signals from neurons in the (final) output layer of the network forms the overall network response to the activation patterns provided by the source node of the (first) input layer.

The architectural graph in Figure 2.9 illustrates the layout of a multilayer feedfor-

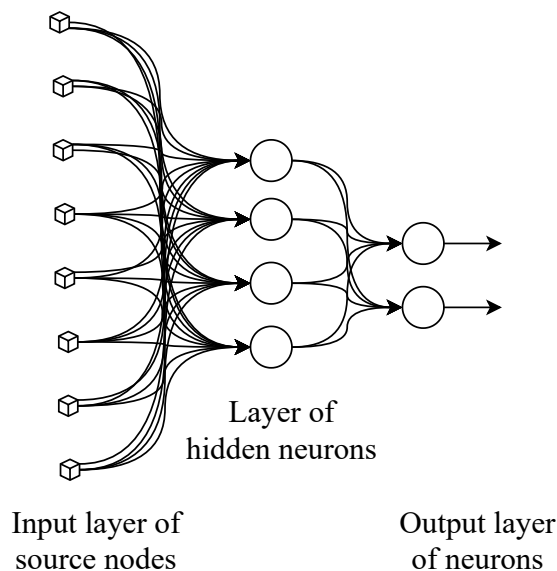


Figure 2.9: Fully connected feedforward network with one hidden layer and one output layer [86].

ward neural network for the case of a single hidden layer. For brevity, the network in Figure 2.9 because it has ten source nodes, four hidden neurons, along with two output neurons, is known as a 10-4-2 network. The neural network in Figure 2.9 is identified as fully connected because every node in each layer of the network is connected to every other node in the adjacent forward layer. However, if some of the network's communication links (synaptic connections) are absent, it is called a partly connected network.

- **Multilayer Perceptron (MLP)**

The MLP is the real example of a deep neural network. The architecture of an MLP consists of multiple hidden layers to capture more complex relationships that exist in the training data-set [87]. An illustration of an MLP is shown in Figure 2.10.

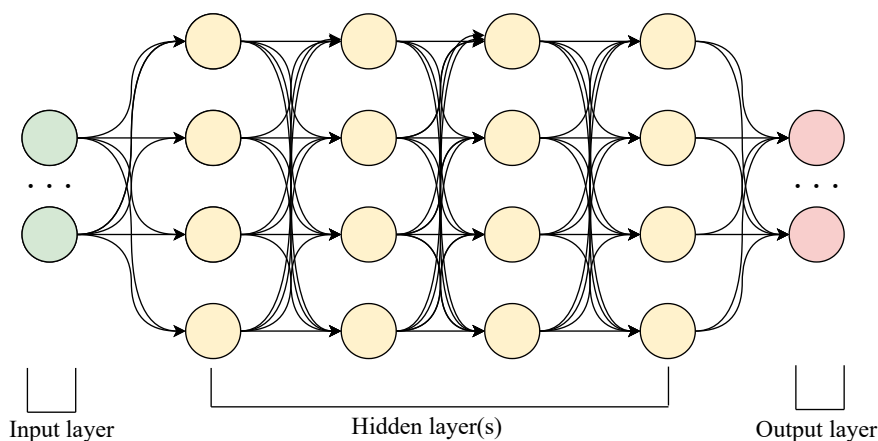


Figure 2.10: Architectural graph of a MLP with four hidden layers [87].

- **Recurrent Neural Networks**

A recurrent neural network (RNN) differentiates itself from a feedforward neural network with at least one feedback loop. For instance, a recurrent network may consist of a single layer of neurons. Each neuron feeds its output signal back to the inputs of all the other neurons, as illustrated in the architectural graph in Figure 2.11.

There is no self-feedback in the network in the structure shown in this figure. It should be mentioned that self-feedback refers to the situation where the neuron's output is fed back to its input. The recurrent network illustrated in Figure 2.11 also has no hidden neurons. The presence of feedback loops in the RNN structure regarding Figure 2.11 has a severe effect on the learning capability of the network

and its performance. Moreover, the feedback loops involve the use of particular branches composed of unit-delay elements (denoted by Z^{-1} or $Z^{-transform}$). The symbol Z^{-1} for naming a one-sample delay operator (equivalent to a memory unit).

This results in a nonlinear dynamical behaviour, assuming that the neural network contains nonlinear units [88].

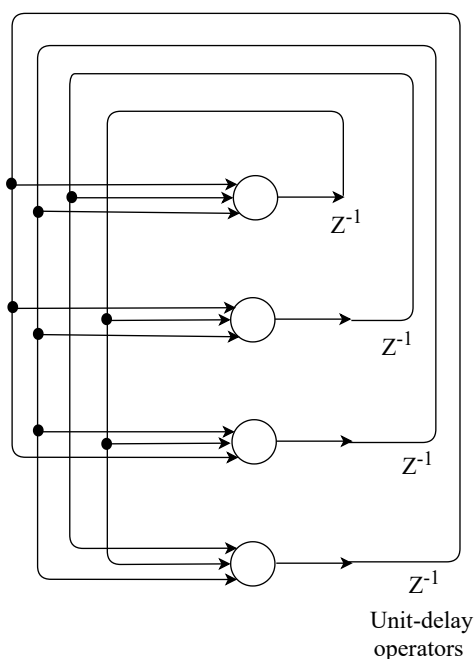


Figure 2.11: Recurrent network with no self-feedback loops and no hidden neurons [88].

- **Convolutional Neural Network**

convolutional neural network (CNN) Arrange has achieved groundbreaking results over the past decade in various areas related to pattern acknowledgement, from image preparation to voice recognition. CNN constitutes a specific class of DNN. The foremost advantageous perspective of CNNs is decreases the number of parameters in ANNs. This accomplishment has incited both analysts and designers to approach larger models to fathom complex errands, which was not conceivable with traditional ANNs. The most critical assumption is that the CNN solves problems should not have spatially dependent features. In other words, for example, in a confront discovery application, does not matter where the faces are located in the images. The only concern is to detect them regardless of their position in the given images. Four well-know CNNs are AlexNet, VGG-16, GoogleNet, and ResNet [89], [90].

2.4 Learning Processes

ANNs are models defined to imitate the learning capability of the human brain [91]. As in humans, training, validation and testing are essential components in creating such computational models. Artificial neural networks learn by receiving a variety of data-sets (which may be labelled or unlabelled) and computationally adjusting the network's free parameters adapted from the environment by means of simulation. Based on the learning rules and training methods, learning in ANNs can be categorised into supervised, reinforcement and unsupervised learning [92]–[94].

- **Supervised Learning**

In supervised learning, as the name suggests, an artificial neural network is supervised by a teacher (such as a systems designer) who uses their knowledge of the environment to train the network with labelled data-sets [94]. Thus, artificial neural networks are trained by receiving input and target pairs of different observations from labelled data-sets, processing the input, comparing the output with the target, calculating the error between the output and the target, and computing the signal use. Backpropagation regulates weights connecting neurons in a network to minimize errors and optimise performance. Fine-tuning of the network continues until the weights are reached, which reduces the discrepancy between the output and the desired output. Figure 2.12 shows the block diagram conceptualizing supervised learning in ANNs. Supervised learning techniques are used to solve classification and regression problems [95]. The output of a supervised learning algorithm can either be a classifier or a predictor [93]. The application of this method is limited to when the supervisor's knowledge of the environment is sufficient to supply the networks with input and desired output pairs for training.

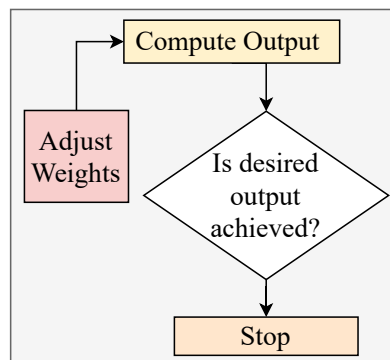


Figure 2.12: Supervised learning [96].

- **Unsupervised Learning**

Unsupervised learning is used when it is impossible to augment the training data sets with class identities (labels). This occurs in situations where there is no knowledge of the environment or the cost of acquiring such knowledge is too high [97]. In unsupervised learning, as its identify implies, the ANN is no longer below the supervision of a “teacher”. Instead, it is supplied with unlabelled data sets (containing only the input data) and left to locate patterns in the data and build a new model. In this situation, ANN learns to categorise the data by exploiting the distance between clusters within it.

- **Reinforcement Learning**

Reinforcement learning is another type of learning that concerns interaction with the environment, obtaining the state of such territory, choosing an action to change this state, sending the action to a simulator and receiving a numerical reward in the shape of feedback that can be positive or negative to learn a policy [93]–[95], [98]. Activities that increase the reward are chosen by trial-and-error methods [99]. Figure 2.13 illustrates the block diagram to explain the idea of reinforcement learning.

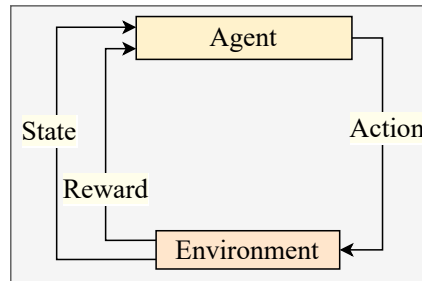


Figure 2.13: Reinforcement Learning [95].

2.5 Learning Algorithms

The MLP’s learning algorithm consists of two main steps: forward-propagation and backward-propagation. The hidden layer h_t and the output layer activation y_t are the significant blocks as a result of two main reasons; the higher power consumption and the extent of the occupation silicon area compared to other blocks in the MLP design. Chapter 4 presents the applications and investigated these two significant blocks h_t and y_t in a number of case studies.

2.5.1 Feed-forward Operations

- **From the input layer to the hidden layer**

The first operation is to feed all the neurons in the input layer with input values and weights. All inputs and weights are stored in matrices. This operation is the input into the hidden layer that can be described using equation (2.3).

$$h_t = f \sum (W_{hx}x_t + b_h), \quad (2.3)$$

$$a_h = \Phi(h_t) \quad (2.4)$$

where W_{hx} is the weight matrices for hidden connection. f and Φ are the activation functions for the hidden and b_h is the bias vectors for the hidden layer.

- **From the hidden layer to the output layer**

The output of the hidden layer will include bias, which is defined as an additional parameter in the neural network. However, the benefit of the bias is that it adjusts the output. This operation is the hidden layer to the output that can be described using the equation (2.5).

$$y_t = \sum (W_{yx}a_h + b_y), \quad (2.5)$$

$$a_y = \Phi(y_t) \quad (2.6)$$

where W_{yx} represents the weight matrices for the connection at the output layer and b_y is the bias vectors for the output layer.

2.5.2 Activation Functions

An important components is the nonlinear activation function, which is used at the output of every neuron. Several different activation function are currently available, such as Sigmoid, hyperbolic tangent (tanh), and Rectified Linear Unit (ReLU), Leaky ReLU, as well as Maxout. The most frequently used activation function in back propagation neural networks is the sigmoid function. Activations are typically represented by a number within the range of 0 to 1. The weight is double the activation function. It is most commonly a sigmoid function. It can be described using equation (2.7) [100].

$$sig(x) = \frac{1}{1 + e^{-x}} \quad (2.7)$$

2.5.3 Back-propagation Algorithm

Back-propagation is at the heart of every neural network. It is used to minimize the prediction error which results from the feed-forward operation. However, the Back-propagation Algorithm has certain advantages, such as it is simple to program, flexible as it does not require prior knowledge concerning the network and the function features do not require any particular discussion to be learned. Similarly, it is a standard method that generally works well. To calculate the error coefficient, the following equation in the output layer is used [101].

$$e_i = output_{desired} - out_{actual} \quad (2.8)$$

then the error gradient (g) at the output layer is calculated.

$$g_i = Dsig(Y_i)(e_i) \quad (2.9)$$

To reduce the prediction error, the weights and bias should be updated. The formula to change/update the weights and bias at the output layer is based on its derivative. The equations can be described as below.

$$\Delta W_{ji} = (\alpha)(Y_j)(g_i) \quad (2.10)$$

$$\Delta B_i = (\alpha)(-1)(g_i) \quad (2.11)$$

at the hidden layers, the error gradient is calculated using the below equation[101].

$$g_j = Dsig(Y_j)(W_{ji})(g_i) \quad (2.12)$$

Afterwards, the network weights and bias are updated, based on the following equation:

$$\Delta W_{ji} = (\alpha)(X_j)(g_i) \quad (2.13)$$

$$\Delta B_j = (\alpha)(-1)(g_i) \quad (2.14)$$

the deltas acquired are added as below:

$$W_j = W_j + \Delta W \quad (2.15)$$

$$B_j = B_j + \Delta B \quad (2.16)$$

According to equation (2.8) the new weight can be calculated by calculating the error coefficient and subtracting the desired output by the actual output which

provides two valuable pieces of information: the sign and magnitude. The derivative sign creates the relationship between the weight and the error. If the sign is positive, then the relationship is direct; whereas if it is negative, then it is the inverse. The derivative magnitude will give the needs value of the increasing or decreasing of the value of the weight. Furthermore, the system will continue to update the weights according to the derivatives and retrain the network until an acceptable error level is achieved.

Table 2.2: List of Abbreviations [100].

Abbreviations	Variables Name
e_i	Error Coefficient
g_i	Error Gradient
ΔW_{ji}	The change in Weights
ΔB_j	The change in Bias
e_i	Perseptron
e_i	Target
$output_{desired}$	Desired output
out_{actual}	Actual output
sig	Sigmoid Activation Function
$Dsig$	Derivative of the Sigmoid Activation Function
X_i	Network's input
Y_i	Network's input
α	Alpha
y_i	Network's output

Chapter 3

Configurable Approximate Multiplier

The previous chapter investigated approaches used to improve energy efficiency in the area of approximate circuit design. This chapter identifies the main core research contributions of this thesis and presents the proposed run-time configurable adaptive approximation method for multiplication that is capable of managing energy and performance trade-offs and which is also ideally suited for these systems. Central to the proposed approach is a significance-driven logic compression (SDLC) multiplier architecture that can dynamically adjust the level of approximation depending on power and accuracy constraints. The architecture can be configured to operate in the exact mode with no approximation or in progressively higher approximation modes at 2 to 4-bit SDLC. The architecture is designed in SystemVerilog and synthesized using Synopsys Design Compiler. The proposed method is implemented in both ASIC and FPGA. Post-synthesis experiments show a substantial decrease in power consumption as well as silicon area, compared to existing methods. The efficiency of the proposed design is evaluated through a number of case studies. The results demonstrate that the proposed approach incurs a negligible loss in output quality when using the 4-bit SDLC configuration for the 8-bit multiplier. The scalability of the configurable approximate multiplier approach and the feasibility of performing signed multiplication also discussed.

3.1 Introduction

Over the years, approximate hardware designs have been extensively researched. Many studies consider the trimming of arithmetic complexity leveraged by the inherent error-resilience of certain real-world applications. Examples span across various design strategies, such as accuracy scaling [19], approximation of parallel logical patterns [5], [102] and hardware/software approximations for NNs [103]. The commonality among these examples is to achieve power savings by employing static-configuration methods for specifying fixed approximate processing units, including approximate multipliers and adders [104], [105], based on design-time predictions of environment and data conditions.

In approaches that are aware of the bit-level precision [106], the more significant blocks are processed using exact arithmetic units, whilst non-significant blocks are processed using approximate and low-complexity ones. Recently, a significance-driven logic compression (SDLC) approximation method was proposed for multipliers, where complex logic operations are replaced by low-complexity logic gates for a group of partial product terms depending on progressive bit significance [25]. This method allows the computations to operate at different logic compression levels, designed in different multiplier units. By suitably choosing the multiplier logic compression level, the method offers a varying degree of energy efficiency and performance. To enable this, a number of multipliers are designed with selection circuitry, which is expensive in terms of area and leakage power [107].

Emerging ubiquitous systems, in particular, systems based on harvesting energy from the real-world through vibration[108], thermal energy[109], or from the environment through solar or kinetic energy [110] represent a paradigm shift from traditional systems. The energy supply of such systems can vary temporally and spatially within a dynamic range, essentially making computation extremely challenging [5], [111]–[113]. Adaptive hardware approximations with tunable energy and accuracy trade-offs can present opportunities in these systems for elastically continuing computation under varying power levels. Certain real-world applications rely on approximate computing due to its inherent error resilience. For instance, in the region 82% of run-time is spent on unnecessary computations, which can alternatively be executed in approximate modes [54] to reduce energy consumption.

3.1.1 Organization of the Chapter

The remainder of the chapter is organised as follows : Section 3.2 describes the SDLC method and the reason for this work. Section 3.3 explains in detail the pro-

posed design method. Section 3.4 presents error analysis and validates the configurable architecture by comparing with individual approximate multipliers. Section 3.5 compares the non-functional metrics, for instance area, speed and energy with competing designs. Section 3.6 presents the energy-aware configuration algorithm (EACA) algorithm and two cases of real-world problem solving. Finally, Section 3.7 concludes the chapter.

3.2 Existing SDLC Method and Motivation

Recently, Qiqieh *et al.* [102] suggested an approximate multiplier design with different levels of logic compression depending on bit significance known as Significance-driven Logic Compression (SDLC). Their investigation highlighted energy-accuracy trade-offs corresponding to these levels. The principle idea is to combine the partial product terms and compress them progressively based on significance (i.e. least significant bits are more compressed compared to more significant bits) by means of replacing the AND gates with the OR gates. Architecturally, this leads to reduction of the carry propagation chain length and thereby energy efficiency at the cost of minimal loss of accuracy loss.

Figure 3.1 illustrates the SDLC approach using a dot notation for the (8×8) parallel multiplier. This can be explained in three steps, as follows. Initially, (8×8) AND logic gates are used to generate the partial product matrix (PPM). The first step aims to form a cluster of several rows in the PPM. The depth of the clusters may be 2 or 4 bits (2- or 4-bit SDLC). The present work differs from the SDLC proposed by Qiqieh *et al.* [102] presented in Figure 2.5 and section 2.2.6, in that the 3-bit SDLC depth cluster has been removed due to its similarity to the 2-bit SDLC depth cluster in terms of accuracy and power consumption. The idea is to utilise the array of OR logic gates to reduce the number of product terms within each cluster. The next step describes how OR logic array is used to compress the terms into a single row of bits leading to a reduction of the total number of product terms within each cluster in the PPM to decrease, as seen in step 2. A commutative remapping of the bit sequence is applied in the third step to form a reduced number of the partial product rows after applying the logic compression. The number of the rows is dependent on the depth of the clusters selected in the first step. The dotted rectangles indicate the critical column's height, which is reduced by half in the case of 2-bit SDLC and by quarter in regard to the 4-bit SDLC in step 3 compared to the exact accumulation tree in step 1.

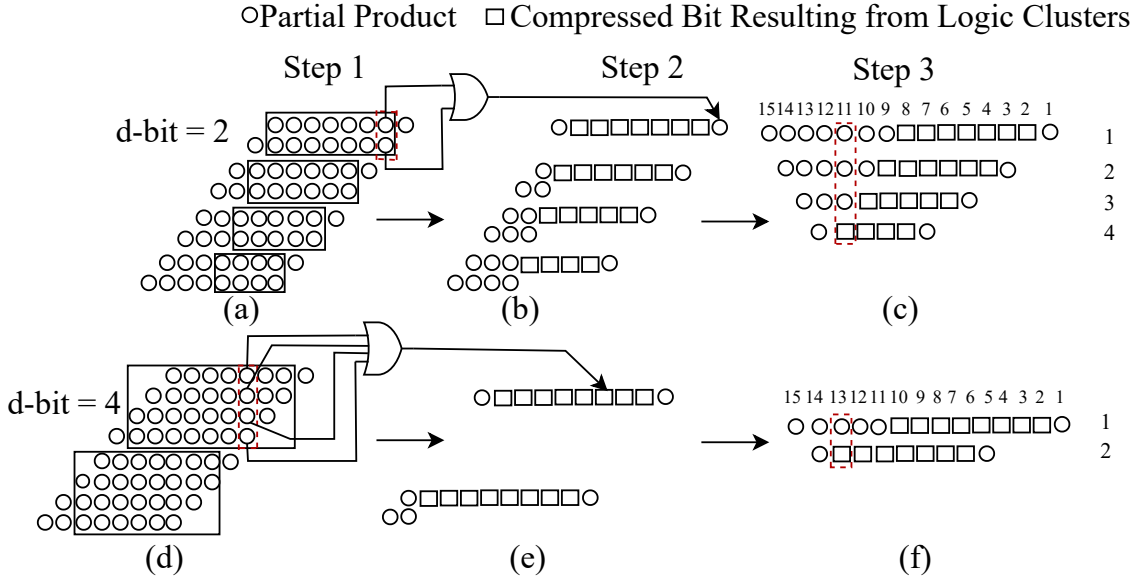


Figure 3.1: Two different sizes of logic clusters are used to compress partial products based on their progressive bit-significance in an (8 x 8) parallel multiplier architecture. (a) clustering a group of bits within two successive rows in the partial product bit-matrix after bitwise multiplication; (b) generating a reduced set of product terms after targeting the depth of 2-row logic compression; (c) ordered matrix after applying commutative remapping of the bit sequence resulting from the SDLC approach; (d), (e), and (f) the same process when applying 4-bit depth of logic clusters. The $d\text{-bit}$ indicates the depth of logic cluster.

Following these three steps, the resulting accuracy varies depending on the significant of the bit sizes. More compression (i.e. clustering of rows) is performed for bits of lower significance (compression occurs in the square cells) and progressively less compression is performed for higher significance bits (no compression for the round cells). The final product is then generally approximated with its loss of accuracy under control. As a consequence of shorter carry chain paths and a reduction of the number of single-bit adders, the energy is reduced drastically and speed is improved. Although the SDLC method provides substantial energy reduction, it lacks run-time configurability. In other words, the accuracy cannot be adjusted dynamically with variable compression levels. As a result, the multiplier models produced can provide only a specific range of performance, energy and power due to static design. In certain fields of application, such as artificial intelligence and signal processing and including speech technology, image processing, and mobile communication, variable accuracy can be leveraged opportunistically in favour of energy savings depending on availability of power or energy. For example, when power delivery is high it may be possible to tune the accuracy mode to a higher level, while in low power situations

the accuracy can be scaled down. This is completed with the aim of obtaining a longer operating lifetime and the survivability of the execution. Section 3.6 includes an exemplar of similar applications. To facilitate this, the SDLC based multiplier needs to be re-designed for run-time reconfigurability with power awareness. The key hypothesis is as follows: by designing variable SDLC knobs in a multiplier, it is possible to leverage SDLC levels in response to power levels dynamically and improve adaptability and survivability under in variable power situations. To corroborate the hypothesis, a new configurable and energyaware multiplier is designed using three different modes: an exact and two different approximate levels of 2- and 4-bit SDLC multipliers.

3.3 Proposed Configurable Approximation Hardware

Whilst it is possible to construct a system with copies of separate exact and approximate (e.g. SDLC) multipliers and select which one to use depending on the environmental conditions, such a design is wasteful of silicon and may cause substantial energy loss through leakage. This method is therefore not suitable for systems that have to operate under power and energy uncertainty. The following subsections present a novel, run-time configurable multiplier design that allows dynamic approximation needs in such applications.

3.3.1 Configurable Multiplier Architecture

A configurable multiplier is proposed, which provides exact and approximate versions for use under appropriate conditions. A configurable multiplier is proposed, which provides exact and approximate versions for use under appropriate conditions. This multiplier only requires an insignificant additional amount of silicon compared to a regular exact multiplier. This is realised by maximally re-using the single-bit adders for different configurations of the multiplier.

