UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

Pedro Henrique Aquino Silva

Exploring Approximate Comparator Circuits in the Energy Efficient Design of Decision Trees

Florianópolis
2022

Pedro Henrique Aquino Silva

Exploring Approximate Comparator Circuits in the Energy Efficient Design of Decision Trees

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina – UFSC como requisito parcial para obtenção do título de bacharel em Ciências da Computação.
Orientadora: Profª. Cristina Meinhardt, Dra.
Coorientador: Prof. Mateus Grellert, Dr.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Silva, Pedro Henrique Aquino
   Exploring Approximate Comparator Circuits in the Energy Efficient Design of Decision Trees / Pedro Henrique Aquino Silva ; orientadora, Cristina Meinhardt, coorientador, Mateus Grellert, 2022.
   79 p.

   Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2022.

   Inclui referências.

   1. Ciências da Computação. 2. computação aproximada. 3. árvores de decisão. 4. circuitos comparadores. I. Meinhardt, Cristina. II. Grellert, Mateus. III. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. IV. Título.

Pedro Henrique Aquino Silva

Exploring Approximate Comparator Circuits in the Energy Efficient Design of Decision Trees

O presente trabalho em nível de Graduação foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Bacharel em Ciências da Computação.

Coordenação do Curso de Graduação em
Ciências da Computação

Prof^{a}. Cristina Meinhardt, Dra.
Orientadora

Florianópolis, 2022.

Este trabalho é dedicado a todos que comigo caminharam
até aqui.

ACKNOWLEDGEMENTS

Firstly, I wish to thank my family, who in spite of our differences, have continued to support me in all the ways they can. Their careful maintenance of my health and well-being, and their incentive for my personal and career development are sources of immense inspiration that one day I hope to be able to replicate.

Thanks is overdue to my advisor, Cristina Meinhardt, who welcomed me to the Embedded Computing Laboratory when I was all but starting my studies and barely knew anything about circuits. I am grateful for her patience and kindness, and for her guidance in all of our interesting discussions. I am also thankful to my co-advisor, Mateus Grellert, for introducing me to a world very different to the one I was accustomed to and for his knowledge and help in exploring it. I could not be more thankful for the opportunity to work with them, and for the many lessons they continue to teach me.

I also wish to express my gratitude to my colleagues, who have been very understanding through the periods of anxiety while writing this text. Lastly, I would like to thank my friends, with whom I have shared both incredibly hard and amazingly joyful times. I have nothing but gratitude for all the support and encouragement they have given me, and I will always cherish all the memories of food, music and dance we enjoyed together. I hope to have been able to reciprocate their love in the same quantity, and I truly wish for us to continue to grow together.

"Para de se comparar com todo mundo."
(Minha mãe, sem saber que eu ia acabar trabalhando com comparadores, ∼2008)

"I don't know where I'm going from here but I promise it won't bore you."
(David Bowie, 1997)