Figure 3.2 illustrates the difference between the exact and the proposed multiplier. Generally, the final product of exact multiplication can be generated after following three main steps: 1) the partial product matrix (PPM) is formed, 2) PPM is reduced to a height of two rows using any accumulation method such as Wallace or Dadda trees, then 3) these two rows are combined by using a (carry propagation adder (CPA) since a carry propagation adder is just needed to generate the product (no extra delay required for accumulation tree) (see Figure 3.2 (a)). In the proposed multiplier, the SDLC approach is included after the formation of the PPM (as shown in Figure 3.2 (b)), in order to minimize the number of rows in the PPM using logic

compression (see Figure 3.1). Therefore, this decreases the delay needed for the PPM to be reduced to a height of two rows (i.e. during stage 2). Subsequently, the minimized PPM is accumulated to a height of two rows and summed up using CPA. Likewise the same occurs in steps 2 and 3 of the exact multiplier. In this work, the proposed approach based on the Wallace tree structure is demonstrated. Like other multi-stage tree multipliers, it achieves high speed due to a lower logic depth compared to more conventional designs [114][115].

For this work the Wallace tree serves as an example. The method can be applied based on any exact multiplier design including multi-stage tree structures such as Wallace and Dadda trees. Fundamentally, adders in the multiplier serving as the exact configuration are re-purposed via re-wiring to implement the approximate configurations.

Figure 3.3 illustrates how the proposed configurable (8 x 8) multiplier design performs the exact and approximate multiplication. In this design, each circle rep-

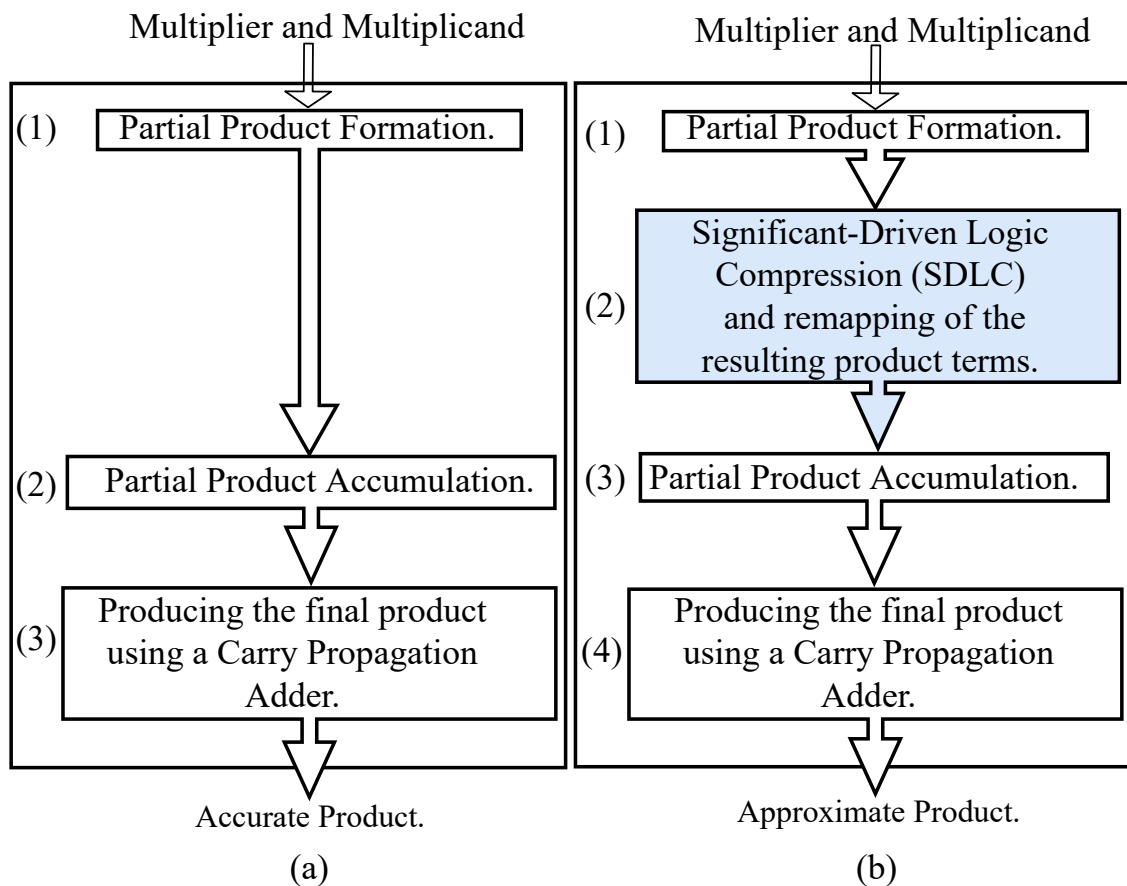


Figure 3.2: Process chart explaining the difference between the main stages in (a) exact multiplication and (b) the proposed multiplication using the SDLC approach.

Table 3.1: The number of full and half adders used by the accumulation stages and CPA for different exact, approximate multipliers and the Proposed configurable (8×8) Wallace tree.

Multiplier \rightarrow	Exact		2-bit SDLC		4-bit SDLC		Proposed	
	HA	FA	HA	FA	HA	FA	HA	FA
Stage 1	4	12	3	9	-	-	4	12
Stage 2	3	13	6	6	-	-	3	13
Stage 3	4	6	-	-	-	-	3	9
Stage 4	4	7	-	-	-	-	4	8
CPA	10		11		11		11	

resents one partial-product bit. The necessary half adders are marked by rectangles spanning columns of two partial product bits and full adders are marked by rectangles spanning columns of three bits. The exact configuration in (a) represents a traditional Wallace tree multiplier which uses the largest number of half and full-adder units within the reduction stages. The 2-bit SDLC configuration presented in (b) is considerably smaller, whilst the 4-bit SDLC in (c) is the smallest. It is worth noting that the proposed configurable design requires a few additional number of adders on top of the already existing adders needed by the exact configuration. By investigating the proposed configurable multiplier design with its exact and approximate (2 and 4-bit SDLC) configurations, structural similarities are identified in the shapes of the SDLCs and parts of the Wallace tree multiplier. For instance, in Figure 3.3, the 2-bit SDLC's reduction stages and the carry-propagate addition (CPA) parts in (b) can be mapped onto stages 3 and 4 and the CPA of the exact reduction tree in (a), with only ten additional partial-product bits required (shown in black and diameter-line circles). Similarly, the 4-bit SDLC in (c) can be mapped onto the CPA in (a) with just a single additional bit. This bit is also shared with the 2-bit SDLC.

Table 3.1 may be read in conjunction with Figure 3.3 to represent the required single-bit adders in all parts of the proposed configurable multiplier design. As can be noticed from the table, the exact configuration requires the majority of half and full adders. Only a very small number of adders are required in addition to the exact configuration to accommodate the SDLC configurations. In Section 3.5 the additional silicon area required by the SDLC's is calculated.

3.3.2 Hardware Knobs for Run-Time Configuration

In the case of the (8×8) multiplier, a 3-to-1 multiplexer (MUX) is essential for top-level configuration selection. This switch provides the actuation facility for controlling the configuration according to whatever rules the designer sets for the

configuration strategy. A configuration signal is received from the controller and fed to the MUX to select one of the three configurations accordingly. The configuration procedure is low-overhead, no more than a couple of layers of parallel switches. Even after including the two-bit to three-wire one-hot signal decoding logic to supply all the control signals required with no area overhead, the entire procedure should fit comfortably within a single clock cycle for any reasonably modern technology. In this work, the hardware into field programmable gate array (FPGA) was synthesized to characterise the energy and power required for executing each configuration (see Section 3.5). This is because it is eventually implemented on FPGA. If other technologies are used in the implementation, the same characterisation can be performed on the relevant technology. From these characterisation data, it is possible to derive energy thresholds that an energy-aware configuration algorithm may use to make control decisions.

If the rules of configuration are not determined by the availability of energy, different characterisation experiments may be carried out to produce appropriate threshold values in the alternative physical parameters that are important for the configuration control. This work, concentrates on an energy-aware configuration. For instance, one of the case studies, the EACA, which will be presented in Section 3.6.1, concentrates on efficient energy usage. Consequently, all experimental work is focused on energy being the crucial physical parameter. The hardware facilities that provide the hooks for configuration described in this section target energy-aware designs, but are generic enough to need no adjustment or slight adjustment for control based on other parameters.

With three different configurations in total, a configuration signal produced by the control module consists of two binary bits. The following settings are used: 00 for the exact configuration, 01 for the 2-bit SDLC configuration and 1x (with x representing "don't care", i.e. can be either 0 or 1) for the 4-bit SDLC configuration. This is shown in Figure 3.4. Assigning the 'don't care' to the 4-bit SDLC configuration shows the energy-centric design priority, as the smallest circuit is used to select the smallest configuration, which is likely chosen when the energy supply is low. Based on the configuration selection input signal, the control hooks perform the following actuation:

- the right groups of adders are included in the configuration - for the example in Section 3.3.1, the three selectable groups roughly correspond with the three parts of the exact Wallace tree: Stages 1 and 2, Stages 3 and 4, and the CPA. The exact configuration includes all three groups, the 2-bit SDLC configuration includes Stages 3 and 4 and the CPA, while the 4-bit SDLC includes the CPA.

- additional adders are included as appropriate for the SDLC configurations.
- appropriate re-wiring of the full and half adders (growing some half adders to full adders and shrinking some full adders to half adders) for the appropriate SDLC configurations.

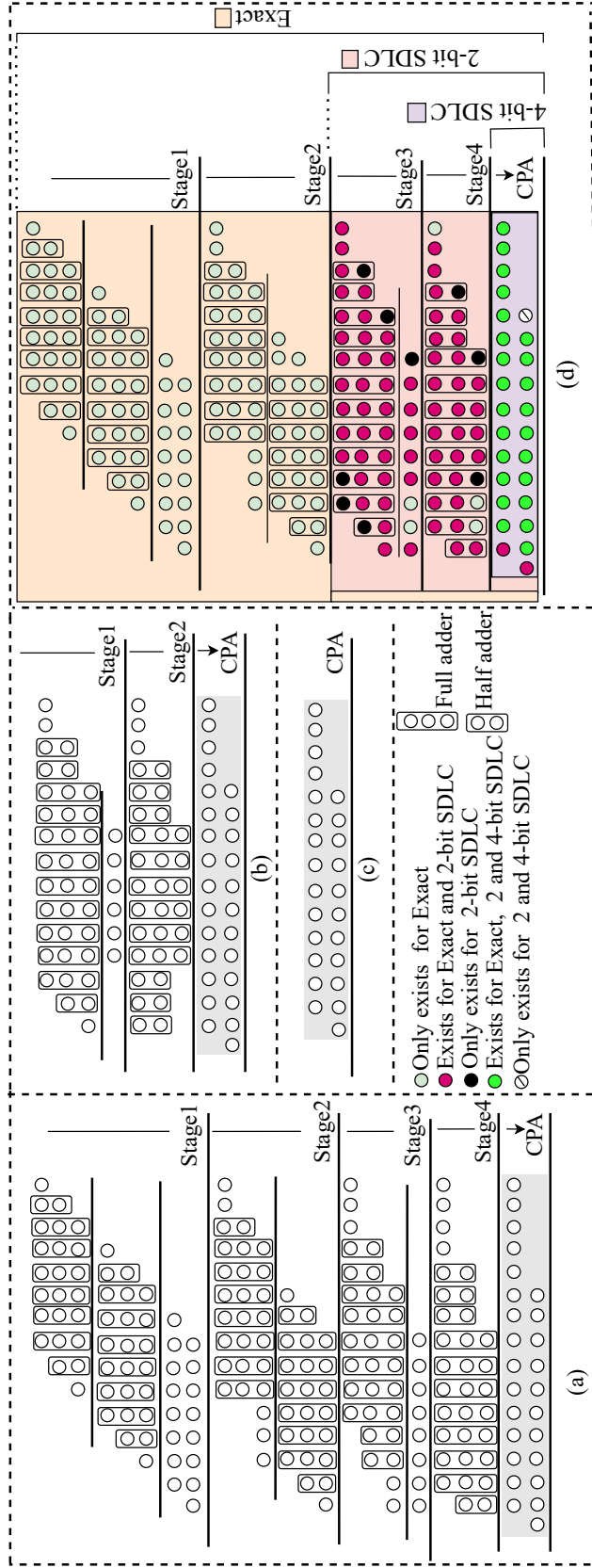


Figure 3.3: The reduction stages of an (8×8) Wallace tree multiplication illustrate the accumulation method for the PPM formed from exact and two different sizes of logic clusters (shown in Figure 3.1) (a) four reduction stages are required in the case of the (8×8) traditional Wallace tree multiplier(WTM); (b) two reduction stages are required by means of the Wallace accumulation method to reduce the PPM generated by the 2-bit SDLC(see Figure3.1 (c)); (c) no reduction stages for the 4-bit SDLC as the height of the PPM is only two rows(see Figure 3.1 (f)), and (d) configurable (8×8) Wallace tree multiplication includes the common similarities and variations shown in (a), (b) and (c).

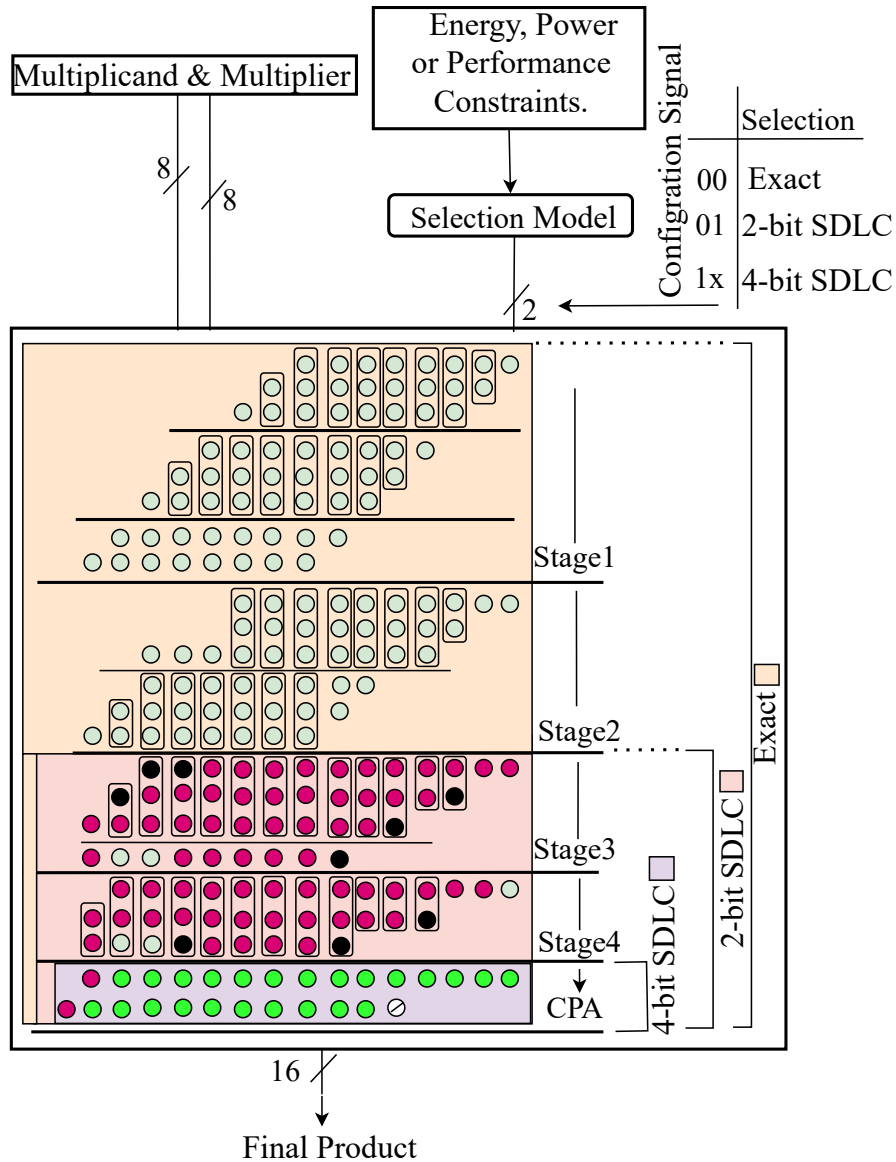


Figure 3.4: Diagrammatic sketch of the proposed hardware architecture of the configurable (8×8) multiplier with exact, 2-bit and 4-bit SDLC modes.

3.4 Error Analysis

In this section, the impact on accuracy of the proposed approach is investigated in the form of error analysis. Several error metrics have been discussed in [116] and [117] in relation to evaluating and quantifying errors. The error distance (ED) is defined as the arithmetic difference between the accurate (exact) product (P) and erroneous (approximate) product (P'). The relative error distance (RED) is the

ratio of ED over the accurate output. It is defined as in [118]:

$$RED = \frac{ED}{P} = \frac{|P - P'|}{P}. \quad (3.1)$$

The mean RED (MRED) is one more error metric for any ($N \times N$) approximate multiplier. It is defined as:

$$MRED = \frac{\sum_{i=0}^{2^{2N}-1} RED_i}{2^{2N}}. \quad (3.2)$$

MRED is the mean value of RED across all possible different operand multiplication pairs. Without recognising the application, is assumed here that each unique pair of operand values has exactly the same probability of occurring.

To compare multipliers of different degrees and methods of approximation, the normalised MRED (NMED)[119] is used:

$$NMED = \frac{MRED}{P_{max}} = \frac{\sum_{i=0}^{2^{2N}-1} RED_i}{2^{2N} P_{max}}, \quad (3.3)$$

where P_{max} is the maximum product that can be obtained from an ($N \times N$) exact multiplier, i.e. $P_{max} = (2^{N-1})^2$.

Simulation studies are performed in Matlab by incorporating a functional model of the proposed multiplier using different logic clusters with (8×8) Wallace tree accumulation.

Table 3.2: MRED and NMED for different approximate configurations in an (8×8) configurable multiplier.

Multiplier ↓	MRED (%)	NMED(%)	Difference from [102]
2-bit SDLC	1.9883	0.3527	0 (%)
4-bit SDLC	10.5836	3.2723	0 (%)

Table 3.2 lists the error trade-off when changes take place between the two degrees of approximation (2- and 4-bit SDLC) in the new design. As can be seen, the MRED is only minimal for the 2-bit SDLC type. The 4-bit SDLC also records a small error. A similar observation can be made in the case of the NMED metrics. Moreover, the table includes the difference between the approximation parts in this new design and the previous work [102], in order to investigate if by moving to a configurable design, a price is paid in errors. The results indicate that the differences are 0%, and there is no error overhead.

For more extensive analysis involving error metrics, for instance in the case study

in Section 3.6.2, different configurations of the proposed configurable multiplier are used in calculating the Gaussian blur algorithm. Such metrics as peak signal to noise ratio (PSNR) are used. PSNR is a fidelity metric used to measure the quality of the output images. PSNR is expressed as:

$$PSNR = 20 \log_{10} \frac{255}{\sqrt{MSE}}, \quad (3.4)$$

where MSE is the mean squared-error measured with respect to the reference pixel [120]. The total MSE of the image can be determined as the sum of all sub-image MSE values as follows:

$$MSE_B^k = \frac{1}{m \cdot n} \sum_{i=1}^m \sum_{j=1}^n (B_k(i, j) - B'_k(i, j))^2. \quad (3.5)$$

These PSNR analysis methods are used in Section 3.6.2.

3.5 Comparative Evaluations

In this section, the area, delay and power trade-offs of the proposed multiplier design are compared with recently proposed approaches, considering two hardware implementations: application-specific integrated circuit (ASIC) and FPGA. These implementations are included in order to achieve a wider insight of comparative evaluations as some of the compared implementations are in ASIC, while others are in FPGA.

3.5.1 Area, Delay & Power Trade-offs in ASIC Implementations

For ASIC comparisons, a generic System-Verilog code¹ is used to generate synthesizable modules for the proposed configurable multiplier. Mentor Graphics Questa Sim is used to compile the System-Verilog code and run the associated testbenches. Synopsys Design Compiler is used for synthesising the multiplier configurations. The circuits are implemented in the Faraday 90nm technology library. The compared methods are implemented in exactly the same way, so that comparisons can be performed on the same implementation technology node and library. Figure 3.5 demonstrates the main steps for evaluating the proposed design using Synopsys Design Compile.

¹The source code is available on [121].

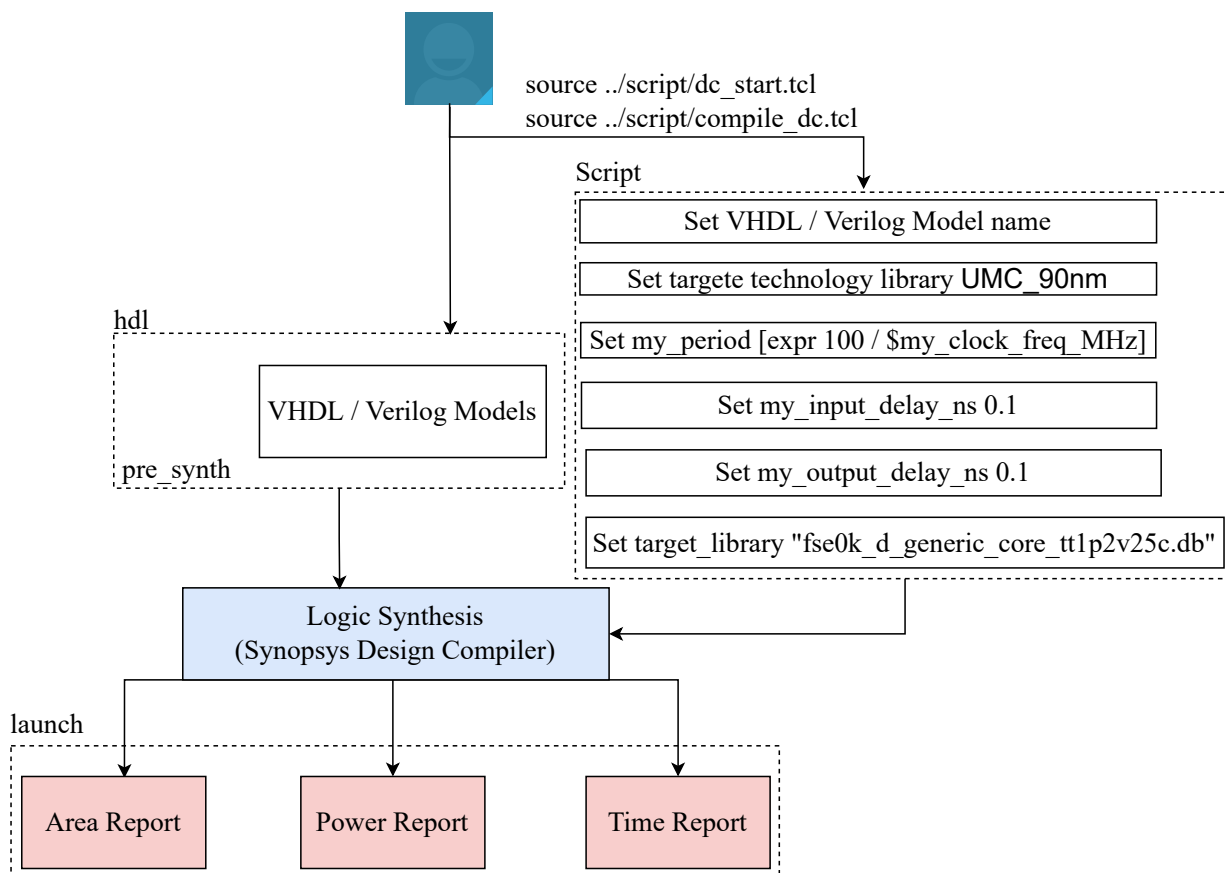


Figure 3.5: Flowchart diagram showing the main steps for evaluating the proposed design using Synopsys Design Compile.

Table 3.3 presents area, delay and power trade-off figures when compared with [102]. As can be noticed, the configurable hardware is larger in terms of area than the exact multiplier alone. A delay overhead is also expected in the configurable design because of the increased number of adders.

Table 3.3: comparing existing multiplier designs and the proposed configurable design in terms of power (P), area (A) delay (DL) and Power-delay product (PDP).

Multiplier ↓	$P(\mu W)$	$A(\mu m^2)$	$DL(\text{ns})$	$PDP(fJ)$
Fixed Configs [102]	158.39	3495.71	7.82	1238.6
Proposed Exact	66.20	1450.40	2.66	176.1
Exact [25]	62.42	1417.47	2.63	164.2
Proposed 2-bit SDLC	39.21	904.56	2.11	82.7
Proposed 4-bit SDLC	25.42	501.37	1.35	34.32

It can be seen from Table 3.3 that the area overhead, which compares the configurable multiplier (Proposed Exact) to the exact multiplier on which it is based

(Exact [25]), is 2.3% increase of $1417.47(\mu m^2)$. The power overhead is 6%, the latency overhead is 1.1%, and the power-delay product overhead is 7.2% increase of $164.2(fJ)$. However, the proposed model (second row) saves more than 58.5% decrease of $3495.71(\mu m^2)$ which is the area of the solution proposed in [102](first row), which includes separate exact and SDLC multipliers for run-time selection. This large area reduction implies significant power savings because of leakage power, leading to large PDP reductions. The competitive figures obtained on these non-functional metrics suggest that this configurable design would also compare favourably against the other designs in [25] and [102].

3.5.2 Area, Delay & Power Trade-offs in FPGA Implementations

For a more flexible design, the configurable multiplier is also implemented in FPGA using Xilinx Vivado Design Suite for the Ultra96-V2 platform [122]. The compared existing designs, originally also on FPGA, are re-implemented on this same platform for fair comparisons.

A previous study includes different designs of exact multipliers on FPGA (see Table 3.4), focusing on performance [42]. These are compared with the three configurations for non-functional parameters. Table 3.4 lists the results for each design in terms of area, delay and power. It can be observed that the proposed configurable multiplier, in its different configurations (between exact to 4-bit), is competitive in terms of area, delay, and power compared to Modified Radix2 Booth Multiplier modified radix2 booth multiplier (MRBM) and wallace tree multiplier (WTM). It is noteworthy that the exact configuration is competitive in delay with these multipliers, which were designed for speed.

Table 3.4: Comparing non-functional metrics with Kumar et al [42].

Multiplier ↓	Area (LUT's)	Power (W)	Delay (ns)
MRBM [42]	137	1.010	6.721
WTM [42]	96	1.061	6.102
Proposed Exact	91	1.22	6.102
Proposed 2-bit SDLC	65	0.32	4.1
Proposed 4-bit SDLC	42	0.23	3.8

3.6 Case Studies

In this section, an example configuration selection algorithm is first illustrated. This algorithm attempts to find the maximum usage of available energy by selecting the

least approximate configuration. Later, a case study is presented for the configurable MAC in real-world problems. The case study demonstrates the capabilities of the proposed configurable multiplier as well as the validity of the configuration selection algorithm.

3.6.1 Energy-Aware Configuration Algorithm (EACA)

In order to have a design that can survive under unreliable power supply (nondeterministic fluctuations in power levels) and guarantee reliable computation, a system with survival instincts needs to be built. A configuration controller is designed taking advantage of the three-mode configurable multiplier architecture to implement energy-aware execution. The EACA model, which fits into the "Selection Model" box in Figure 3.4, is shown in Algorithm 1.

Algorithm 1 Energy-aware configuration algorithm

```

1: Initialize with energy figures
2: const: k = number of different approximation configurations
3: for i = 1 to k do
4:   E (i) = energy required for the  $i^{th}$  configuration
5: end for                                ▷ i = 1 is exact and i = k is the min config
6: begin EACA
7: while true do
8:   Eia = 0                                ▷ instantaneous available energy
9:   j=1                                    ▷ initialize index
10:  wait control cycle time length
11:  while Eia < E(k) do
12:    obtain Eia from energy supply
13:  end while                               ▷ until enough energy for min config
14:  while Eia < E(j) do
15:    j = j+1
16:  end while                               ▷ find the least approx config for Eia
17:  select the  $j^{th}$  configuration
18: end while
19: end EACA

```

The EACA assumes the existence of energy supply information at run-time, which is not uncommon among well-designed energy harvesting systems [123],[124],[125]. The available energy is subsequently compared with the required energy to execute the least energy-hungry configuration. If the available energy is insufficient, the multiplier is not executed whilst the monitoring of incoming energy continues. Once the available energy is confirmed to be a sufficient amount to execute at least the configuration with the smallest energy requirement (in the (8×8) multiplier ex-

ample in this work, this is the 4-bit SDLC configuration), the EACA continues to the configuration selection stage. In this stage, the EACA progressively tests the available energy against the energy required by the different configurations one by one, starting from the heaviest configuration (Exact multiplier configuration) with the smallest approximation and moving orderly towards the lightest configuration, which has the greatest degree of approximation. In principle, the EACA attempts the following optimisation problem: Treating the available energy as a constraint, maximise the multiplication accuracy (or minimize the multiplication approximation) by selecting the correct configuration by sending a signal to activate the configuration that fits the available energy.

3.6.2 Gaussian Blur Filter

In this case study, the efficacy of the proposed technique is evaluated with a real-life image processing application, which consists of additions and multiplications using the three multiplier configurations. The analysis considers the Gaussian blur filter [126], given that it is commonly employed in graphics software, typically to reduce image noise and artifacts (e.g. Moiré effects) by acting as a low-pass filter. Figure 3.6 demonstrates a test platform to examine the effectiveness of the proposed multiplier on the quality of final output image processed by Gaussian blur filter. All configurations (2-bit and 4-bit SDLC) are implemented in Matlab. The Gaussian kernel is (3×3) with a 1.5 standard deviation value. It uses 8-bit fixed point arithmetic and is applied to 8-bit grayscale input images comprising (500×500) pixels. Gaussian blur is approximated by replacing the standard multiplication in the Gaussian filter with the two approximate multipliers configurations (2-bit and 4-bit SDLC).

Figure 3.7 illustrates the impact of different configurations on the image quality after applying the Gaussian blur filter. As shown in Figure 3.7 the use of the SDLC approach can yield reasonable results. The PSNR values for the case of 2- and 4-bit logic clustering for (8×8) SDLC are 50.2dB and 30dB respectively. The values of PSNR values are computed compared to the reference image, obtained from applying Gaussian blur filtering using the exact configuration, according to (3.4).

To calculate the energy consumed in the multiplier when processing an input image, the equation 3.6 is follow

$$Energy = Power * Delay * N, \quad (3.6)$$

where Power and Delay are obtained for one multiplier design from the synthesis tool.

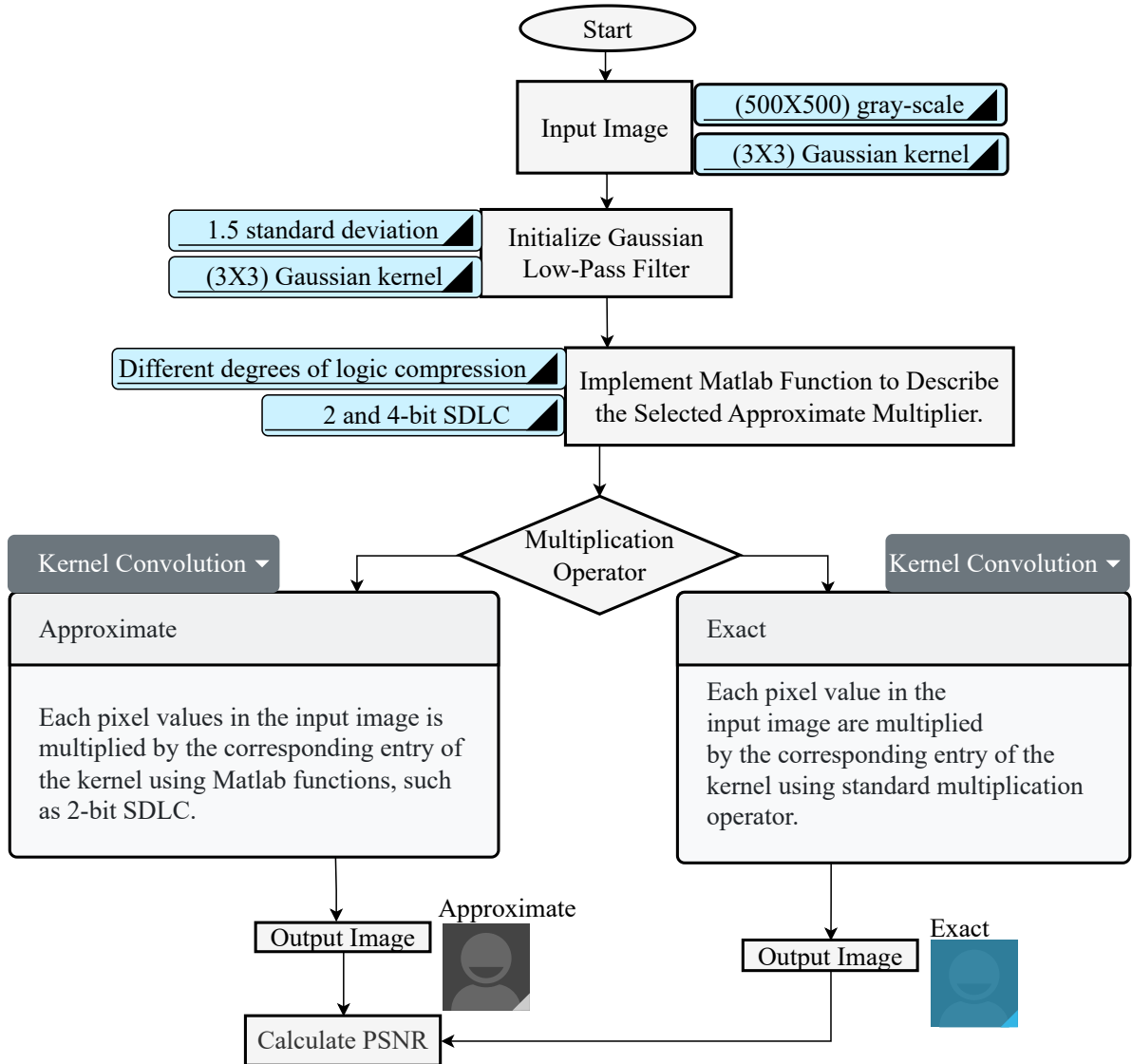


Figure 3.6: Flowchart diagram showing the main steps for evaluating the impact of the proposed multiplier on the final quality of image processed by Gaussian blur filter.

N is the number of multiplications necessary to treat the input image by Gaussian filter. The energy savings are subsequently calculated compared to the conventional exact multiplier. The energy savings and PSNR results obtained, shown in Figure 3.7, demonstrate that the proposed approach can provide significant dynamic energy savings with acceptable image qualities. The quality-energy trade-off evident throughout the three configurations.




Exact Multiplier	2-bit SDLC	4-bit SDLC
		
Reference Image	PSNR = 50.2	PSNR = 30
Energy Saving/Image	34.8 %	62.6 %

Figure 3.7: Output quality and energy consumption for Gaussian blur filtering using the three different configurations of the proposed (8×8) multiplier.

3.6.3 Energy-Aware Approximation

To evaluate the energy consumed by the proposed configurable multiplier, the configurable multiplier is described in System-Verilog, including its exact and approximate variations. The configurable multiplier is then synthesized using Synopsys Design Compiler [127]. This has been achieved by mapping the circuit designs to Faraday's $90nm$ technology library [128] and following the design set-up of $1V$ voltage units, $1.0pf$ capacitance units and $1ns$ time units to measure the power consumption related to each configuration. These measurements are then incorporated into the EACA model to allocate the optimal design configurations for each possible scenario. For example, in the case of the (8×8) multiplier, the threshold values $E(i)$ listed in line 4 in Algorithm 1 are obtained after the synthesis of the exact, 2-, and 4-bit SDLC variations.

Figure 3.8 demonstrates an example execution trace where the supply energy is variable within a wide range over time. The highest amount of energy available is shown to be 250 fJ, whereas the lowest amount of energy available is shown to be 35 fJ. EACA's energy-aware configuration supports the system to achieve reasonable PSNR figures in the execution for the available energy at any time. The priority is survival of the continuous execution by optimising PSNR whilst satisfying energy constraints. This trace confirms that this configurable multiplier supports

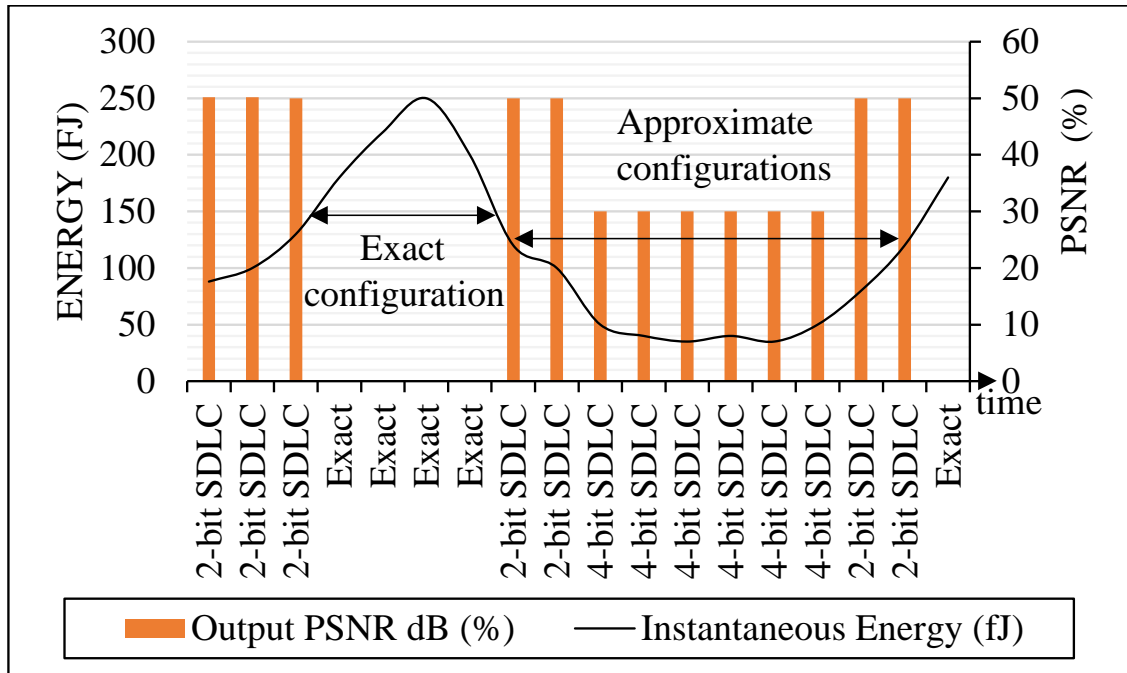


Figure 3.8: Different scenarios for the EACA model operate at run-time under highly variable energy conditions while sustaining execution.

the EACA’s sustainability focus. In this experiment, the extra overhead used in the configuration circuits prove to be negligible compared to the multiplier’s costs. The EACA itself is assumed to be run on an outside processor whose energy is external to the energy budget shown, without losing generality.

3.7 Conclusions

This chapter presented an investigation into facilitating energy efficiency. This investigation involves a configurable (8×8) multiplier with different levels of approximations achieved via configurable logic clustering. The System-Verilog design was implemented on an ASIC, using Synopsys Design Compiler and Xilinx Vivado Design Suite. The synthesised results confirm that up to 43.75% of energy savings and a reduction in critical path delay of virtually 38.03% can be achieved. The proposed multiplier is also implemented on the Ultra96v2 Evaluation FPGA Platform. The results confirm that the proposed approach can provide significant energy and area savings with negligible loss in output quality (the worst PSNR is 30dB for the 4-bit SDLC multiplier). The capabilities of the configurable multiplier are demonstrated by introducing the EACA method of obtaining the optimal configuration of the mul-

multiplier depending on the available energy.

The results reveal that the EACA model allows the proposed design to operate at run-time under highly variable energy conditions while sustaining execution. The highlight of this configurable multiplier is the sharing of the same adders between different configurations, saving both silicon and leakage energy. It is believed that the proposed multiplier architecture leverages arithmetic approximation to support energy-efficiency for applications where detailed accuracy may not be important, such as ML. The following chapter investigates the application of the proposed multiplier architecture as a ML hardware design.

Chapter 4

Neural Network Design with Run-time Configurable Approximate MAC

Chapter 3 described a Run-time Configurable Approximate Multiplier using significance-driven logic compression. This chapter introduces a configurable-approximation multiply-accumulate (MAC) unit, which is the fundamental arithmetic component in Neural Networks based machine learning (ML) systems. The new configurable MAC unit is designed using an adaptive SDLC multiplier architecture that is presented in Chapter 3. Additionally, this chapter explains the implementation that allows the evaluation of the capabilities of the proposed MAC unit as the key processing blocks in neural network applications and the validation of the dynamic control of the trade-off between accuracy and power consumption. The new MAC architecture using variable approximation, supports run-time configuration, and the proposed MAC architecture is implemented in an ANN and validated using different data-sets to evaluate trade-offs between performance, energy, and quality. The chapter also demonstrates the implementations of the proposed architectures in a number of case studies. To evaluate the trade-offs, the hardware architecture is designed in System-Verilog and synthesized using Synopsys Design Compiler, employing UMC 90nm digital complementary metal-oxide semiconductor (CMOS) technology as well as Field Programmable Gate Arrays (FPGAs).

4.1 Introduction

Energy efficiency is a primary design objective to enable powerful artificial intelligence (AI) applications at the microedge [129]. The traditional hallmark in AI systems is inspired by the principle of neural network (NN), originally proposed by Rosenblatt [130], [131]. An NN is organised in multiple layers where a learning problem is defined by the weighted sum of the input data. An activation function normalises the weight updates in each path, which are carried out by iterative gradient descent exercises during training. The NN implementations consist of the modular electronic neurons, known as multiply-accumulate (MAC) units, that primarily feature arithmetic operations [132]. The number of MACs as well as the complexity increases with more inputs, along with the learning problem at hand. As such, achieving the required training accuracy under a limited power or energy budget has remained challenging [133]. Over the years, energy-efficient AI designs have been extensively researched. Many studies consider arithmetic complexity trimming leveraged by the inherent resilience of AI applications. Examples span over various design strategies, such as accuracy scaling, network sparsification [134], approximation of parallel logical patterns [5] and hardware/software co-design for NNs [19], [103]. Approximate arithmetic, such as approximate multipliers and adders [28], can reduce energy requirements, enhance speed and reduce cost [105]. As such, the primary emphasis will be on approximation methods applied in multipliers as part of the new MAC designs. A common trait among existing methods is to minimize energy by exploiting static approximation methods that provides for the pruning of the hardware resources used to compute a pre-established number of least significant bits (LSBs) of the product [135]–[137]. However, this sort of static configuration does not allow boosting computational capability under varying power or energy at run-time, which this work aims to achieve. Recently, a significance-driven logic Compression (SDLC) approximation approach has been proposed for multipliers, where complex logic operations are replaced by low-complexity logic gates for a group of partial product terms depending on progressive bit significance [25]. This approach provides the computations to operate at different logic compression levels, designed in different multiplier units. By suitably choosing the multiplier logic compression level, the method offers a varying degree of energy efficiency and performance.

4.1.1 Contributions

For this work, the SDLC approximation method is extended further as follows. A new configurable MAC unit is design using an adaptive SDLC multiplier architec-

ture. The architecture leverages a tunable approximation method to specify the logic compression level depending on the model-driven energy and accuracy trade-offs.

Thus, by allocating the appropriate configurations during run-time, the computation capability under variable power or energy budgets can be adjusted. Specifically, this study makes the following *contributions*:

- propose a new MAC architecture using variable approximation, which supports run-time configuration; and
- implement the proposed MAC architecture in an ANN and validate it by employing different data-sets such as (modified national institute of standards and technology (MNIST) and street view house numbers (SVHN) to evaluate the performance-energy-quality trade-offs.

4.1.2 Chapter Organisation

The remainder of the chapter is organised as follows: Section 4.2 presents the reason for this work. Section 4.3 explains in detail the proposed design method, presents the neuron structure design using configurable MACs. Section 4.4 shows the simulation and evaluation results and compares the non-functional metrics, for instance, area, speed and energy with competing designs. Section 4.5 presents a case of real-world problem solving. Finally, Section 4.6 concludes the chapter.

4.2 SDLC Method and Motivation

In certain fields of application, such as AI and signal processing, variable accuracy can be leveraged in favor of energy savings opportunistically depending on the power or energy availability. For example, when power delivery is high it may be possible to tune the accuracy mode to a higher level, while in low power situations the accuracy can be scaled down. This is completed with the aim of a longer operating lifetime and the survivability of the execution. Section 4.5 includes an exemplar of similar applications. Using the SDLC approach this can be undertaken by static allocation of different MAC units with separately designed multipliers with variable degrees of approximation. However, this is likely to significantly add to the overhead of the design in terms of power and area costs.

To facilitate a run-time configurable MAC architecture with low overheads caused by a few additional adders on top of the already existing adders, it is important to re-design the SDLC based multiplier for run-time reconfigurability with power

awareness. The key hypothesis is as follows: by designing a low-complexity configurable SDLC MAC with variable compression knobs(SDLC levels 2-bit and 4-bit), it is possible to leverage the SDLC levels in response to power levels dynamically and improve adaptability under variable power situations. To corroborate the hypothesis, a new configurable and energy-aware MAC is designed using three different modes built on the same MAC architecture: an exact and two different approximate levels of 2- and 4-bit SDLC multipliers(depths of logic clusters in 8×8 multiplier size). Further details of the new MAC design as well as the neural network structure are provided next.

4.3 Run-time Configurable Approximate Neuron Design

This section illustrates the configurable approximation neuron architecture, followed by the reconfigurable circuits supporting three different modes(Exact, 2-bit and 4-bit SDLC).

4.3.1 Configurable Neuron Architecture

A configurable multiply-accumulate (MAC) unit is designed using the SDLC approach as the core part of a neuron module. A carry look-ahead adder (CLA) is employed to accumulate the results of the multiplication. CLA is preferred over other types of adders for low power and high-performance considerations [138]. Besides MACs, there are other components that require multiplication and other arithmetic, such as gradient update units. However, given that MACs constitute a significant proportion of neural networks (NNs), they are the focus for run-time adaptation in this work.

The neuron module shown in Figure 4.1 accepts the weights and input pairs (fixed-point)[139], where each pair is multiplied together through MACs. For each input pair, the result of the multiplication is sequentially accumulated by a CLA. The final sum is passed through the activation function to produce the neuron's output. An energy-aware configuration algorithm (EACA)(see Chapter 3, Section 3.6.1, Algorithm 1)is used to allocate the optimal configuration of the MAC unit in the neuron module depending on the instantaneous available power and provide higher performance, with acceptable levels of accuracy. All neurons are fed with the available power to avoid being ideal.