RESUMO

A Computação Aproximada aplicada ao projeto de circuitos digitais consiste em descrever circuitos que eventualmente tenham resultados inexatos ou imprecisos, de modo a obter arquiteturas mais eficientes em área, atraso ou dissipação de potência. Diversos trabalhos recentes evidenciam os resultados desta técnica em circuitos aritméticos, principalmente somadores e multiplicadores. Contudo, ainda existe uma lacuna no estudo de técnicas voltadas para circuitos de comparação, os quais são amplamente utilizados por uma gama de aplicações. Este trabalho investiga o uso de aproximação em comparadores em nível de circuito visando eficiência energética, e sua aplicação em modelos de aprendizado de máquina baseados em Árvores de Decisão (DT). A aproximação é inserida tanto em circuitos dedicados (AxDC), como em comparadores baseados em *full adders* (FA), de modo a minimizar a dissipação de potência do circuito. Foram propostas duas arquiteturas dedicadas com aproximação em nível de portas lógicas, denominadas AxDC1 e AxDC2. Estas arquiteturas exploram 25% ou 50% de aproximação dos bits menos significativos respectivamente, por meio das técnicas de truncamento e cópia. Para as versões baseadas em FA, foi utilizado um comparador *ripple carry* com 100% de blocos aproximados, por meio de três FAs aproximados retirados da literatura (SMA, AMA1 e AMA2). Versões de 8 bits dos circuitos foram descritos em tecnologia FinFET de 7 nm, e avaliados contra um comparador exato de referência, utilizando-se de simulações elétricas. O impacto da aproximação foi avaliado na caracterização elétrica e no estudo do erro dos comparadores. Posteriormente, o efeito de se utilizar a aproximação como teste de atributo em DTs foi estudado, empregando o algoritmo C5.0 e 5 datasets do *UCI Machine Learning Repository*. Os resultados experimentais obtidos para cada aproximação de comparador sugerem que o AxDC1 é o melhor candidato ao uso em uma implementação em *hardware* de DTs, uma vez que tem um impacto mínimo na acurácia, de somente 0.12% em média, enquanto traz uma redução de consumo energético de 28% na média em relação ao comparador exato de referência. O AxDC2, por conta de sua aproximação mais agressiva, obteve resultados insatisfatórios na acurácia, com piora de 65% em média, em relação ao comparador de referência, ao passo que não houve melhoras na eficiência energética devido ao maior número de operações realizadas na classificação com este comparador. Por fim,a versão de comparador baseado no AMA1 também apresentou resultados promissores, obtendo o maior ganho em eficiência energética, apesar da queda de acurácia na classificação.

Palavras-chave: Computação Aproximada. Árvores de Decisão. Comparadores.

ABSTRACT

Approximate Computing applied to the design of digital circuits consists in describing circuits that eventually present inexact or imprecise results, with the goal of obtaining architectures with improved characteristics in area, delay, or power dissipation. Various studies highlight the results of this technique in arithmetic circuits, mainly adders and multipliers. However, there is still a gap in the study of techniques focused on comparator circuits, that are widely used in a range of applications. This work investigates the adoption of approximation techniques in the design of comparators at the logic gate and circuit levels, aiming to improve energy efficiency in Machine Learning models based on Decision Trees (DTs). The approximation is inserted both in dedicated comparator circuits (AxDC) and in comparators based on full adders (FA) with a focus on minimizing the area and power dissipation of the circuit. Two architectures with gate-level approximation were proposed for the dedicated comparators, called AxDC1 and AxDC2. These architectures exploit 25% or 50% approximation of Least Significant Bits (LSB) respectively, through truncation and copy strategies. For the FA-based versions, a Ripple Carry Comparator with 100% approximation was used, where the approximation was inserted through three approximate FAs taken from the literature (SMA, AMA1, and AMA2). 8-bit versions of the circuits have been described in 7 nm FinFET process node, and evaluated against a reference exact comparator, using electrical simulations. The impact of the approximation was first evaluated in the electrical characterization and the error behavior of the comparators. Subsequently, the impact of implementing the approximation on the attribute testing of DT classifiers was studied using the C5.0 algorithm and 5 datasets from the UCI Machine Learning Repository. The experimental results obtained for each comparator version suggest that AxDC1 is the best candidate for use in an implementation in a DT hardware implementation, since it has a minimal impact on accuracy, of only 0.12% on average, while providing a reduction of nearly 28% on average in the energy consumption of the inference, compared to the exact reference comparator. Due to its more aggressive approach, the AxDC2 obtained unsatisfactory results in accuracy, with a decrease of 65% on average, in relation to the reference comparator, while there were no improvements in energy efficiency due to the greater number of operations carried out in the classification with this comparator. Finally, just like the AxDC1, the AMA1-based comparator also showed promising results, obtaining the greatest gain in energy efficiency, despite the drop in classification accuracy.

Keywords: Approximate Computing. Decision Trees. Comparator circuits.