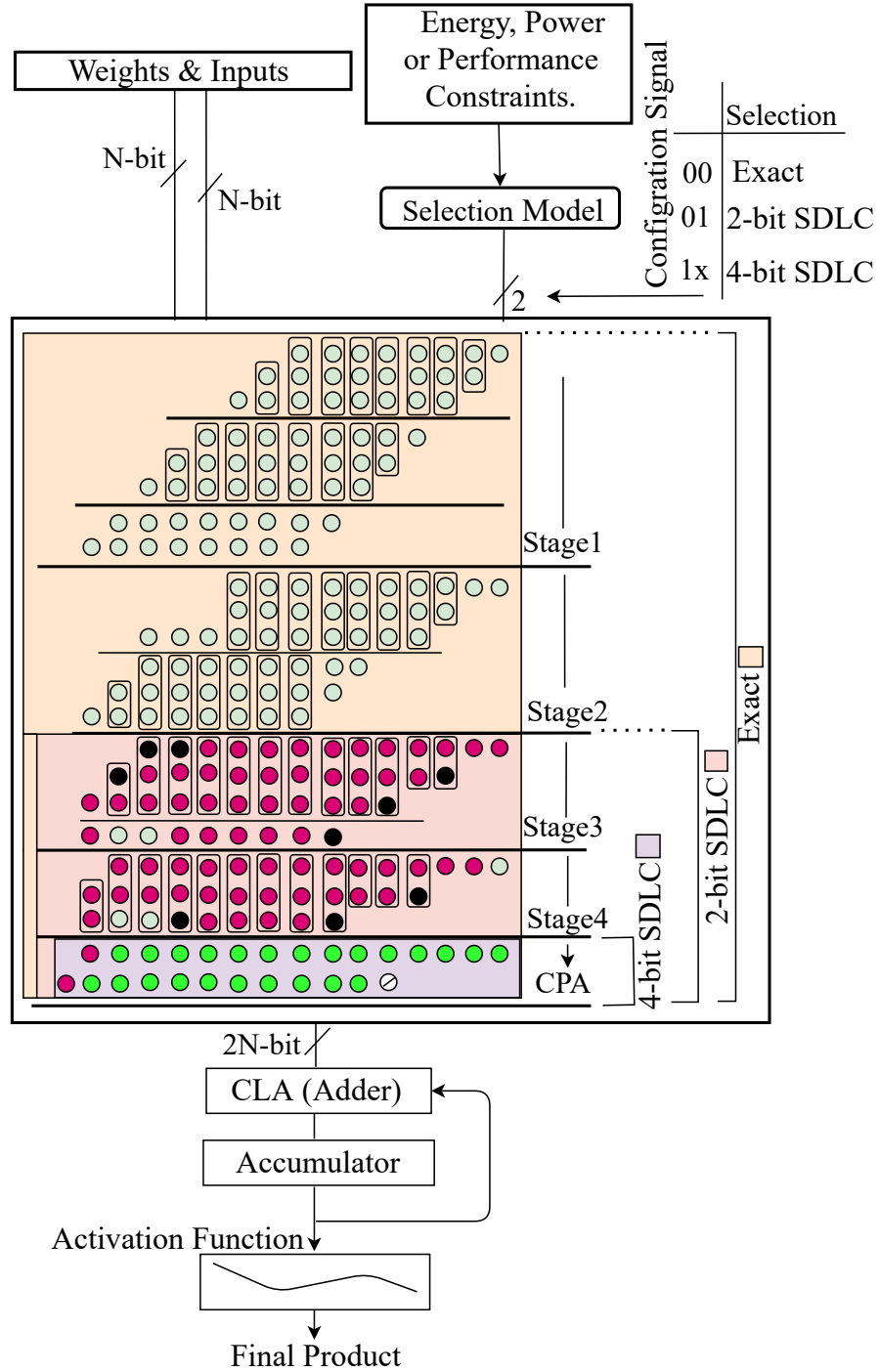


Figure 4.1: Neuron structure-serial processing including the proposed hardware architecture of the configurable MAC unit.

4.3.2 MAC unit reconfigurable circuits

The proposed design shown in Figure 4.2 includes three modes consists of three subcircuits and works as follows. During the Exact mode the EACA send a signal to active all three subcircuits and disable unused circuits that required for other

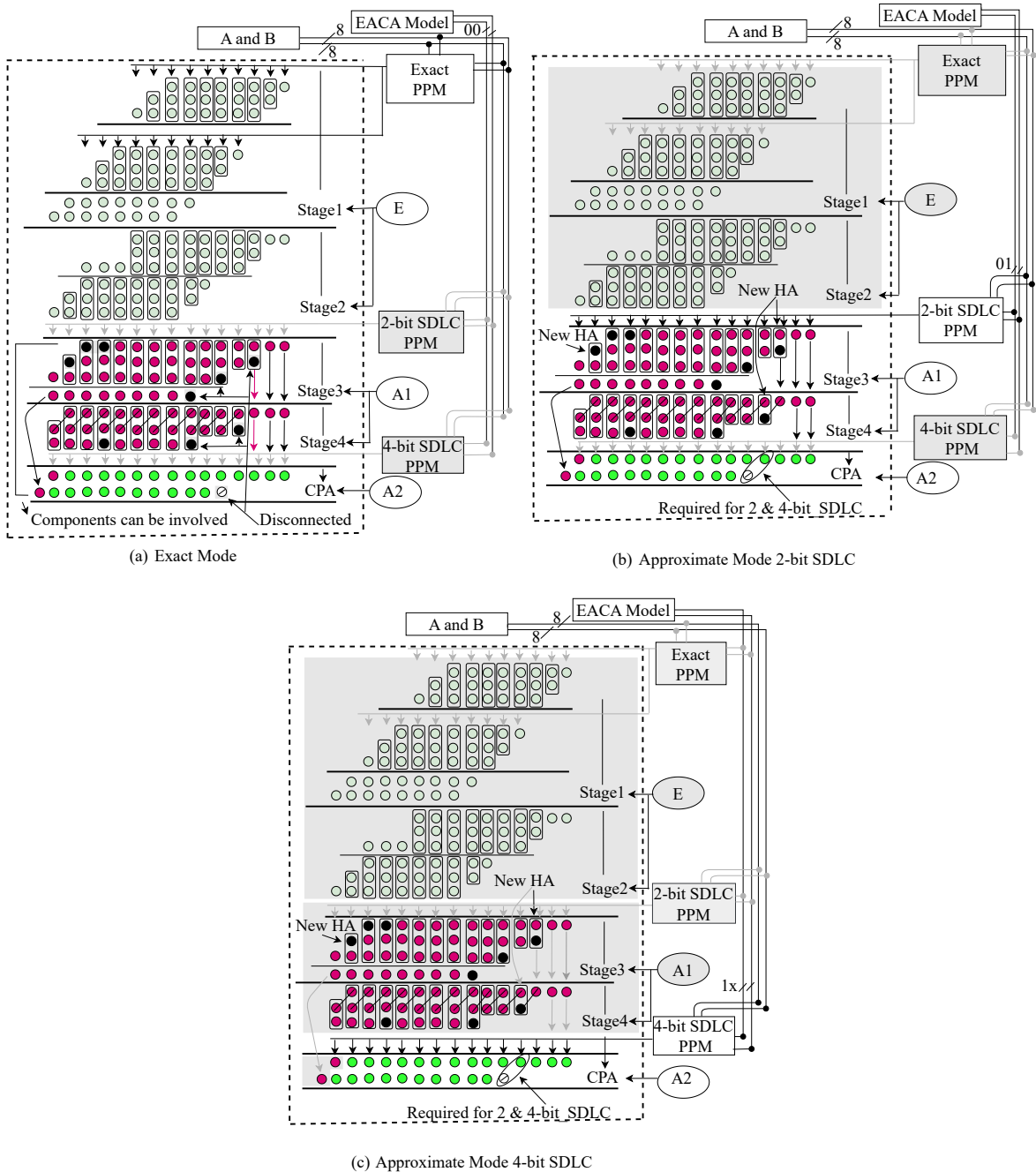


Figure 4.2: Representation of the reconfigurable circuits supporting three different modes.

modes in order to prepare the circuit for the inputs coming from PPM operation. As well the EACA is responsible for generating a signal to select the PPM mode. Then the PPM will send the results to the adders in the stage. The approximate mode, includes two circuits A1, A2 where a majority components can be involved between and active during the Exact and 2-bit SDLIC approximate Mode. The remaining logic in E circuit will switched off by EACA model. It means that both circuits A1 and A2 are only active in this mode (see Figure4.2 (b)). The third mode is 4-bit SDLIC located at A2 circuit. In this mode the scenario EACA is optimized to switch off all the two subcircuits E and A1 and some components in A2 can remain disconnected. The EACA is optimized for the scenario in which the approximate mode is active for a predominant part of the whole runtime. This is the reason, why A2 is always powered. The goal of the evolution is to find a circuit showing the highest possible power reduction when it operates in the approximate mode.

4.4 Implementation of ANNs

Using the neuron module (Figure 4.1), a number of ANNs are implemented targeting well-known ML data-sets. Table 4.1 lists the data-sets including the required number of MACs required in neurons. The Noisy XOR data-set contains 12 binary inputs,

Table 4.1: Numbers of input, layers and required neurons/MACs for different data-sets.

Data-set	No. of Input	No. of Layers	No. of Neurons
Noisy XOR [140]	12	2	13
Binary IRIS[141]	16	2	103
MNIST[142]	784	3	660
SVHN[143]	1024	6	1560

two of which are related by XOR with the remaining 10 inputs randomised. The training set provides 10000 examples and 40% of the outputs are inverted for added noise. For this reason, learning accuracy is limited to 80%. The test set provides a further 10000 examples, this time without output inversion, meaning that 100% test accuracy is theoretically possible [144].

The binary IRIS data-set is a further ANN architecture which has been implemented. Found in [145] and [146], the proposed ANN consists of two hidden layers. Each layer includes the proposed MAC unit connected to every node (see Table 4.1). After every hidden layer, the neuron module includes an activation function to define the output (Figure 4.1). To achieve a more efficient model and control the learning process all the parameters such as the numbers of neurons, layers and inputs have

been set. The learning algorithm of the ANN consists of two main stages, namely forward-propagation and backward-propagation. However, a single hidden layer, which computes a hidden layer h_t and the activation output layer y_t are the significant blocks, as previously mentioned in 2, due to two main reasons: its higher power consumption and its silicon area compared to other blocks in the ANN design. The following part describes the hidden layer and activation of output layer.

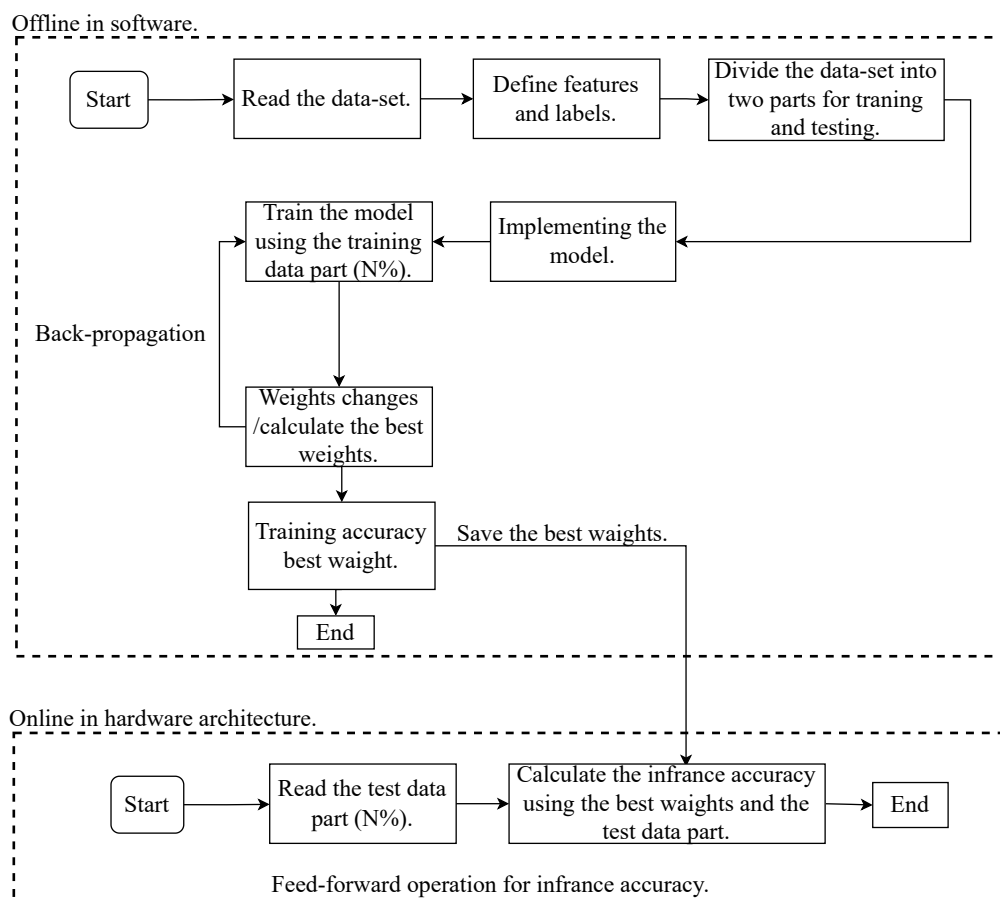


Figure 4.3: Main steps for training and testing a Neural Network.

To demonstrate the proposed approach, the neuron module in System-Verilog to implement the ANN is described. The neuron modules are then synthesised with different configurations using Synopsys Design Compiler with the circuits mapped onto the Faraday 90nm technology library to measure the power consumption, silicon area and the performance profiles related to each configuration. Additionally, the library setup time is $9.93ns$ for the operation speed. These measurements are subsequently modeled within the EACA model to allocate the optimal design re-

quirements for the ANN.

After the ANN circuit is implemented, it is trained using 60% of the Noisy XOR and Binary IRIS data-set [141] with the back-propagation learning algorithm by working offline to calculate the best weights (see Figure 4.3). Then, the remainder of the data-set is used to test the data-sets online by means of the feed-forward in hardware architecture after adding a range of possible formulations that are the different MAC unit configurations. In terms of learning, it is apparent that the best weights were achieved at the 5th iteration for the Noisy XOR data-set, and at the 50th iteration for the Binary IRIS data-set.

The *Exact* ANN version using wallace-tree multiplier (WTM) [27] is the most efficient for learning, followed by the SDLCs types 2-, and 4-bit SDLCs, as demonstrated in Figure 4.4. Additionally, there is a directly proportional relationship between the size of the logic clusters in the chosen MAC unit configuration and learning performance. Depending on the selected multiplier configuration, the size of the logic clusters is determined by the range of accuracy losses, which will detect the number of the required iterations to determine the best weights. As a result, the SDLC configuration models (i.e. 2-, and 4-bit) have achieved the best weights at the 10th and 18th epochs for the Noisy XOR data-set, and 90th and 100th epochs for the Binary IRIS data-set, after the *Exact* configuration model, as shown in Figure 4.4.

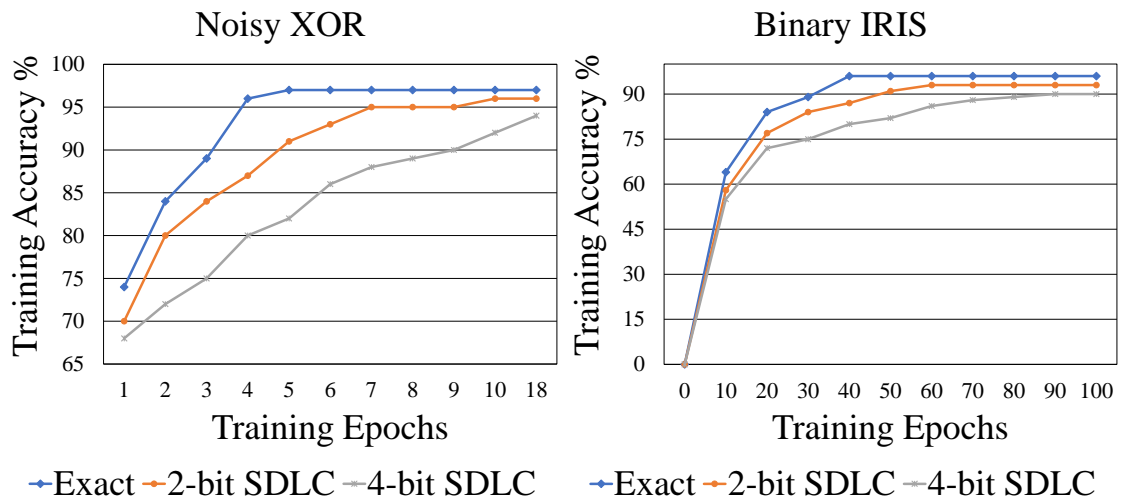


Figure 4.4: Back-propagation learning rule results for various selections of ANN hardware configurations using Noisy XOR and Binary IRIS data-sets.

A similar method is used to generate the weights but with a different number of layers and neurons (see Table 4.1). As shown in Figure 4.5 the *Exact* version records the best at 96%. Conversely, in terms of power, energy and latency, the SDLCs have

higher efficiency (see Table 4.2), at broadly satisfactory levels of inference accuracy within the range of 91% and 89%. Consequently, this design will represent the best power-adaptive hardware design that can operate in conditions of varying power constraints while delivering power efficiency and learning efficacy.

Moving on to the more complex problem of ‘Hand Written Digit Recognition’ using modified national institute of standards and technology (MNIST) [142] data-set, the 4-bit SDLC multiplier’s accuracy reduces to 89%.

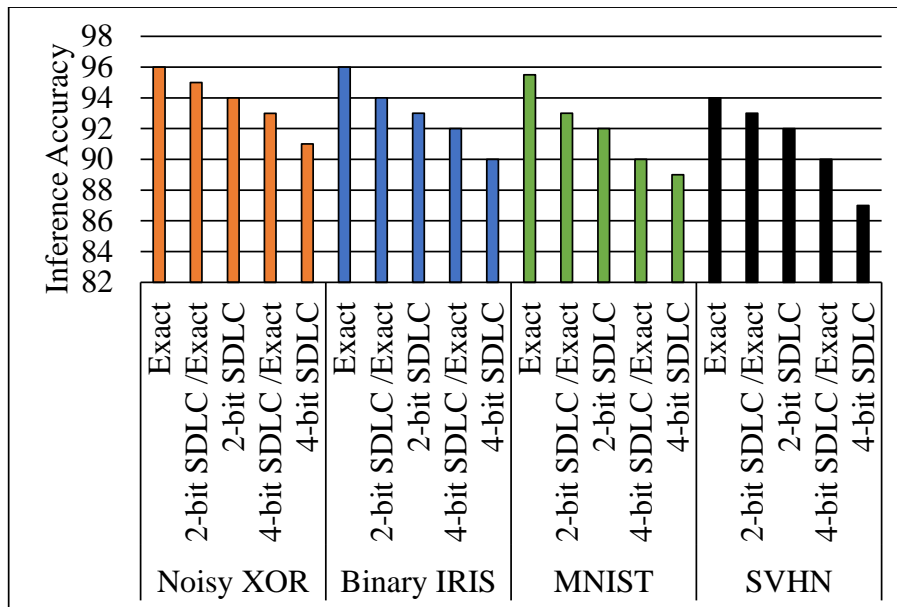


Figure 4.5: Inference accuracy results for various selections of ANN hardware architectures using various data-sets.

The SVHN data-set [143] is considerably more challenging for classification than MNIST with large numbers of inputs and neurons (Table 4.1) and is therefore used next to demonstrate the capabilities of the proposed configurable MAC. The accuracy results are shown in Figure 4.5, with the 4-bit SDLC achieving a respectable 87% accuracy.

The results in Table 4.2 are obtained from Synopsys Design Compiler using the CMOS 90-nm library for ASIC implementations of ANNs using the proposed configurable MAC unit with hidden layer focusing on h_a , and o_a .

Table 4.2 lists the results of area, power, delay and Power-Delay Product (PDP) for different configurations of MAC unit focusing on specific configurations. As can be seen, in row 1 of every model of neutral data-sets architecture, the model has significantly higher power consumption and delay, area and energy compared with

Table 4.2: Area (A), power(P), delay(DL) and Power-delay product (PDP) results for different MAC configurations.

Data-set \rightarrow	Noisy XOR			
h_t / y_t	$A(\mu m^2)$	$P(\mu W)$	DL(ns)	$PDP(fJ)$
Exact	1.84E+04	8.11E+2	3.42E+1	2.77E+04
2-bit SDLC/Exact	1.23E+04	5.10E+2	2.80E+1	1.43E+04
2-bit SDLC	1.18E+04	5.07E+2	2.74E+1	1.39E+04
4-bit SDLC/Exact	7.44E+03	4.37E+2	1.88E+1	8.22E+03
4-bit SDLC	6.52E+03	3.25E+2	1.74E+1	5.66E+03
Data-set \rightarrow	Binary IRIS			
h_t / y_t	$A(\mu m^2)$	$P(\mu W)$	DL(ns)	$PDP(fJ)$
Exact	1.46E+05	6.43E+03	2.71E+2	1.74E+06
2-bit SDLC/Exact	9.47E+04	4.09E+03	2.19E+2	8.96E+05
2-bit SDLC	9.32E+04	4.02E+03	2.17E+2	8.72E+05
4-bit SDLC /Exact	5.44E+04	2.69E+03	1.43E+2	3.85E+05
4-bit SDLC	5.17E+04	2.53E+03	1.37E+2	3.47E+05
Data-set \rightarrow	MNIST			
h_t / y_t	$A(\mu m^2)$	$P(\mu W)$	DL(ns)	$PDP(fJ)$
Exact	9.35E+05	4.12E+04	1.74E+3	7.17E+07
2-bit SDLC /Exact	6.02E+05	2.60E+04	1.40E+3	3.64E+07
2-bit SDLC	5.97E+05	2.57E+04	1.36E+3	3.50E+07
4-bit SDLC/Exact	3.40E+05	1.69E+04	9.04E+2	1.53E+07
4-bit SDLC	3.31E+05	1.65E+04	8.91E+2	1.47E+07
Data-set \rightarrow	SVHN			
h_t / y_t	$A(\mu m^2)$	$P(\mu W)$	DL(ns)	$PDP(fJ)$
Exact	2.21E+06	9.74E+04	4.74E+3	4.00E+08
2-bit SDLC /Exact	1.47E+06	6.14E+04	3.30E+3	2.02E+08
2-bit SDLC	141E+06	6.12E+04	3.36E+3	2.01E+08
4-bit SDLC/Exact	8.48E+05	4.00E+04	2.12E+3	8.47E+07
4-bit SDLC	7.82E+05	3.97E+04	2.8E+3	8.36E+07

the other models. This is because, at that stage, the model has operated using only the *Exact* model in all the design blocks. In relation to the stages between rows 2-5 when traditional complex *Exact* blocks are replaced by low-complexity design ones, the power consumption and delay are reduced depending on the type of MAC configuration using the approximation level. Therefore, row 5, where the model includes the 4-bit SDLC model and the highest approximation and low complexity, is the optimized model chosen for energy and power efficiency and low delay (decreasing in area complexity and hence in power, energy and delay).

The accuracy of this model is good compared with that of the *Exact* model in row 1. The configuration in Table 4.2 h_t/y_t is selected as the higher approximation

for the hidden layer and not the output layer because it directly affects the output (see Figure 4.5). The results show that up to more than 5.1 times higher (expressed in terms of power-delay product (PDP)[147]) that is calculated to make a clearer assessment of the energy efficiency achieved in the case of the 4-bit SDLC MAC, suggesting that this SDLC version is the most efficient in terms of power, energy, and delay, with a comparatively moderate loss of accuracy.

4.4.1 Area Reduction & Inference Accuracy in FPGA Implementations

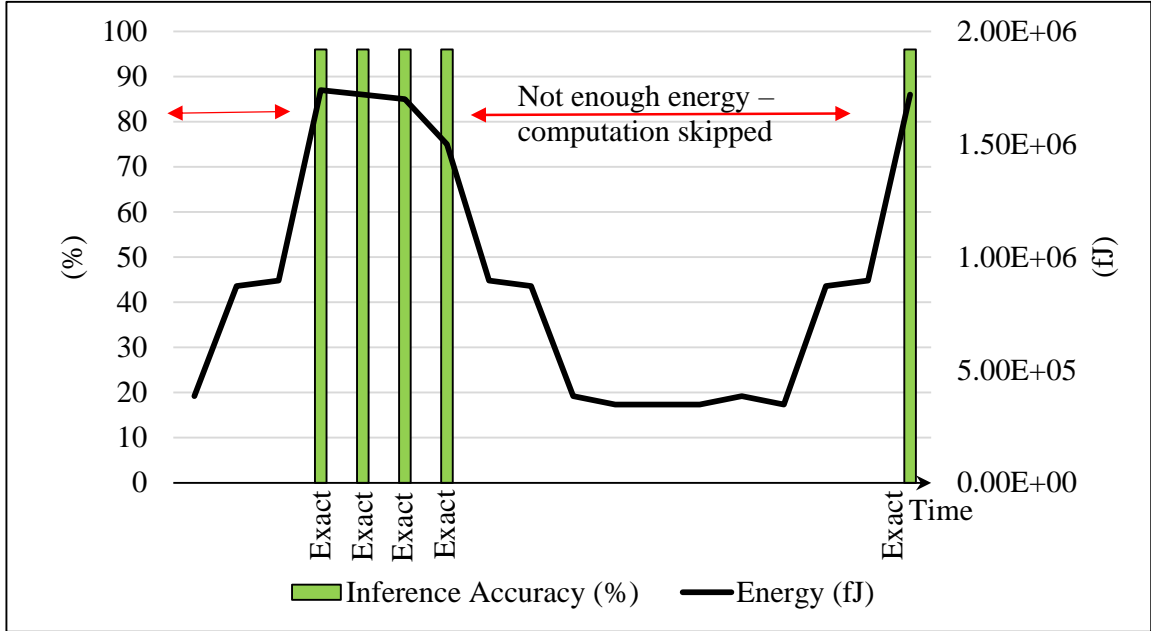
To compare with existing designs the configurable MAC and Ullah, S et al [148] designs were re-implemented in FPGA using Xilinx Vivado Design Suite and Ultra96-V2 platform [122] for fair comparisons. For validation experiments, the proposed MAC performance is examined using the MNIST database and the LeNet-5 convolutional neural network (CNN). The architecture of LetNet-5 [149] consists of two convolution layers, two max-pooling layers, and two fully connected layers. The CNN application was created using the Pythirch framework [150] for the training module. The CNN is trained with 128 batch size and 20 epochs. Then the CNN is implemented in System-Verilog with the best weight and biases achieved from the training stage. To perform the inference, the 4-bit SDLC approximate and Exact MACs are used. Since the convolution layer requires the most computation, the Exact is replaced with the 4-bit SDLC MAC. The proposed 4-bit SDLC MAC configuration achieves 91% inference accuracy according to Table 4.3 that compares the Inference Accuracy and convolutional layer’s area reduction of two multipliers (Exact and [148] design). The proposed 4-bit SDLC approximate MAC configuration obtains a negligible loss in output quality with the least area when compared to the existing approximate MAC design.

Table 4.3: Comparing with Ullah, S et al [148] in terms of Area reduction and Inference accuracy.

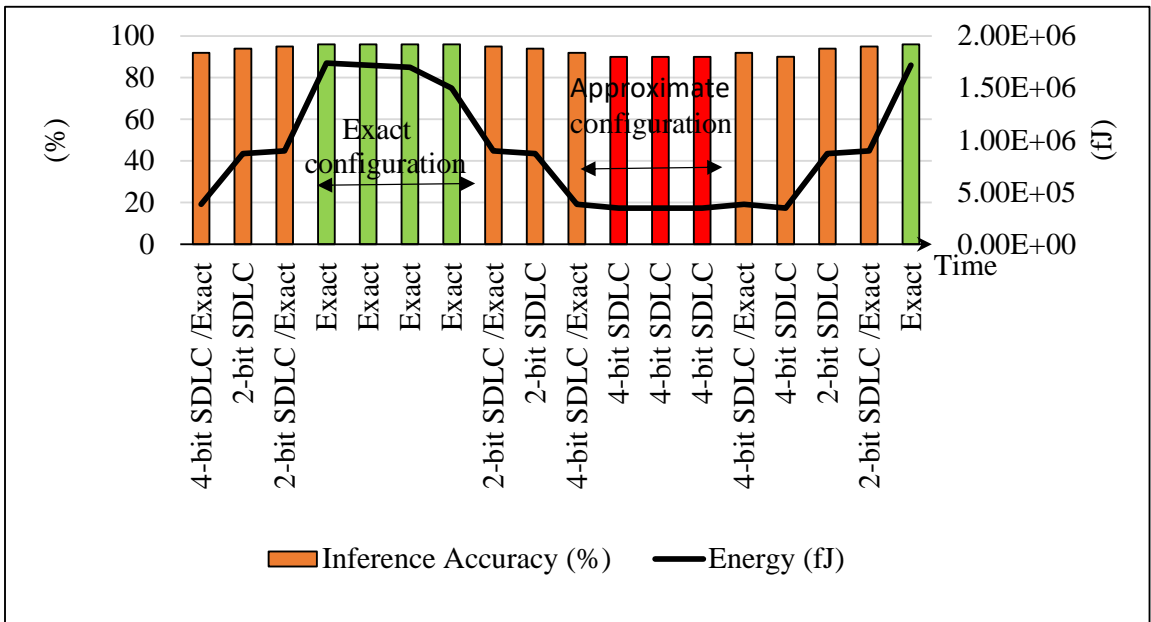
Data-set →	MNIST	
	MAC ↓	
Proposed Exact	-	98.2
<i>Acc_{app}</i> [148]	4.02	64.1
Proposed 4-bit SDLC	45.2	91

4.5 Case Study

In this section a case study is presented where the configurable MAC is controlled by the EACA algorithm (Algorithm 1) solving real-world problems such as continuing



(a)



(b)

Figure 4.6: Different run-time scenarios for the EACA model to operate under highly variable energy conditions. (a) only *exact* configuration allowed and (b) all configurations allowed.

to provide a required computation capacity under variable energy supply conditions. The case study demonstrates the capabilities of the proposed configurable multiplier as well as the validity of the EACA configuration selection algorithm.

In this investigation, two different MAC configuration systems are compared. The first case is presented in Figure 4.6(a) where only the *exact* MAC configuration is allowed. The EACA algorithm's response to the energy variations is recorded and shown in the figure. Also recorded and exhibited in the figure are the accuracy of the ANN (vertical bars) and the instantaneous energy availability (black curve). The EACA runs the *exact* configuration only when the energy is sufficient for that configuration. When the energy is not enough the EACA waits, until more energy becomes available to restart. The second case is shown in Figure 4.6(b) where all configurations are allowed. The EACA attempts to fit the least approximate configuration to the instantaneous available energy. When the energy is low, more approximate configurations are chosen to maintain processing; when it is high, less approximate configurations are chosen for superior accuracy. Case 1 mimics a conventional exact-only system, which has poor adaptability under variable energy supply.

4.6 Conclusion

In this chapter presents a new method for power-adaptive ML hardware design. Firstly, ability of the significant-driven logic compression (SDL) multiplier approach is taken advantage of to design a configurable MAC unit. Secondly, the EACA model was employed to allocate the optimal configuration of the MAC unit depending on the available instantaneous energy. The proposed configurable MAC designs were implemented in System-Verilog and synthesised using Synopsys Design Compiler. The synthesised results explained that up to $5\times$ higher (expressed in terms of PDP) energy savings can be achieved. Thirdly, the proposed MAC is utilised to operate into neuron module targeting ML systems. The EACA model proved capable of tuning the required computation capability of the neuron under variable power budgets. Then, the presented neuron module was scaled up to build the proposed model-driven power-adaptive ANN design. The results reveal that the proposed approach can provide significant energy and area savings with negligible loss in output quality (the worst inference accuracy is 91% for the 4-bit SDL MAC) compared to an existing work. Moreover, the results demonstrate that the EACA model allows the ANN to operate at run-time under significant energy variations while meeting the quality requirements.

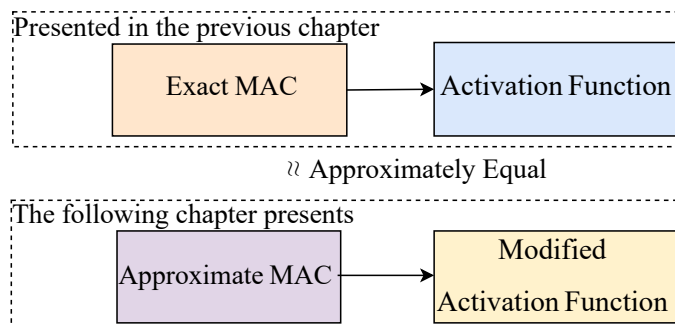
Chapter 5

NN Design with Modified Activation Function

In chapter 4, a new method including a configurable MAC unit for the design of power-adaptive ML hardware was introduced. This chapter introduces a configurable modified activation function in order to minimize prediction error caused by the approximate configurations as mentioned in chapter 4. The estimation of the accuracy of approximation by these methods of activation functions is performed. To evaluate the effectiveness of modified activation function, a new neuron design which includes a configurable MAC (exact and 4-bit SDLC) and the modified activation function are designed in System-Verilog and used to generate synthesizable modules. The synthesized designs are further characterized for power, area, delay, and energy using Synopsys Design Compiler, employing UMC 90nm digital CMOS technology and the Xilinx Vivado Design Suite. Also, an extensive error analysis is provided.

5.1 Introduction

A significant part of neuron implementation is the development of activation function hardware. Sigmoid and rectified linear unit (ReLU) activation functions are the most frequently used activation functions in backpropagation neural networks (NNs) [151]–[154]. The NN implementations consist of the modular electronic neurons that primarily feature arithmetic operations such as MAC units. The arithmetic complexity increases with more inputs as well as the learning problem at hand. Intrinsically, achieving the required performance under restricted power or energy budget remains challenging [131], [133]. This chapter presents a new configurable MAC unit taking advantage of the proposed MAC unit presented in chapter 4. The architecture leverages between the Exact and approximate configurations to specify the optimize configuration depending on the model-driven power and accuracy trade-offs (see Subsection 5.6.4). Thus, by allocating the appropriate configurations during run-time, the computation capability under variable power or energy budgets can be adjusted. Furthermore, a configurable modified activation function to minimize the prediction error was proposed (see the following Figure).



5.1.1 Contributions

This chapter presents a new configurable modified activation function, which supports run-time configuration multiplier. The architecture leverages a tunable approximation method to specify the logic compression level depending on the model-driven energy and accuracy trade-offs. Thus, by allocating the appropriate configurations during run-time, the computation capability under variable power or energy budgets can be adjusted the rate of errors in multiplication can be corrected. In this chapter, the following *contributions* are made:

- Propose a new neuron design with a configurable modified activation function;

- implement the proposed new neuron architecture in a MLP and validate it by exercising a data-set to evaluate the performance-energy-quality trade-offs.

The remainder of the chapter is organised as follows: Section 5.4 explains in detail the proposed design method of configurable multiply-accumulate (MAC) unit and modified activation function. Section 5.5 presents the synthesised results, whilst section 5.6 demonstrates the simulation and evaluation results. Finally, section 5.7 concludes the chapter.

5.2 Methods of Activation Functions

An important component is the nonlinear activation function, which is used at the output of every neuron. Several different activation functions are available today, such as sigmoid [155], hyperbolic tangent (tanh)[156], and Rectified Linear Unit (ReLU) [157], along with Leaky ReLU and Maxout. Activations are typically represented by a number within the range of 0 to 1. The weight is double the activation function. The most frequently used activation function in backpropagation NNs is the sigmoid function. It can be described using equation (5.1) [158].

$$sig(x) = \frac{1}{1 + e^{-x}} \quad (5.1)$$

5.3 Proposed Adaptive Approximation Method

The configurable MAC unit presented in chapter 4 is redesigned to have the largest size of the d-bit SDLC approach using the exact area as the core part of a neuron module. In this work the Wallace tree serves as an example. The proposed method can be applied based on any exact multiplier design including multi-stage tree structures, for instance Wallace and Dadda trees. Fundamentally, adders in the multiplier acting as the exact configuration are re-purposed through re-wiring to implement the approximate configurations. Figure 5.1 illustrates how the proposed configurable (8 x 8) multiplier design performs the exact and approximate multiplication. In this design, each circle represents one partial product bit. The required half adders are marked by rectangles spanning columns of two partial product bits and full adders are marked by rectangles spanning columns of three bits. The exact configuration in (a) represents a traditional Wallace tree multiplier which uses the largest number of half and full adder units within the reduction stages. The 4-bit SDLC configuration presented in (b) is considerably smaller, while (c) reveals the new configurable design that includes both the exact and 4-bit SDLC using few

additional numbers of adders in addition to the already existing adders required by the exact configuration.

The neuron module shown in Figure 5.2 accepts the weights and input pairs, where each pair is multiplied together. For each input pair, the multiplication result is sequentially accumulated by the CLA. The final sum is passed through the activation function to produce the neuron's output. A 2x1 multiplexer (MUX) is added for the configurable selection of the logic compression level performed by the MAC unit.

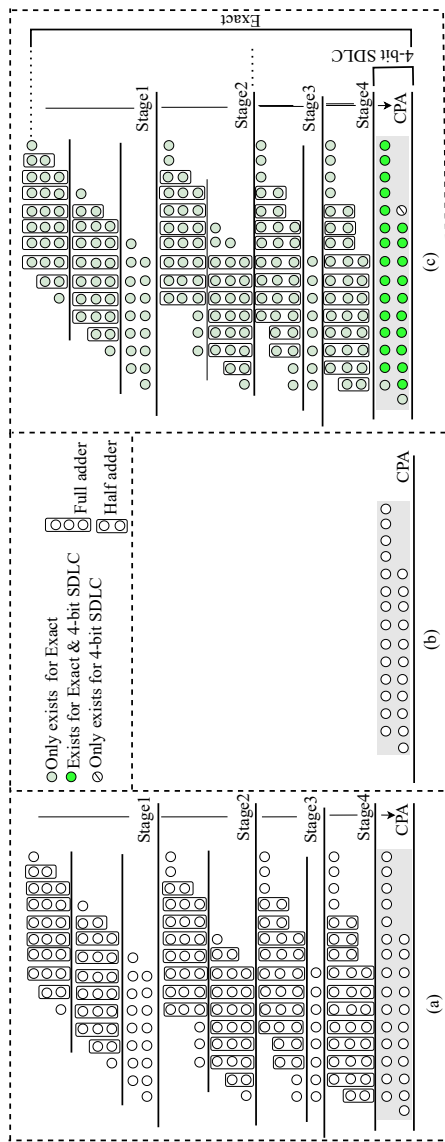


Figure 5.1: The reduction stages of an (8×8) Wallace tree multiplication illustrate the accumulation method for the PPM formed from the exact and logic clusters of two different sizes (a) four reduction stages are required in case of (8×8) traditional Wallace tree multiplier(WTM); (b) no reduction stages for the 4-bit SDLC as the height of the PPM is only two rows, and (c) configurable (8×8) Wallace tree multiplication.

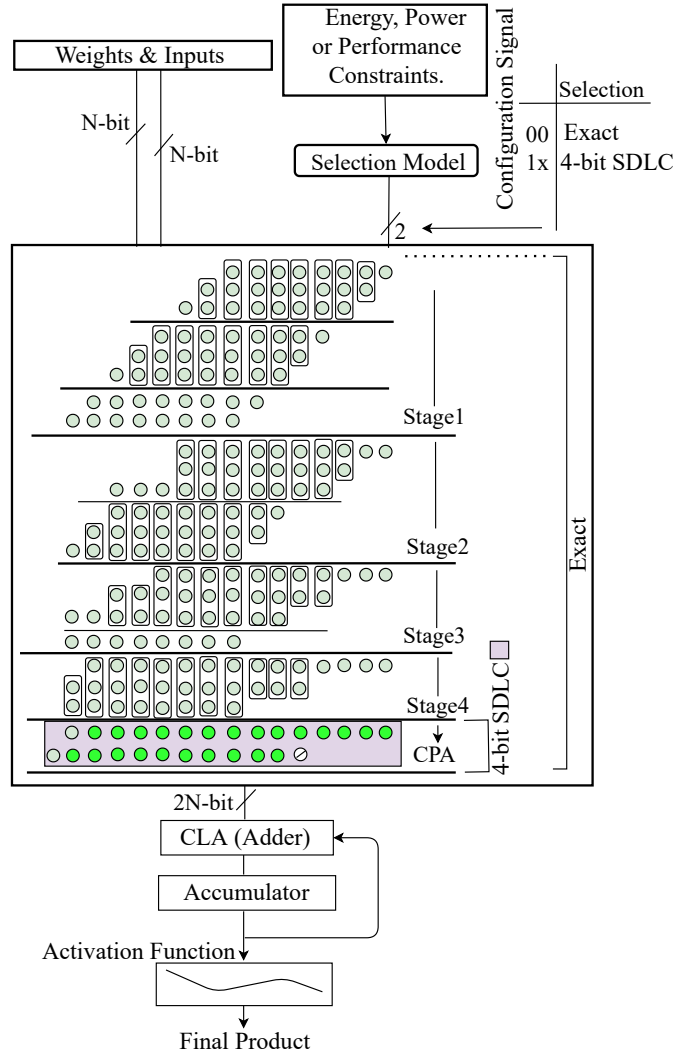


Figure 5.2: Neuron (structure-serial) processing including the proposed hardware architecture of the configurable MAC unit.

Table 5.1: The number of full and half adders used by the accumulation stages and CPA for different exact, approximate multiplier and the Proposed configurable (8×8) Wallace tree.

Multiplier \rightarrow	Exact		4-bit SDLC		Proposed	
Adder type	HA	FA	HA	FA	HA	FA
Stage 1	4	12	-	-	4	12
Stage 2	3	13	-	-	3	13
Stage 3	4	6	-	-	4	6
Stage 4	4	7	-	-	4	7
CPA	10		11		11	

Table 5.1 may be read in conjunction with Figure 5.1 to represent the required single-bit adders in all parts of the proposed configurable multiplier design. As can be noticed from the table, the exact configuration requires the majority of half and full adders. Only a very small number of adders are required in addition to the exact configuration to accommodate the 4-bit SDLC configuration. In Section 5.6.4 the additional silicon area required by the 4-bit SDLC is calculated.

5.4 Proposed Configurable Modified Activation Function

The proposed research begins with the following hypothesis (5.2). For every activation function, AF_e can be used to construct a NN Ne with (Me) . Let's represent this by $NeMe, AF_e$. An activation function exists (AF_a) such that $Na=NaMa$, AF_a is functionally the same as or approximately the same as (Ne) . This concept can be represented by:

$$Na \{Ma, AF_a\} \approx Ne \{Me, AF_e\} \quad (5.2)$$

where (Na) is a neuron that includes (Me) , the exact multiplier for weights and inputs multiplications and (AF_e) is the exact activation function.

Table 5.3 shows the multiplier's accuracy factors. The error distance (ED) parameter is a metric for assessing output quality but is not illustratively compared to other metrics such as the error rate (ER), mean relative error distance (MRED), and normalised mean error distance (NMED). These metrics (ER, MRED and NMED) are considered to evaluate the output quality of the approximate circuits in error resilient applications [159]–[162]. It is worth noting that in many imprecise applications, such as those focusing on bit Significance-Driven Logic Compression, the difference between the exact and inaccurate results is more important than their relative difference. The MRED and NMED are considered to be the best metrics for these applications [102]. Moreover, in these applications, it is necessary that the multipliers comprising different sizes are compared. Accordingly, in these cases the

Table 5.2: Normalised Mean Error Distance (NMED) for (8x8) multiplier size with a different d-bit.

d-bit↓	NMED (%)
2-bit	0.0035
3-bit	0.0101
4-bit	0.0327

NMED metric would be more illustrative [102].

Table 5.3: Error Distance (ED), Error Rate (ER), Mean Relative Error Distance (MRED), Normalised Mean Error Distance (NMED), are different metrics used to evaluate a multiplier.

Metrics	Error Metric	Approximate circuits	Bit SDLC	Comparing multipliers of different sizes
ED	[✓]	[×]	[×]	[×]
ER	[✓]	[✓]	[×]	[×]
MRED	[✓]	[✓]	[✓]	[×]
NMED	[✓]	[✓]	[✓]	[✓]

However, the proposed neuron model including (Configurable MAC unit and modified activation function) is built based on the NMED as a metric that is considered to be the best error metric as previously mentioned (see Table 5.3). To estimate the accuracy of the approximate multiplier configuration (4-bit SDLC), comprehensive simulations are performed using Vivado and MATLAB simulation (see Figure 5.3) by implementing a functional model the proposed MAC unit. The response of the approximate multiplier configuration is evaluated for all possible combinations of operands (see Table 5.2). The traditional sigmoid function described in equation (5.1). Equation (5.4) shows the modified sigmoid function, to which the variable CM was added. The CM is the compensation described in Equation 5.3 to be equal to the NMED in Table 5.2 that will be used to minimize the prediction error that caused by the approximate multiplier configuration (4-bit SDLC). In addition, the focus has been on the higher level SDLC multipliers where energy reductions are achieved at the cost of reduced quality (e.g., 4-bit SDLC in 8x8 multiplier size). In essence, the same approach is used in the ReLU activation function. Figure 5.4 represents a single neuron architecture with the activation function shown in (a) and the proposed modified activation function shown in (b) including the variable CM and the CLA adder. The additional overhead silicon area caused by the registers and adder in Figure 5.4(b) is calculated in Subsection 5.6.3.

$$CM = NMED \quad (5.3)$$

In fact, by performing simple transformations for 5.1 and 5.3, is obtained

$$sig(x) = \left(\frac{1}{1 + e^{-x}}\right) + CM \quad (5.4)$$

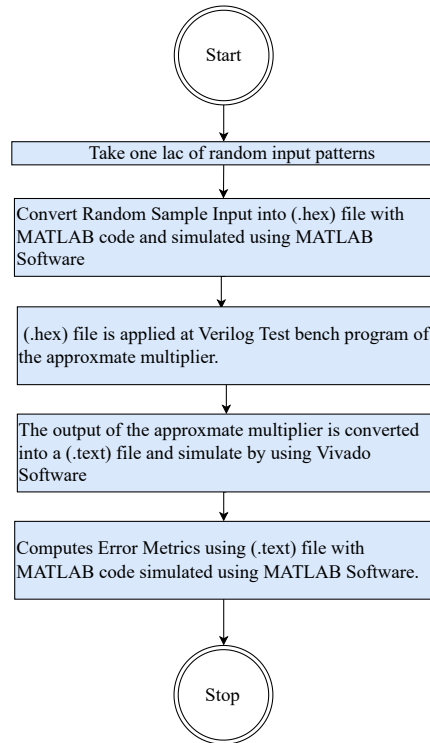


Figure 5.3: Simulation process of error metrics calculation.

5.5 Experimental Results

To demonstrate the proposed approach, the neuron module is described in System-Verilog to implement the MLP. Then, neuron modules were synthesised with different configurations using Synopsys Design Compiler and the circuits were mapped to the Faraday’s 90nm technology library to measure the power consumption, silicon area and the performance profiles related to each configuration. These measurements are then modelled within the EACA model (see Chapter 3 (Section 3.6.1)) to allocate the optimal design requirements for the MLP. Next, the proposed neuron is implemented on an embedded FPGA (Xilinx Ultra96 v2) Evaluation Platform using Xilinx Vivado Design Suite. The following subsections list the synthesised result of different configuration of the proposed neuron including MAC unit module and a case study for the neuron module learning algorithm.

5.5.1 Area, Delay & Power Trade-offs in ASIC Implementations

For ASIC comparisons, a generic System-Verilog code is used to generate synthesisable modules for the proposed configurable MAC. Synopsys Design Compiler is applied to synthesise the MAC configurations and the circuits are implemented in

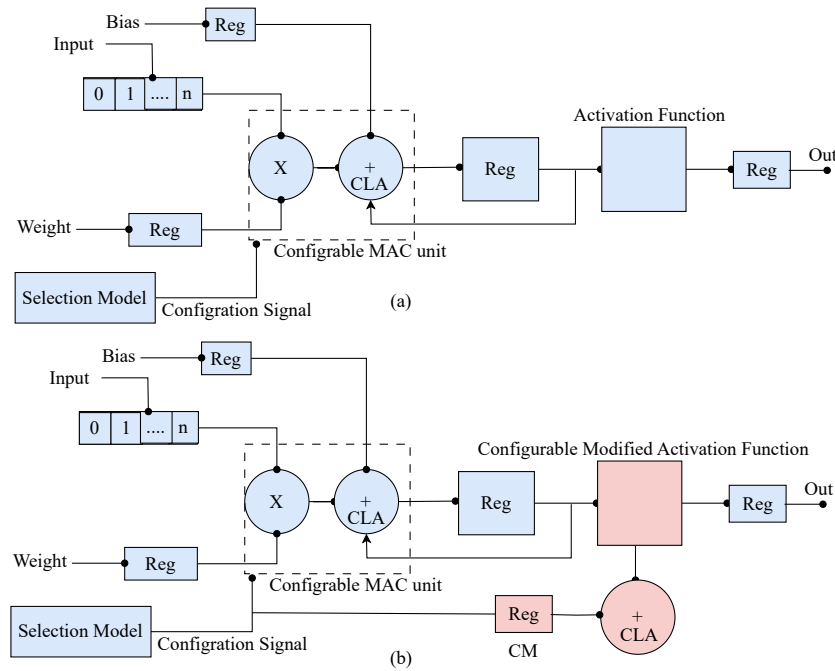


Figure 5.4: A schematic representation of a single neuron architecture with the activation function is shown (a) and the proposed modified activation function is shown in (b).

the Faraday 90nm technology library.

The compared methods [102], [163] and [25] are re-implemented using the same technology library (Faraday 90nm technology library), the Synopsys Design Compiler synthesis tool and the library setup time is 9.93ns for the operation speed. Thus, comparisons can be performed on the same implementation technology node and library. Table 5.4 presents area, delay and power trade-off figures when compared with [102], [163] and [25]. As can be realised, the configurable hardware (row 1) is larger in terms of area than the exact multiplier [25] alone. A delay overhead is also expected in the configurable design because of the increased number of adders.

Table 5.4 shows that the area overhead that compares the configurable multiplier Proposed Exact to the Exact multiplier on which it is based (Exact [25]), is 0.12% increase of $62.42 \text{ } \mu\text{m}^2$. Power overhead is 0.14%, the latency overhead is 0.3% whereas the PDP overhead is 0.6% increase of 164.2 fJ Exact [25]. However, the proposed model saves more than 59.4% of area compared with the solution proposed in [102], which includes separate Exact, 2-bit and 4-bit SDLC multipliers for runtime the selection. This large reduction in area implies significant power and energy savings. The competitive figures obtained for these non-functional metrics suggest

Table 5.4: comparing existing multiplier designs and the proposed configurable design in terms of power(P), area (A) delay(DL) and Power-delay product (PDP).

Multiplier ↓	$P(\mu W)$	$A(\mu m^2)$	DL (ns)	$PDP(fJ)$
Fixed Config(s) [102]	158.39	3495.71	7.82	1238.6
Proposed Exact [163]	66.20	1450.40	2.66	176.1
Exact [25]	62.42	1417.47	2.64	164.2
Proposed Exact	62.50	1419.20	2.65	165.3
Proposed 4-bit SDLC	25.42	501.37	1.35	34.32

that this configurable design would also compare favourably against the other designs in [25], [163] and [102].

5.5.2 Area, Delay & Power Trade-offs in FPGA Implementations

For a more flexible design, the configurable multiplier is also implemented in FPGA using Xilinx Vivado Design Suite for the Ultra96-V2 platform [122]. The compared existing designs, originally also on FPGA, are re-implemented on this same platform for fair comparisons. A previous study includes different designs of exact multipliers on FPGA (see Table 5.5), focusing on performance [164] and [163]. These are compared with the two proposed configurations for non-functional parameters. Table 5.5 lists the results for each design in relation to area, delay and power. It can be observed that the proposed configurable multiplier, in its different configurations (between exact and 4-bit), is competitive in terms of area, delay and power compared to modified radix2 booth multiplier (MRBM) and WTM. It is notable that the exact configuration is competitive in delay with these multipliers, which were designed for speed.

Table 5.5: Comparing non-functional metrics with other approaches.

Multiplier ↓	Area (LUT's)	Power (W)	Delay (ns)
MRBM [164]	137	1.010	6.721
WTM [164]	96	1.061	6.102
Proposed Exact [163]	91	1.22	6.102
Proposed Exact	89	1.18	6.101
Proposed 4-bit SDLC	42	0.23	3.8

5.5.3 Neuron Module Learning

In this section, a case study for the perceptron learning algorithm is presented, in which a training set is utilised to train the perceptron to classify inputs correctly.

This is achieved by adjusting the connecting weights and the bias to precisely handle linearly separable sets. Figure 5.5 shows a test platform to evaluate the effectiveness of the proposed multipliers on perceptron-based machine learning application. All input sets are randomly generated and independently distributed from 0 to 255 to allow exact or approximate 8-bit multiplication to be employed. Then, the classifier is evaluated against a test set of 1000 two-dimensional points that belong to two classes [0,1].

The exact and the approximate multiplier 4-bit SDLC multiply the perceptron inputs by the weight vectors. After that, the proposed modified sigmoid function was used to generate the final output. The error rate (ER) is the mismatch ratio between the classified class and the actual output. Table 5.6 and Figure 5.6 are demonstrate the comparison of the classification problem with the exact (8×8) multiplier and exact activation function in (a). In (b), the approximate 4-bit SDLC proposed in [102] and an exact activation function were used. However, in (c), the proposed 4-bit SDLC was applied with design that is proposed in [102] with the proposed configurable activation function. The results reveal that in (c) compared to (a), where the exact multiplier and exact activation function were used, the proposed SDLC multiplier and configurable activation function (c) mismatch only two points from the 1000 points in the testing set, as class 1 by mistake. Note that even the design applying exact multiplier and activation function (a) cannot classify all points correctly (one mismatched point due to the random initialisation of the weights).

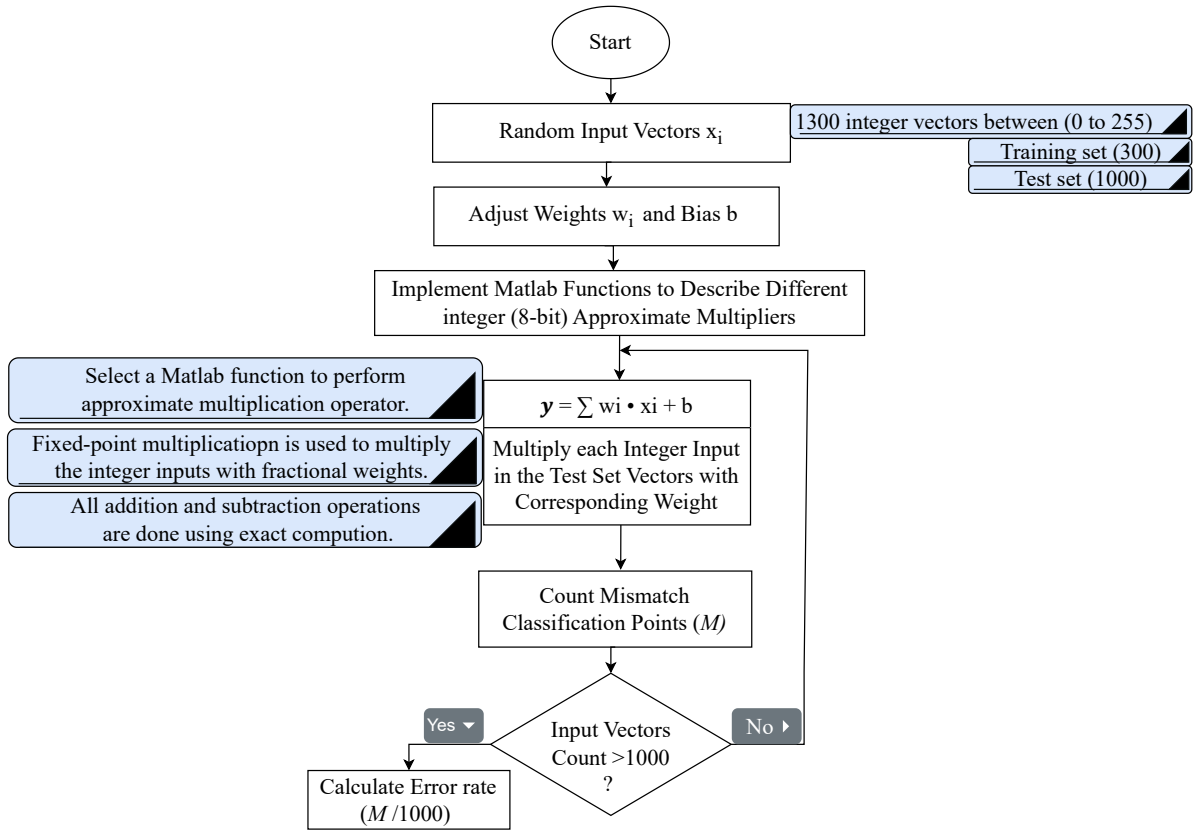


Figure 5.5: Flowchart diagram demonstrating the main steps for evaluating the impact of the proposed MAC unit with modified activation function and on a perception based Classifier.

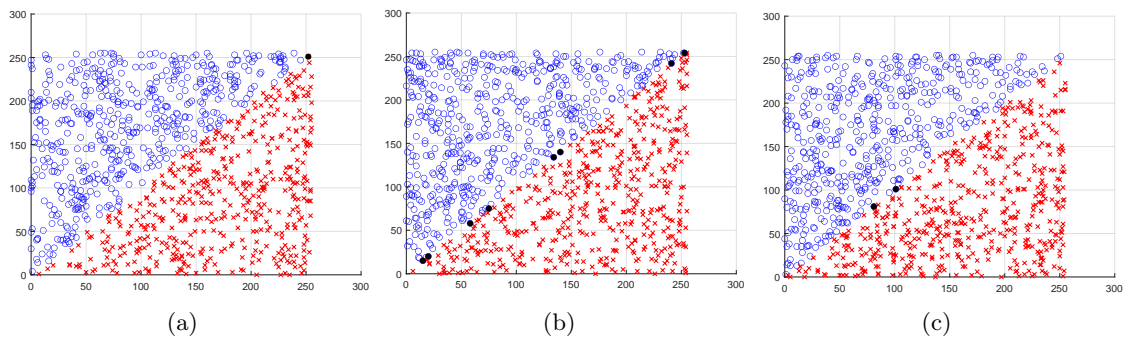


Figure 5.6: The test set perceptron classification using; (a) exact multiplier and exact activation function ; (b) 4-bit SDLC multiplier in [102] and exact activation function; (c) proposed 4-bit SDLC multiplier and a configurable activation function (blue and red points represent two classes 0 and 1, black dots mismatch classification points and the axes show the random inputs between 0 to 255 for 8x8 multiplier).

The proposed approach outperforms the other designs and can provide acceptable error rates (considering 10% average relative error as an acceptable accuracy metric for all applications, verified by [165]) compared with the proposed design in [102] illustrated in (b), which recorded eight mismatch points.

Table 5.6: ER results for different designs (a),(b) and (c).

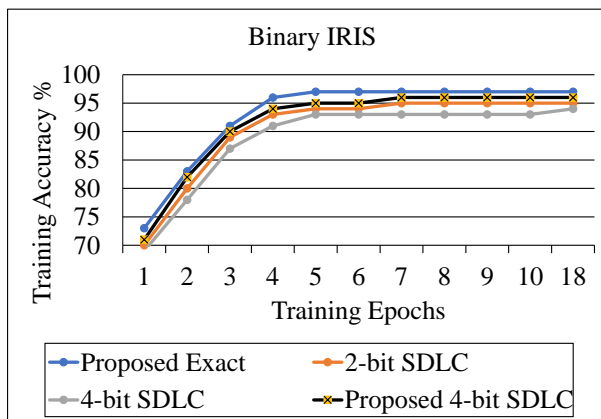
VERSION	ER (%)
Exact	0.01
4-bit SDLC [102]	0.08
Proposed 4-bit SDLC	0.02

5.6 Case Studies

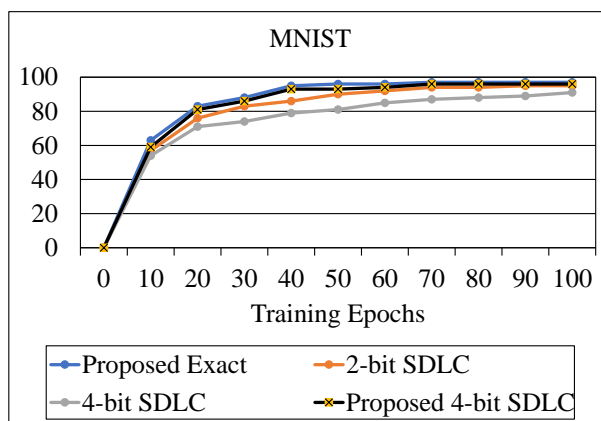
To demonstrate the proposed neural model presented in section 5.3, the neuron model was scaled up to implement the MLP. Next, the MLP was synthesised with different configurations using Synopsys Design Compiler to measure the power consumption, silicon area and the performance profiles related to each configuration. In this section six case studies are set up. The first and second case studies exhibit the learning and inference accuracy results for various selections of MLP configurations using different MAC configurations with fixed and modified activation functions in several well-known data-sets. The third case study compares the area overhead between the fixed and modified activation function. The fourth case study evaluates different scenarios for power variation using MLP. The fifth case study takes advantage of the reconfigurability and focuses on specific blocks in the MLP architecture. These case studies demonstrate the capabilities of the proposed configurable MAC with a modified activation function as well as the validity of the configuration selection algorithm.

5.6.1 Data Classification Results using MLP

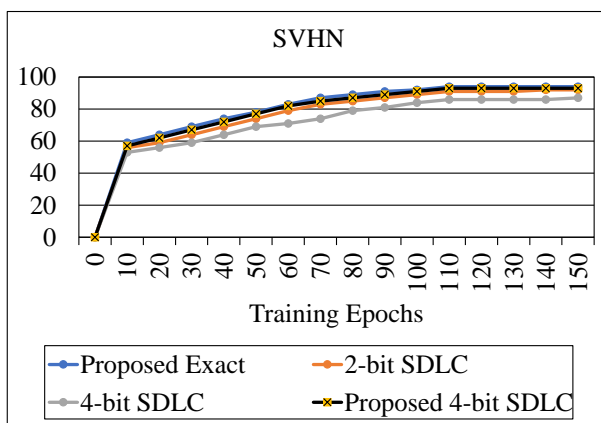
Using the neuron module (Figure 5.2), MLP are implemented targeting well-known ML data-set. To demonstrate the proposed approach, the neuron module is described in System-Verilog to implement the MLP. The neuron modules are then synthesised with different configurations using Synopsys Design Compiler with the circuits mapped onto the Faraday 90nm technology library to measure the power consumption, silicon area and the performance profiles related to each configuration. These measurements are then modelled within the EACA model to allocate the optimal design requirements for the MLP.



(a)

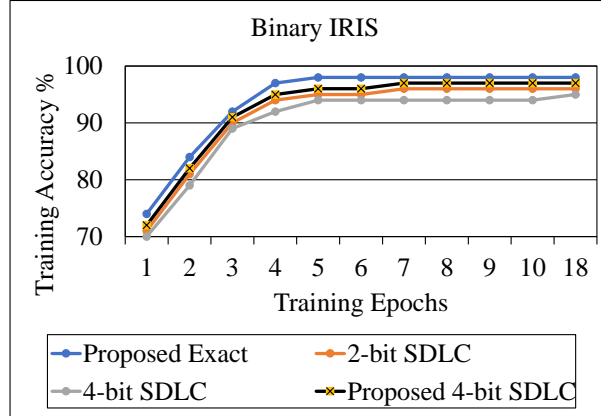


(b)

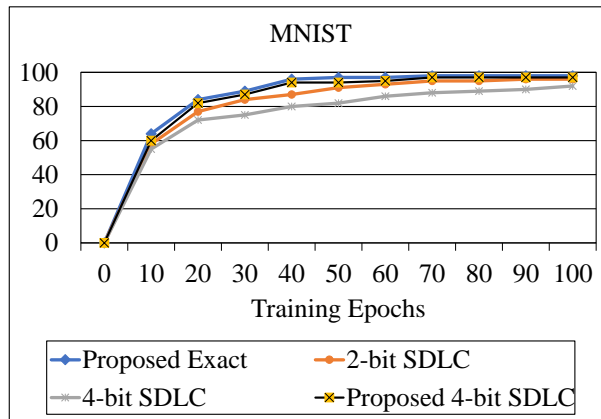


(c)

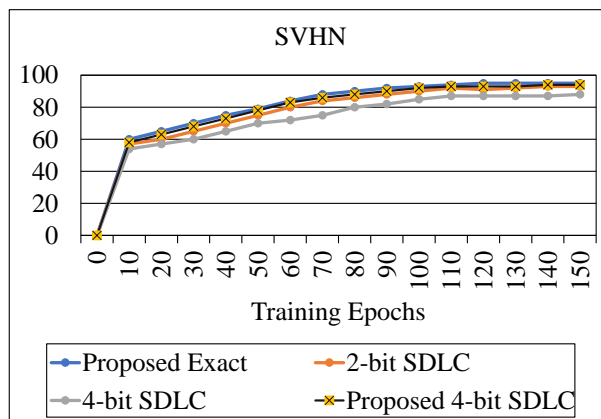
Figure 5.7: Backpropagation learning rule results for various selections of ANN hardware configurations using different MAC configurations with the modified Sigmoid activation function in several well-known data-sets.



(a)



(b)



(c)

Figure 5.8: Backpropagation learning rule results for various selections of ANN hardware configurations using different MAC configurations with the modified ReLU activation function in several well-known data-sets.

The binary IRIS data-set is an ANN architecture that has been implemented. Observed in [145] and [146], the proposed ANN consists of two hidden layers. After the ANN circuit is implemented, it is trained using 60% Binary IRIS data-set including the back-propagation learning algorithm by working offline to calculate the best weights. Subsequently, the remainder of the data-set is applied to test the data-sets after adding a range of possible formulations that are the different MAC unit configurations with the proposed activation function. In terms of learning, it is evident that the best weights were achieved at the 7th iteration for the Binary IRIS data-set using Exact MAC with fixed activation function. The proposed 4-bit SDLC achieved the best weights at the 8th iteration applying the new modified Sigmoid function (see Figure 5.7 (a)). Additionally, for The proposal concerning the ReLU activation function the best weights were recorded with a smaller number of iterations at the 6th iteration using Exact MAC and at the 7th iteration with the proposed 4-bit SDLC (see Figure 5.8 (a)).

To demonstrate more flexibility, the more complex problem of ‘Hand Written Digit Recognition’ is addressed next. The MNIST data-set [142] is different ANN architecture that has been implemented. A three-layer MLP can achieve 95.5% accuracy [166] on the 784-input, 10-output problem utilising 128 neurons in the hidden layer. The MNIST has 55,000 training images divided into two sets, one set of 45,000 images are used as training data with the backpropagation learning algorithm by working offline to calculate the best weights. The other set of 10,000 images are employed to validation data. 100 is used to train the model with the training images, based on different learning rates. Next, the 10,000 validation images are used to evaluate the 100 results. Finally, the selected parameters are utilised to test the data-set after adding a range of possible formulations that comprise the different MAC unit configurations and activation function. Figures 5.7 and 5.7 (b) exhibit the training results of different MAC configurations and activation functions. It is apparent that the that the proposed 4-bit SDLC with both types of activation function can achieve the best weight more rapidly, in comparison to the fixed SDLC with the fixed activation function. For instance, the newly proposed 4-bit SDLC

Table 5.7: Numbers of input, layers and required neurons for different data-sets.

Data-set	No. of Input	No. of Layers	No. of Neurons
Noisy XOR	12	2	13
Binary IRIS	16	2	103
MNIST	784	3	660
SVHN	1024	6	1560

with the ReLU activation function attains the best weights with fewer numbers of iterations, 70, in total. In contrast the fixed 4-bit SDLC required a greater number of iterations and recorded less learning accuracy at 87% and 91% after the 100th iteration.

The SVHN data-set [143] is considerably more challenging as regards classification than the MNIST with large numbers of inputs and neurons (Table 5.7). Therefore, it is applied next to demonstrate the capabilities of the proposed configurable MAC with various types of activation functions. Training Accuracy results presented in Figures 5.7 and 5.8 (c) record less accuracy due to the increase in the number of approximate neurons (fixed 2, 4-bit SDLC). However, the new proposed 4-bit SDLC achieves the best learning accuracy with 1% difference compared with the exact configuration.

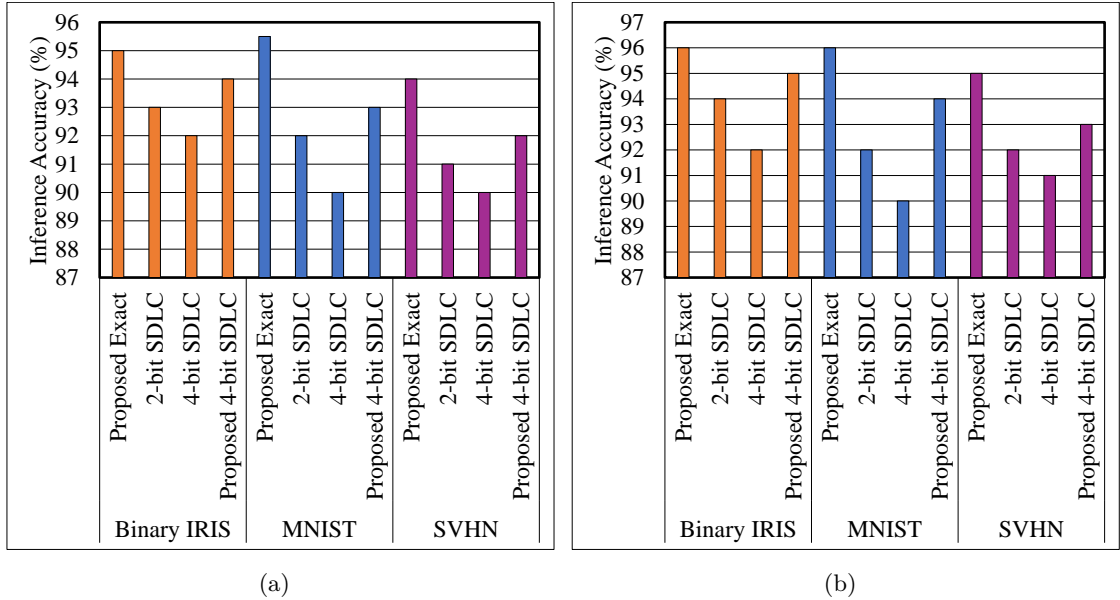


Figure 5.9: Inference accuracy results for various selections of ANN hardware architectures using various data-sets with (a)- Sigmoid and (b)- ReLU activation functions.

5.6.2 Inference Accuracy Results for Different ANN Hardware Configurations with Different Data-sets and Activation Functions

Figure 5.9 presents testing accuracy results for various selections of ANN hardware architectures using various data-sets with (a)- Sigmoid and (b)- ReLU activation functions. It can be noticed that the accuracy of three MLP network architectures using the exact version of MAC and consisting of the ReLU activation function (Figure 5.9 -(b)) records the best at ranges between 95% and 97%, whilst fixed

approximate versions (2,4-bit SDLC) record the best accuracy in the range that is between 87% and 94%. However, the proposed 4-bit SDLC with modified activation functions (Sigmoid and ReLU) including the (CM) value to minimize the prediction error caused by the approximate configurations), records the best accuracy in the range that is between 1 to 2 percentage points less than the exact configuration.

Table 5.8: Area overhead results for the modified Sigmoid activation function ($Modified_{AF}$) including the additional number of adders in hidden layers (h) and output layer (y) compared to the fixed ($Fixed_{AF}$) activation function.

Data-set \rightarrow	MNIST					
	Hidden/Output Layers		Area Per Neuron	Area Per Layer	Total Area(um^2)	
Layers Blocks	h_1	h_2	y	$Fixed_{AF}$	$Modified_{AF}$	
Activation Functions \downarrow				$layer_{(h)}$	$layer_{(y)}$	
$Fixed_{Af}$	500	150	10	132.49	-	86118.5
Additional adders (+)	+ 500	+ 150	+ 10	-	70.2	+ 45630
$Modified_{Af}(Fixed_{Af} + adder)$	500 + 500	150 + 150	10 + 10	-	202.69	131748.5
						2026.9
						133775.4 (40.2%) \uparrow

5.6.3 Modified Activation Function Area Overhead in ASIC Implementations

In this case study, the area overhead of the newly proposed modified activation function has been measured and compared to the fixed activation function. As an initial step, the MLP was implemented with the same architecture used for the MNIST data-set. This is equal to 660 neurons divided into two hidden layers with 650 neurons each and ten neurons assigned for the output layer (see section 5.6.1, Table 5.7). In addition, the Sigmoid activation function was utilised as a case study model. CLA has been chosen as an additional adder to meet the requirements of equation 5.4. Table 5.8 shows the number of neurons in the hidden layers. The first hidden layer (h_1) is equal to 500 neurons, whereas the second hidden layer (h_2) equals 150 neurons. The result illustrates that the overhead silicon area of the proposed activation function is 40.2% by adding a silicon area equal to $70.2 \text{ } \mu\text{m}^2$ in comparison to the fixed activation function of $132.49 \text{ } \mu\text{m}^2$ (see the first row in Table 5.8). The adder (CLA in this case) consumes that additional silicon area and is applied per neuron to modify the fixed activation function. According to Table 5.8, the additional number of adders equals the number of neurons. There are 650 neurons in each of the two hidden layers, requiring the same number of adders (500 for the first and 150 for the second hidden layer). Ten neurons require ten adders in the output layer. Section 5.6.5 presents different MLP architecture scenarios focusing on specific layer blocks and a reduction in power/area/delay/energy when the proposed activation function is added only to the output layer (y).

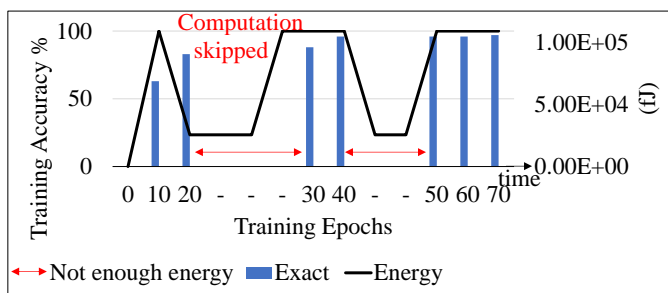
5.6.4 Energy-Aware Variation Scenarios

In this case study, the same MNIST MLP architecture in subsection 5.6.1 and the RELU activation function are used. Additionally, the configurable MAC is controlled by the EACA algorithm presented in [163]. However, in this study, three different MAC configuration setups are compared. The first case is shown in Figure 5.10-(a) where only the *Exact* MAC configuration is allowed. The second case is presented in Figure 5.10-(b) where only approximate configuration (4-bit SDLIC) is permitted. The third case is shown in Figure 5.10-(c) where all configurations are allowed.

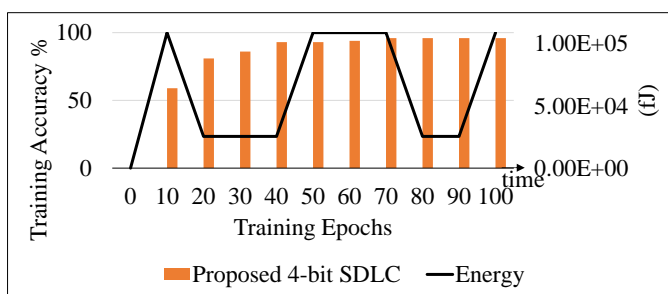
The EACA attempts to fit the least approximate configuration to the instantaneous available energy. The response to the energy variations is recorded and shown in the figure. Also recorded and shown in the figure is the accuracy of the ANN (vertical bars) and the instantaneous energy availability (black curve). EACA runs the *Exact* configuration only when the energy is enough for that configuration.

When the energy is not enough the EACA pauses (see Figure 5.10-(a)), the period between 20th epoch and 30th epoch, until more energy becomes available to resume. As a consequence the *Exact* configuration only record high delay with additional 50 epochs comparing with Figure 5.10-(b).

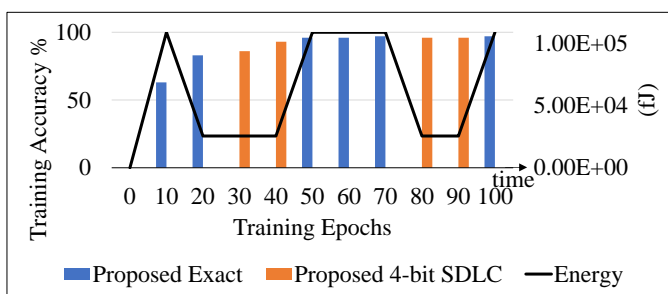
Concerning accuracy and performance, when the energy is low, more approximate configurations are chosen to maintain processing. Moreover, when it is high, less approximate configurations are chosen for improved accuracy. Case 1 (Figure 5.10-



(a)



(b)



(c)

Figure 5.10: Backpropagation learning rule results for various selections of ANN hardware configurations using different MAC configurations with the ReLU activation function in MNIST data-set. Different run-time scenarios for the EACA model to operate under highly variable energy conditions. (a) Only Exact configuration allowed. (b) Approximate (4-bit SDLC) configurations allowed (c) All configurations allowed.

(a)) mimics a conventional Exact-only system, which has poor adaptability under variable energy supply. Case 2 (Figure 5.10-(b)) a conventional approximate-only system records a high performance but less accuracy. Case 3 (Figure 5.10-(c)) demonstrates the ability of the proposed configurable MAC to make best use of an unpredictable energy supply.

5.6.5 Configurable Neuron Architecture Scenarios

This case study focused on the hidden layers (h) and output layer (y) blocks using the same MNIST MLP architecture in section 5.6.2. Table 5.9 lists the results of the area, power, delay, power-delay product, inference accuracy and area overhead for different configurations of the MAC unit and the fixed/the proposed modified activation. As shown by the first row (first configuration), the *Exact* MAC with fixed activation function ($[\times]$) has significantly higher power consumption and delay, area and energy than the other configurations (rows 2 - 4). The reason is that only the *Exact* MAC is used at all the design blocks. In the other rows, on the other hand, the traditional complex *Exact* blocks were replaced with low-complexity design ones (4-bit SDLC/the proposed 4-bit SDLC). Therefore, the second row includes the fixed 4-bit SDLC (the highest approximation) and low complexity with fixed activation function on both the hidden layers (h $[\times]$) and the output layer (y $[\times]$). The third row (third configuration) includes the proposed 4-bit SDLC with a modified activation function on both the hidden layers (h $[\checkmark]$) and output layer (y $[\checkmark]$). The fourth row (fourth configuration) includes the proposed 4-bit SDLC with a modified activation function only at the output layer (y $[\checkmark]$).

In terms of overheads, the third and fourth configurations (third and fourth rows, respectively) to the fixed 4-bit SDLC proposed in [163] have been compared with the fixed activation function on the h and y layers. Table 5.9 shows the additional area overhead caused by the increased silicon (small number of adders) area in the both proposed 4-bit SDLC and the activation function inserted in both the (h) and (y) layers. For this reason, the third row is not the optimised configuration as result of 36.4% of the area overhead compared to the fixed configuration in the second row. Meanwhile, the optimised model chosen for energy, power efficiency and low delay is the fourth configuration (fourth row) where the modified activation has been inserted only in the output layer.

The synthesised results signify that up to 63% of the energy savings were achieved when the fourth configuration (fourth row) was selected compared to the *Exact* model (first configuration) with a low area overhead 0.57% compared to fixed approximate configuration in the second row. In terms of training accuracy, the fourth

configuration model is good compared to the *Exact* model (first row) and the third configuration (third row) where the proposed activation inserted in both layers (h) and (y) target high learning accuracy in a smaller number of epochs (see Figure 5.11). Additionally, concerning Inference Accuracy (IA), the fourth configuration records high accuracy compared to the fixed approximate configuration in the second row (see Table 5.9). The proposed 4-bit SDLC MAC with the modified activation function can significantly save energy and area significantly with negligible loss in output quality, while using it only on the output layer(y).

Table 5.9: Area (A), Power (P), Delay (DL), Power-delay product (PDP), Inference Accuracy (IA) and Area overhead ($Aoverh$) results for different MAC configurations using fixed/modified ReLU activation function in the hidden layers (h) and output layer (y).

Data-set \rightarrow	MNIST									
	h	y	$A(um^2)$	$P(\mu W)$	$DL(ns)$	$PDP(fJ)$	$IA(\%)$	$Aoverh(um^2)$		
Multiplier \downarrow	[\times]	[\times]	9.56E+05	4.14E+04	1.86E+03	7.70E+07	98	-		
Proposed Exact [163]	[\times]	[\times]	3.51E+05	1.69E+04	1.01E+03	1.71E+07	90	-		
Proposed 4-bit SDLC	[\checkmark]	[\checkmark]	4.79E+05	1.96E+04	1.64E+03	2.55E+07	96	1.28E+05	(36.4%)	\uparrow
Proposed 4-bit SDLC	[\times]	[\checkmark]	3.53E+05	1.72E+04	1.02E+03	1.75E+07	94.6	2.00E+03	(0.57%)	\uparrow

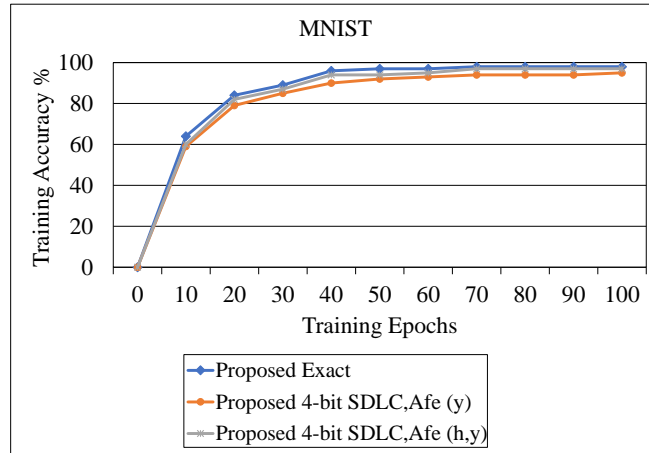


Figure 5.11: Backpropagation learning rule results for various selections of ANN hardware configurations using different MAC configurations with fixed and modified ReLU activation function in the MNIST data-set.

5.7 Conclusions

In this chapter, a novel neuron method is proposed. Firstly, the ability of the configurable approach presented in chapter 4 is taken advantage of and re-designed to include two configurations (exact and 4-bit SDLC). The 2-bit SDLC is not considered since the focus on this chapter is on the highest-level SDLC multipliers where energy reductions are achieved at the cost of reduced quality (for example, with the 4-bit SDLC in an 8x8 multiplier size) to build the new configurable MAC unit. The estimation of the accuracy of approximate configuration (4-bit SDLC) is performed using Vivado and MATLAB simulation to find the value of normalised mean error distance (NMED) that can be used as a compensation value (CM) to minimize the prediction error caused by the approximate configurations. Secondly, a modified activation function is proposed to minimize the prediction error caused by the approximate multiplier (4-bit SDLC configuration) by adding the CM value. Subsequently, the Energy-Aware Configuration Algorithm (EACA) model presented in chapter 3 is employed to allocate the optimal configuration of the MAC unit depending on the available instantaneous power. The proposed neuron design method is implemented in System-Verilog and synthesized by applying the Synopsys Design Compiler and the Xilinx Vivado Design Suite. The evaluation results indicate savings of more than 59.4% of area compared with an existing proposed solution. Thirdly, the proposed neuron including the configurable MAC and modified activation function is operated in a neuron module targeting ML systems. The presented neuron module is then scaled up to build the proposed model-driven power-adaptive multi-layer perceptron (MLP) design. Finally, an analytical model is built to sim-

ulate performance, power and energy for various examples of well-known types of MLP architectures such as MNIST.

The evaluation results illustrate that energy savings of up to 63% can be achieved with a minimal loss in output quality, compared to an existing system if the 4-bit SDLC MAC and the modified activation function are applied to the output layer on the MLP.

Chapter 6

Conclusions and Future Work

6.1 Summary

In many imprecision-resistant applications, approximate computing has recently acquired significant interest as a competitive alternative to exact computing. It provides several design methods for creating extremely energy- and performance-efficient on-chip systems at various levels of abstraction. Approximate arithmetic, which includes adders and multipliers, is one of the de facto sub-areas of approximate circuits that has gotten significant attention in the literature. This section summarises the main conclusions drawn from this thesis. The state-of-the-art techniques of approximate multipliers face different challenges, which are discussed in chapter 2. To mitigate the impact of challenges, a novel idea proposed a configurable multiplier design with the ability to dynamically tune the approximation in the multiplier via logic compression control. The capabilities of the configurable multiplier were demonstrated by introducing the energy-aware configuration algorithm (EACA) method to ascertain the optimal multiplier configuration depending on the available energy. The results show that the EACA model allows the proposed design to operate at run-time under highly variable energy conditions while sustaining execution. This configurable multiplier highlights sharing the same adders between different configurations, saving both silicon and leakage energy.

Additionally, a new method for power-adaptive ML hardware design was recommended. Firstly, the ability of the configurable approximate multiplier approach presented in chapter 3 is taken advantage of to design a configurable MAC unit. Secondly, the EACA model is employed to allocate the optimal configuration of the MAC unit depending on the available instantaneous energy. Thirdly, the proposed MAC operates in neuron modules targeting ML systems (see chapter 4). Next, the presented neuron module was scaled up to build a model-driven power-adaptive ANN design. The results show that the proposed approach can provide significant energy and area savings with negligible loss in output quality.

Finally, in chapter 5 a configurable modified activation function was proposed to minimise the prediction error caused by using the approximate multiplier. The configurable multiplier design (presented in chapter 3) can be suitably used for energy-efficient multiplier designs with a minor loss in image quality requirements. Additionally, the second and the third approaches (configurable MAC unit and modified activation function discussed in chapters 4 and 5) can be used within the power-adaptive neuron modules to extract manifold benefits for NNs with a minimal loss in output quality.

6.2 Critical Review and Future Work

The objectives of this research can be extended to open up new research directions for upcoming configurable multiplier design. Therefore, several different research directions can be drawn and determined by this thesis to achieve more performance and energy efficiency. Directions for future research are discussed as follows:

- **Scaling up the current architecture of the proposed configurable MAC unit:**

Design energy-efficient ML hardware continues to be challenging due to its overwhelming arithmetic complexities. This study proposes an adaptive approximation method for Multiply-Accumulate (MAC) units that are fundamental arithmetic components in ML systems which is discussed in chapter 4. However, scaling up the current architecture to a higher level of complexity is a promising direction for future work.

- **Dynamic Voltage Frequency Scaling (DVFS):**

The critical path latency in the proposed configurable multiplier (when utilizing 4-bit SDLC) is significantly shortened as a result of the decreased number of rows in the accumulation tree (see chapter 3). This can be used to reduce energy consumption or increase multiplication throughput without adding further timing errors by allowing the voltage/frequency to be set for power-adaptive computing purposes. For instance, a design may use different implementations of configurable multiplier (approximate and exact multipliers) to carry out the task for power-adaptive computing objectives. Slack reclamation strategy [167], which uses the free time between jobs completed by approximative multipliers to purposefully slow down execution, can explain this (e.g. scaling down the operational clock frequency). The goal is to use less power and energy while still completing tasks on time.

It is believed that the research outcomes produced by this thesis will be beneficial for the circuit design community and continue to inspire further research and development in the directions mentioned above.

References

- [1] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, “A review, classification, and comparative evaluation of approximate arithmetic circuits,” *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 4, pp. 1–34, 2017.
- [2] S. Mittal, “A survey of techniques for approximate computing,” *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, pp. 1–33, 2016.
- [3] Q. Xu, T. Mytkowicz, and N. S. Kim, “Approximate computing: A survey,” *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2015.
- [4] A. Harb, “Energy harvesting: State-of-the-art,” *Renewable Energy*, vol. 36, no. 10, pp. 2641–2654, 2011.
- [5] R. Shafik, A. Yakovlev, and S. Das, “Real-power computing,” *IEEE Transactions on Computers*, vol. 67, no. 10, pp. 1445–1461, 2018.
- [6] S. Kemp and D. Reinsel. “2022 global digital reports, [Online], Available:” <https://www.nature.com/nature/>, [last accessed on 07/12/2022]. ().
- [7] M. J. Strydom and S. Buckley, “The big data research ecosystem: An analytical literature study,” in *Research Anthology on Artificial Intelligence Applications in Security*, IGI Global, 2021, pp. 2027–2057.
- [8] A. S. Andrae and T. Edler, “On global electricity usage of communication technology: Trends to 2030,” *Challenges*, vol. 6, no. 1, pp. 117–157, 2015.
- [9] V. De, “Energy-efficient computing in nanoscale cmos,” *IEEE Design & Test*, vol. 33, no. 2, pp. 68–75, 2016.
- [10] G. E. Moore *et al.*, *Cramming more components onto integrated circuits*, 1965.
- [11] J. M. Shalf and R. Leland, “Computing beyond moore’s law,” *Computer*, vol. 48, no. 12, pp. 14–23, 2015.
- [12] M. Schulz, “The end of the road for silicon?” *Nature*, vol. 399, no. 6738, pp. 729–730, 1999.

-
- [13] R. R. Schaller, “Moore’s law: Past, present and future,” *IEEE spectrum*, vol. 34, no. 6, pp. 52–59, 1997.
- [14] L. B. Kish, “End of moore’s law: Thermal (noise) death of integration in micro and nano electronics,” *Physics Letters A*, vol. 305, no. 3-4, pp. 144–149, 2002.
- [15] A. Lingamneni and K. Palem, “What to do about the end of moore’s law, probably,” in *Proc. Design Automation Conference (DAC)*, 2012.
- [16] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, “Power challenges may end the multicore era,” *Communications of the ACM*, vol. 56, no. 2, pp. 93–102, 2013.
- [17] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” in *Proceedings of the 38th annual international symposium on Computer architecture*, 2011, pp. 365–376.
- [18] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, “The eda challenges in the dark silicon era: Temperature, reliability, and variability perspectives,” in *Proceedings of the 51st Annual Design Automation Conference*, 2014, pp. 1–6.
- [19] E. Wang, J. J. Davis, R. Zhao, *et al.*, “Deep neural network approximation for custom hardware: Where we’ve been, where we’re going,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, p. 40, 2019.
- [20] S. Cawley, F. Morgan, B. McGinley, *et al.*, “Hardware spiking neural network prototyping and application,” *Genetic Programming and Evolvable Machines*, vol. 12, no. 3, pp. 257–280, 2011.
- [21] H. Esmaeilzadeh, A. Sampson, M. Ringenburt, L. Ceze, D. Grossman, and D. Burger, “Addressing dark silicon challenges with disciplined approximate computing,” in *Proc. 4th Workshop on Energy-Efficient Design*, 2012, pp. 1–2.
- [22] J. Han and M. Orshansky, “Approximate computing: An emerging paradigm for energy-efficient design,” in *2013 18th IEEE European Test Symposium (ETS)*, IEEE, 2013, pp. 1–6.
- [23] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, “Cross-layer approximate computing: From logic to architectures,” in *Design Automation Conference (DAC), 2016 53rd ACM/EDAC/IEEE*, IEEE, 2016, pp. 1–6.

-
- [24] L. Sekanina, “Introduction to approximate computing: Embedded tutorial,” in *2016 IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, IEEE, 2016, pp. 1–6.
- [25] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, and A. Yakovlev, “Energy-efficient approximate multiplier design using bit significance-driven logic compression,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, IEEE, 2017, pp. 7–12.
- [26] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, “Impact: Imprecise adders for low-power approximate computing,” in *IEEE/ACM International Symposium on Low Power Electronics and Design*, IEEE, 2011, pp. 409–414.
- [27] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, S. Das, and A. Yakovlev, “Energy-efficient approximate wallace-tree multiplier using significance-driven logic compression,” in *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, Oct. 2017, pp. 1–6. DOI: 10.1109/SiPS.2017.8109990.
- [28] K. Al-Maaitah, G. Tarawneh, A. Soltan, I. Qiqieh, and A. Yakovlev, “Approximate adder segmentation technique and significance-driven error correction,” in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Sep. 2017, pp. 1–6. DOI: 10.1109/PATMOS.2017.8106986.
- [29] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, “Enerj: Approximate data types for safe and general low-power computation,” in *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI ’11, San Jose, California, USA: ACM, 2011, pp. 164–174, ISBN: 978-1-4503-0663-8. DOI: 10.1145/1993498.1993518.
- [30] K. Jain and V. V. Vazirani, “Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation,” *Journal of the ACM (JACM)*, vol. 48, no. 2, pp. 274–296, 2001.
- [31] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, “Architecture support for disciplined approximate programming,” *SIGPLAN Not.*, vol. 47, no. 4, pp. 301–312, Mar. 2012, ISSN: 0362-1340.

-
- [32] P. Sangeetha and A. A. Khan, "Comparison of braun multiplier and wallace multiplier techniques in vlsi," in *2018 4th International Conference on Devices, Circuits and Systems (ICDCS)*, IEEE, 2018, pp. 48–53.
- [33] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, "Design of voltage-scalable meta-functions for approximate computing," in *2011 Design, Automation & Test in Europe*, IEEE, 2011, pp. 1–6.
- [34] Y. Liu, T. Zhang, and K. K. Parhi, "Computation error analysis in digital signal processing systems with over-scaled supply voltage," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 18, no. 4, pp. 517–526, 2009.
- [35] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. Strollo, "Truncated binary multipliers with variable correction and minimum mean square error," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 6, pp. 1312–1325, 2009.
- [36] J. E. Stine and O. M. Duverne, "Variations on truncated multiplication," in *Euromicro Symposium on Digital System Design, 2003. Proceedings.*, IEEE, 2003, pp. 112–119.
- [37] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *2011 24th International Conference on VLSI Design*, IEEE, 2011, pp. 346–351.
- [38] C.-H. Lin and C. Lin, "High accuracy approximate multiplier with error correction," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, IEEE, 2013, pp. 33–38.
- [39] T. T. Hoang, M. Sjalander, and P. Larsson-Edefors, "A high-speed, energy-efficient two-cycle multiply-accumulate (mac) architecture and its application to a double-throughput mac unit," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 12, pp. 3073–3081, 2010.
- [40] J.-K. Chang, H. Lee, and C.-S. Choi, "A power-aware variable-precision multiply-accumulate unit," in *2009 9th International Symposium on Communications and Information Technology*, IEEE, 2009, pp. 1336–1339.
- [41] L.-H. Chen, O.-C. Chen, T.-Y. Wang, and Y.-C. Ma, "A multiplication-accumulation computation unit with optimized compressors and minimized switching activities," in *2005 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2005, pp. 6118–6121.

-
- [42] M. S. Kumar, D. A. Kumar, and P. Samundiswary, "Design and performance analysis of multiply-accumulate (mac) unit," in *2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]*, IEEE, 2014, pp. 1084–1089.
- [43] S. Dutt, A. Chauhan, R. Bhadoriya, S. Nandi, and G. Trivedi, "A high-performance energy-efficient hybrid redundant mac for error-resilient applications," in *2015 28th International Conference on VLSI Design*, IEEE, 2015, pp. 351–356.
- [44] D. Esposito, A. G. Strollo, and M. Alioto, "Low-power approximate mac unit," in *2017 13th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME)*, IEEE, 2017, pp. 81–84.
- [45] G. Raut, A. Biasizzo, N. Dhakad, N. Gupta, G. Papa, and S. K. Vishvakarma, "Data multiplexed and hardware reused architecture for deep neural network accelerator," *Neurocomputing*, vol. 486, pp. 147–159, 2022.
- [46] E. Grossi and M. Buscema, "Introduction to artificial neural networks," *European journal of gastroenterology & hepatology*, vol. 19, no. 12, pp. 1046–1054, 2007.
- [47] H. K. Ghritlahre and R. K. Prasad, "Application of ANN technique to predict the performance of solar collector systems - a review," *Renewable and Sustainable Energy Reviews*, vol. 84, pp. 75–88, 2018, ISSN: 1364-0321.
- [48] J. Chang, Y. Choi, T. Lee, and J. Cho, "Reducing mac operation in convolutional neural network with sign prediction," in *Reducing MAC operation in convolutional neural network with sign prediction*, Oct. 2018, pp. 177–182. DOI: 10.1109/ICTC.2018.8539530.
- [49] T. Tsai, Y. Ho, and M. Sheu, "Implementation of FPGA-based accelerator for deep neural networks," in *2019 IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, 2019, pp. 1–4.
- [50] V. Akhlaghi, A. Yazdanbakhsh, K. Samadi, R. K. Gupta, and H. Esmaeilzadeh, "Snapea: Predictive early activation for reducing computation in deep convolutional neural networks," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ser. ISCA '18, Los Angeles, California: IEEE Press, 2018, pp. 662–673, ISBN: 9781538659847.

-
- [51] K. Van Pham, T. Van Nguyen, S. B. Tran, *et al.*, “Memristor binarized neural networks,” *J. Semicond. Technol. Sci.*, vol. 18, no. 5, pp. 568–588, 2018.
- [52] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han, “A comparative evaluation of approximate multipliers,” in *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANO-ARCH)*, IEEE, 2016, pp. 191–196.
- [53] A. Alaghi and J. P. Hayes, “Survey of stochastic computing,” *ACM Transactions on Embedded computing systems (TECS)*, vol. 12, no. 2s, pp. 1–19, 2013.
- [54] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, “Analysis and characterization of inherent application resilience for approximate computing,” in *Proceedings of the 50th Annual Design Automation Conference*, 2013, pp. 1–9.
- [55] A. Paller, A. Alaghi, I. Polian, and J. P. Hayes, “Tomographic testing and validation of probabilistic circuits,” in *2011 Sixteenth IEEE European Test Symposium*, IEEE, 2011, pp. 63–68.
- [56] A. K. Mishra, R. Barik, and S. Paul, “Iact: A software-hardware framework for understanding the scope of approximate computing,” in *Workshop on Approximate Computing Across the System Stack (WACAS)*, vol. 52, 2014.
- [57] K. Natori and N. Sano, “Scaling limit of digital circuits due to thermal noise,” *Journal of applied physics*, vol. 83, no. 10, pp. 5019–5024, 1998.
- [58] K. L. Shepard and V. Narayanan, “Conquering noise in deep-submicron digital ics,” *IEEE Design & Test of Computers*, vol. 15, no. 1, pp. 51–62, 1998.
- [59] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, “Neural acceleration for general-purpose approximate programs,” in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE, 2012, pp. 449–460.
- [60] D. Candrea, A. Sharma, L. Osborn, Y. Gu, and N. Thakor, “An adaptable prosthetic socket: Regulating independent air bladders through closed-loop control,” in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2017, pp. 1–4.
- [61] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. L. O’Brien, “Quantum computers,” *nature*, vol. 464, no. 7285, pp. 45–53, 2010.

-
- [62] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information*, 2002.
- [63] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.
- [64] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja, “An architecture for fault-tolerant computation with stochastic logic,” *IEEE transactions on computers*, vol. 60, no. 1, pp. 93–105, 2010.
- [65] M. B. Parker and R. Chu, “A vlsi-efficient technique for generating multiple uncorrelated noise sources and its application to stochastic,” *IEEE Transactions on circuits and systems*, vol. 38, no. 1, 1991.
- [66] R. Ramachandran, S. Felix, M. Saranya, *et al.*, “Synthesis of cobalt sulfide-graphene (cos/g) nanocomposites for supercapacitor applications,” *IEEE transactions on nanotechnology*, vol. 12, no. 6, pp. 985–990, 2013.
- [67] C. Li, M. Hu, Y. Li, *et al.*, “Analogue signal and image processing with large memristor crossbars,” *Nature electronics*, vol. 1, no. 1, pp. 52–59, 2018.
- [68] X. Zhang, X. Zhang, S. L. Ho, and W. Fu, “A modification of artificial bee colony algorithm applied to loudspeaker design problem,” *IEEE Transactions on Magnetics*, vol. 50, no. 2, pp. 737–740, 2014.
- [69] T. T. Dung, Y. Oh, S.-J. Choi, I.-D. Kim, M.-K. Oh, and M. Kim, “Applications and advances in bioelectronic noses for odour sensing,” *Sensors*, vol. 18, no. 1, p. 103, 2018.
- [70] V. Choi, *Systems, devices, and methods for analog processing*, US Patent 8,190,548, May 2012.
- [71] L. Hardesty, *Analog computing returns*, 2016.
- [72] M. Masadeh, O. Hasan, and S. Tahar, “Comparative study of approximate multipliers,” in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 2018, pp. 415–418.
- [73] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, “Macaco: Modeling and analysis of circuits for approximate computing,” in *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, IEEE, 2011, pp. 667–673.

-
- [74] T. Ohlemueller and M. Petri, “Sample synchronization of multiple multiplexed da and ad converters in FPGAs,” in *14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, IEEE, 2011, pp. 301–304.
- [75] D. May and W. Stechele, “Voltage over-scaling in sequential circuits for approximate computing,” in *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, IEEE, 2016, pp. 1–6.
- [76] R. Ragavan, B. Barrois, C. Killian, and O. Sentieys, “Pushing the limits of voltage over-scaling for error-resilient applications,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, IEEE, 2017, pp. 476–481.
- [77] R. P. Duarte and C.-S. Bouganis, “A unified framework for over-clocking linear projections on FPGAs under pvt variation,” in *International Symposium on Applied Reconfigurable Computing*, Springer, 2014, pp. 49–60.
- [78] V. Mishra, Q. Chen, and G. Zervas, “Reon: A protocol for reliable software-defined FPGA partial reconfiguration over network,” in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, IEEE, 2016, pp. 1–7.
- [79] J. Rabaey, *Low power design essentials*. Springer Science & Business Media, 2009.
- [80] M. Nicolaidis, “Double-sampling design paradigm—a compendium of architectures,” *IEEE transactions on device and materials reliability*, vol. 15, no. 1, pp. 10–23, 2015.
- [81] M. Kawato, K. Furukawa, and R. Suzuki, “A hierarchical neural-network model for control and learning of voluntary movement,” *Biological cybernetics*, vol. 57, no. 3, pp. 169–185, 1987.
- [82] D. S. Levine, “Neural network modeling.,” *Physics of life reviews*, 2002.
- [83] S. Lockery and T. Sejnowski, “Distributed processing of sensory information in the leech. iii. a dynamical neural network model of the local bending reflex,” *Journal of Neuroscience*, vol. 12, no. 10, pp. 3877–3895, 1992.
- [84] H. Li, Z. Zhang, and Z. Liu, “Application of artificial neural networks for catalysis: A review,” *Catalysts*, vol. 7, no. 10, p. 306, 2017.
- [85] T. D. Sanger, “Optimal unsupervised learning in a single-layer linear feed-forward neural network,” *Neural networks*, vol. 2, no. 6, pp. 459–473, 1989.

-
- [86] T. J. Sejnowski and P. Churchland, *The computational brain*. JSTOR, 1992.
- [87] E. Bisong, *Building machine learning and deep learning models on Google cloud platform*. Springer, 2019.
- [88] R. Jozefowicz, W. Zaremba, and I. Sutskever, “An empirical exploration of recurrent network architectures,” in *International conference on machine learning*, PMLR, 2015, pp. 2342–2350.
- [89] O. Abdel-Hamid, L. Deng, and D. Yu, “Exploring convolutional neural network structures and optimization techniques for speech recognition.,” in *Interspeech*, Citeseer, vol. 2013, 2013, pp. 1173–5.
- [90] V. Dumoulin and F. Visin, “A guide to convolution arithmetic for deep learning,” *arXiv preprint arXiv:1603.07285*, 2016.
- [91] D. J. Livingstone, *Artificial neural networks: methods and applications*. Springer, 2008.
- [92] R. Rojas, *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.
- [93] R. Sathya, A. Abraham, *et al.*, “Comparison of supervised and unsupervised learning algorithms for pattern classification,” *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 2, pp. 34–38, 2013.
- [94] M. Bennamoun, “Neural network learning rules,” *University of Western Australia, Australia*, 2018.
- [95] V. Stankovic, “Introduction to machine learning in image processing,” *University of Strathclyde, Glasgow*, 2017.
- [96] S. D. Fabiyi, “A review of unsupervised artificial neural networks with applications,” *International Journal of Computer Applications*, vol. 181, no. 40, pp. 22–26, 2019.
- [97] A. F. Atiya, “An unsupervised learning technique for artificial neural networks,” *Neural Networks*, vol. 3, no. 6, pp. 707–711, 1990.
- [98] A. Gosavi, “Neural networks and reinforcement learning,” *Department of Engineering Management and Systems Engineering Missouri University of Science and Technology Rolla*, 2015.
- [99] F. Woergoetter and B. Porr, “Reinforcement learning,” *Scholarpedia*, vol. 3, no. 3, p. 1448, 2008.

-
- [100] E. haq Shaik and N. Rangaswamy, “Multi-mode interference-based photonic crystal logic gates with simple structure and improved contrast ratio,” *Photonic Network Communications*, vol. 34, no. 1, pp. 140–148, 2017.
- [101] R. R. Asaad and R. I. Ali, “Back propagation neural network (bpnn) and sigmoid activation function in multi-layer networks,” *Academic Journal of Nawroz University*, vol. 8, no. 4, pp. 216–221, 2019.
- [102] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, S. Das, and A. Yakovlev, “Significance-driven logic compression for energy-efficient multiplier design,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 417–430, 2018.
- [103] G. Zhong, A. Dubey, C. Tan, and T. Mitra, “Synergy: A HW/SW framework for high throughput cnns on embedded heterogeneous SoC,” *CoRR*, vol. abs/1804.00706, 2018.
- [104] K. Al-Maaitah, G. Tarawneh, A. Soltan, I. Qiqieh, and A. Yakovlev, “Approximate adder segmentation technique and significance-driven error correction,” in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2017, pp. 1–6. DOI: 10.1109/PATMOS.2017.8106986.
- [105] K. Al-Maaitah, I. Qiqieh, A. Soltan, and A. Yakovlev, “Configurable-accuracy approximate adder design with light-weight fast convergence error recovery circuit,” in *2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, 2017, pp. 1–6. DOI: 10.1109/AEECT.2017.8257753.
- [106] D. Burke, D. Jenkus, I. Qiqieh, R. Shafik, S. Das, and A. Yakovlev, “Special session paper: Significance-driven adaptive approximate computing for energy-efficient image processing applications,” in *2017 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2017, pp. 1–2. DOI: 10.1145/3125502.3125554.
- [107] N. H. Weste and D. Harris, *CMOS VLSI design: a circuits and systems perspective*. Pearson Education India, 2015.
- [108] S. P. Beeby, R. N. Torah, M. J. Tudor, *et al.*, “A micro electromagnetic generator for vibration energy harvesting,” *Journal of Micromechanics and microengineering*, vol. 17, no. 7, p. 1257, 2007.
- [109] J. A. Paradiso and T. Starner, “Energy scavenging for mobile and wireless electronics,” *IEEE Pervasive computing*, vol. 4, no. 1, pp. 18–27, 2005.

-
- [110] S. Chalasani and J. M. Conrad, “A survey of energy harvesting sources for embedded systems,” in *IEEE SoutheastCon 2008*, IEEE, 2008, pp. 442–447.
- [111] S. P. Beeby, R. Torah, M. Tudor, *et al.*, “A micro electromagnetic generator for vibration energy harvesting,” *Journal of Micromechanics and micro-engineering*, vol. 17, no. 7, p. 1257, 2007.
- [112] J. A. Paradiso and T. Starner, “Energy scavenging for mobile and wireless electronics,” *IEEE Pervasive computing*, vol. 4, no. 1, pp. 18–27, 2005.
- [113] D. Dondi, A. Bertacchini, D. Brunelli, L. Larcher, and L. Benini, “Modeling and optimization of a solar energy harvester system for self-powered wireless sensor networks,” *IEEE Transactions on Industrial Electronics*, vol. 55, no. 7, pp. 2759–2766, 2008. DOI: 10.1109/TIE.2008.924449.
- [114] C. S. Wallace, “A suggestion for a fast multiplier,” *IEEE Transactions on Electronic Computers*, vol. EC-13, no. 1, pp. 14–17, 1964. DOI: 10.1109/PGEC.1964.263830.
- [115] K. Bhardwaj, P. S. Mane, and J. Henkel, “Power- and area-efficient approximate wallace tree multiplier for error-resilient systems,” in *Fifteenth International Symposium on Quality Electronic Design*, 2014, pp. 263–269. DOI: 10.1109/ISQED.2014.6783335.
- [116] J. Liang, J. Han, and F. Lombardi, “New metrics for the reliability of approximate and probabilistic adders,” *IEEE Transactions on computers*, vol. 62, no. 9, pp. 1760–1771, 2012.
- [117] C. Liu, J. Han, and F. Lombardi, “A low-power, high-performance approximate multiplier with configurable partial error recovery,” in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2014, pp. 1–4.
- [118] J. M. Phillips and W. M. Tai, “The gaussiansketch for almost relative error kernel distance,” *arXiv preprint arXiv:1811.04136*, 2018.
- [119] J. Liang, J. Han, and F. Lombardi, “New metrics for the reliability of approximate and probabilistic adders,” *IEEE Transactions on computers*, vol. 62, no. 9, pp. 1760–1771, 2012.
- [120] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, “Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 258–261. DOI: 10.23919/DATE.2017.7926993.

-
- [121] Ibrahim.Haddadi. “Thesis codes, [Online],Available:” <https://www.overleaf.com/read/ykhyrmqjmcph>, [last accessed on 26/06/2023]. ().
- [122] 2. Ultra96-V2 Development Board 2018, *Avnet.*, [Online],Available: <http://zedboard.org/product/ultra96-v2-development-board>, [last accessed on 07/12/2022].
- [123] P. D. Mitcheson, E. M. Yeatman, G. K. Rao, A. S. Holmes, and T. C. Green, “Energy harvesting from human and machine motion for wireless electronic devices,” *Proceedings of the IEEE*, vol. 96, no. 9, pp. 1457–1486, 2008.
- [124] A. S. Weddell, M. Magno, G. V. Merrett, D. Brunelli, B. M. Al-Hashimi, and L. Benini, “A survey of multi-source energy harvesting systems,” in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2013, pp. 905–908. DOI: 10.7873/DATE.2013.190.
- [125] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini, “Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems,” *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 15–18, 2014.
- [126] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons, 2011.
- [127] *Synopsys design compiler*, <https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html>, Accessed: 2021-09-10.
- [128] *Faraday technology corporation*, <http://www.faraday-tech.com>, Accessed: 2021-09-12.
- [129] A. Biswas and A. P. Chandrakasan, “Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power cnn-based machine learning applications,” in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb. 2018, pp. 488–490. DOI: 10.1109/ISSCC.2018.8310397.
- [130] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.
- [131] L. Benini, “Plenty of room at the bottom: Micropower deep learning for cognitive cyberphysical systems,” *RTNS Keynote*, 2017.
- [132] M. Cho and Y. Kim, “Fpga-based convolutional neural network accelerator with resource-optimized approximate multiply-accumulate unit,” *Electronics*, vol. 10, no. 22, p. 2859, 2021.

-
- [133] S. Draghici, “Neural networks in analog hardware—design and implementation issues,” *International journal of neural systems*, vol. 10, no. 01, pp. 19–42, 2000.
- [134] S. Li, W. Wen, Y. Wang, S. Han, Y. Chen, and H. Li, “An FPGA design framework for CNN sparsification and acceleration,” in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, IEEE, 2017, pp. 28–28.
- [135] S. Perri, F. Spagnolo, F. Frustaci, and P. Corsonello, “Designing energy-efficient approximate multipliers,” *Journal of Low Power Electronics and Applications*, vol. 12, no. 4, p. 49, 2022.
- [136] C.-H. Chang and R. K. Satzoda, “A low error and high performance multiplexer-based truncated multiplier,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 12, pp. 1767–1771, 2010. DOI: 10.1109/TVLSI.2009.2027327.
- [137] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, “Energy-efficient approximate multiplication for digital signal processing and classification applications,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 6, pp. 1180–1184, 2015. DOI: 10.1109/TVLSI.2014.2333366.
- [138] P. Balasubramanian and N. Mastorakis, “Performance comparison of carry-lookahead and carry-select adders based on accurate and approximate additions,” *Electronics*, vol. 7, no. 12, p. 369, 2018.
- [139] H. Benmaghnia, M. Martel, and Y. Seladji, “Fixed-point code synthesis for neural networks,” *arXiv preprint arXiv:2202.02095*, 2022.
- [140] R. O. Duda, P. E. Hart, *et al.*, *Pattern classification*. John Wiley & Sons, 2006.
- [141] A. Wheeldon, R. Shafik, T. Rahman, J. Lei, A. Yakovlev, and O.-C. Granmo, “Learning automata based energy-efficient ai hardware design for iot applications,” *Philosophical Transactions of the Royal Society A*, vol. 378, no. 2182, p. 20190593, 2020.
- [142] Y. LeCun, C. Cortes, and C. J. Burges, *The MNIST database of handwritten digits*, [Online], Available: <http://yann.lecun.com/exdb/mnist/>, [last accessed on 18/11/2021].

-
- [143] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
- [144] R. O. Duda, P. E. Hart, *et al.*, *Pattern classification*. John Wiley & Sons, 2006.
- [145] A. Eldem, “An application of deep neural network for classification of wheat seeds,” *Avrupa Bilim ve Teknoloji Dergisi*, no. 19, pp. 213–220, 2020.
- [146] A. Eldem, H. Eldem, and D. Üstün, “A model of deep neural network for iris classification with different activation functions,” in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, IEEE, 2018, pp. 1–4.
- [147] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, “A majority-based imprecise multiplier for ultra-efficient approximate image multiplication,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 11, pp. 4200–4208, 2019.
- [148] S. Ullah, S. Rehman, M. Shafique, and A. Kumar, “High-performance accurate and approximate multipliers for fpga-based hardware accelerators,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 2, pp. 211–224, 2021.
- [149] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.
- [150] A. Paszke, S. Gross, F. Massa, *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [151] T. M. Jamel and B. M. Khammas, “Implementation of a sigmoid activation function for neural network using FPGA,” in *13th Scientific Conference of Al-Ma’moon University College*, vol. 13, 2012.
- [152] S. Ngah and R. A. Bakar, “Sigmoid function implementation using the unequal segmentation of differential lookup table and second order nonlinear function,” *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 2-8, pp. 103–108, 2017.

-
- [153] P. W. Zaki, A. M. Hashem, E. A. Fahim, *et al.*, “A novel sigmoid function approximation suitable for neural networks on FPGA,” in *2019 15th International Computer Engineering Conference (ICENCO)*, IEEE, 2019, pp. 95–99.
- [154] C. Banerjee, T. Mukherjee, and E. Pasilio Jr, “An empirical study on generalizations of the relu activation function,” in *Proceedings of the 2019 ACM Southeast Conference*, 2019, pp. 164–167.
- [155] D. Costarelli and R. Spigler, “Approximation results for neural network operators activated by sigmoidal functions,” *Neural Networks*, vol. 44, pp. 101–106, 2013.
- [156] A. H. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi, “Efficient hardware implementation of the hyperbolic tangent sigmoid function,” in *2009 IEEE International Symposium on Circuits and Systems*, 2009, pp. 2117–2120. DOI: 10.1109/ISCAS.2009.5118213.
- [157] S. Goel, S. Karmalkar, and A. Klivans, “Time/accuracy tradeoffs for learning a relu with respect to gaussian marginals,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [158] V. Beiu, J. Peperstraete, J. Vandewalle, and R. Lauwereins, “Close approximations of sigmoid functions by sum of step for vlsi implementation of neural networks,” *Sci. Ann. Cuza Univ.*, vol. 3, pp. 5–34, 1994.
- [159] H. Jiang, J. Han, and F. Lombardi, “A comparative review and evaluation of approximate adders,” in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, 2015, pp. 343–348.
- [160] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, “Approximate hybrid high radix encoding for energy-efficient inexact multipliers,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 421–430, 2017.
- [161] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, “A majority-based imprecise multiplier for ultra-efficient approximate image multiplication,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 11, pp. 4200–4208, 2019. DOI: 10.1109/TCSI.2019.2918241.
- [162] S. Perri, F. Spagnolo, F. Frustaci, and P. Corsonello, “Designing energy-efficient approximate multipliers,” *Journal of Low Power Electronics and Applications*, vol. 12, no. 4, p. 49, 2022.

- [163] I. Haddadi, I. Qiqieh, R. Shafik, F. Xia, M. Al-Hayanni, and A. Yakovlev, “Run-time configurable approximate multiplier using significance-driven logic compression,” in *IEEE International Conference on Computer Design (ICCD 2021)*, 2021.
- [164] M. S. Kumar, D. A. Kumar, and P. Samundiswary, “Design and performance analysis of multiply-accumulate (mac) unit,” in *2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]*, IEEE, 2014, pp. 1084–1089.
- [165] M. Imani, D. Peroni, and T. Rosing, “Cfpu: Configurable floating point multiplier for energy-efficient computing,” in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.
- [166] J. Si, S. L. Harris, and E. Yfantis, “A dynamic relu on neural network,” in *2018 IEEE 13th Dallas Circuits and Systems Conference (DCAS)*, 2018, pp. 1–6. DOI: 10.1109/DCAS.2018.8620116.
- [167] I. Pietri and R. Sakellariou, “Energy-aware workflow scheduling using frequency scaling,” in *2014 43rd International Conference on Parallel Processing Workshops*, 2014, pp. 104–113. DOI: 10.1109/ICPPW.2014.26.