LIST OF FIGURES

LIST OF CODE LISTINGS

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AMA1	Approximate Mirror Adder 1
AMA2	Approximate Mirror Adder 2
ASIC	Application Specific Integrated Circuit
AxC	Approximate Computing
AxDC	Approximate Dedicated Comparator
AxDC1	Approximate Dedicated Comparator 1
AxDC2	Approximate Dedicated Comparator 2
CART	Classification and Regression Trees
CMOS	Complementary Metal-Oxide-Semiconductor
DT	Decision Tree
DUT	Device Under Test
ED	Error Distance
EDC	Exact Dedicated Comparator
EDERP	Energy-Delay-Error Rate Product
EDP	Energy Delay Product
ER	Error Rate
FA	Full Adder
FinFET	Fin Field-Effect Transistor
FO4	Fan-out of Four
FPGA	Field-Programmable Gate Array
IoT	Internet of Things
LSB	Least Significant Bit
LUT	Look-Up Table
MA	Mirror Adder
ML	Machine Learning
MSB	Most Significant Bit
NN	Neural Network
PDERP	Power-Delay-Error Rate Product
PDK	Process Design Kit
PDP	Power-Delay Product
PTL	Pass Transistor Logic
PTM	Predictive Technology Model
RCC	Ripple Carry Comparator
RTL	Register-Transfer Level

SMA	Simplified Mirror Adder
VFDT	Very Fast Decision Tree

CONTENTS

1 INTRODUCTION

In recent years, there has been increased interest in energy efficiency at all levels of computer systems design. This has been partly fueled by the current ubiquity of battery-powered and energy-restricted computer applications, such as mobile devices and sensor networks (BARUA; CHANDRA MONDAL, 2018). Another source of this interest is the ever-growing usage of computing and energy-intensive applications, such as Machine Learning (ML) and Big Data analytics, which in due course has motivated the emergence of concerns on the greenness of such systems (SINGH, S., 2015).

In this context, Machine Learning (ML) applications are present in our daily lives through recommendation systems, healthcare, and various big data applications. To an increasing extent, these ML algorithms are being introduced to mobile and battery-powered devices. While training ML models for these use cases is generally extremely compute and resource intensive (SINGH, S., 2015; BARUA; CHANDRA MONDAL, 2018), their operation in power-restricted environments also requires the use of low-power design techniques to enhance battery life and reliability. These facts highlight the necessity for energy-efficient ML systems (AL-JARRAH et al., 2015), recalling the demand for dedicated hardware solutions. ML algorithms generally involve a large number of independent operations, enlarging the design space to allow for hardware optimization, while being resilient to errors due to the stochastic nature of the training algorithms. These facts indicate Approximate Computing (AxC) as a suitable approach for improving energy efficiency in dedicated hardware for ML systems (HAN, 2016; ABREU; GRELLERT; BAMPI, 2020).

Since the 2010s, approximation has been an increasingly popular research topic in computer science and related fields, as a means of increasing the design space to consider power reduction techniques in digital systems (BARUA; MONDAL, 2019). AxC is an emerging research area that exploits the fact that many applications have soft constraints in terms of accuracy, trading the exactness of operations for significant energy savings (HAN, 2016). These techniques have been explored in both hardware and software in different contexts, such as Internet of Things (IoT) devices, video, and audio processing, ML and other error-tolerant applications (MOREAU; SAMPSON; CEZE, 2015; STROLLO; ESPOSITO, 2018; MARWAHA; SHARMA, A., 2018). When considering ML applications, most recent projects have delved into developing AxC solutions with a particular focus on Neural Networks (NNs) (REAGEN et al., 2016; ZHANG, B.; DAVOODI; HU, 2018; GOEL et al., 2020). However, the use of NNs may still be costly in energy-restricted environments, for example, due to a large number of multiplication operations necessary and other architectural requirements (ABREU; GRELLERT; BAMPI, 2020). All the while, simpler and cheaper ML models, such as Decision Trees (DTs), can provide satisfactory results for a large number of inference problems in various fields of application (LI, Q.; BERMAK, 2011; RUSSELL; NORVIG, 2009).

While most studies evaluating the usage of AxC techniques in ML models investigate architectural and software approaches (KUMAR; GOYAL; VARMA, 2017; GARCÍA-MARTÍN et al., 2021), there is much interest in investigating and designing AxC arithmetic blocks (OSTA et al., 2017; SEKANINA; VASICEK; MRAZEK, 2022).

Be that as it may, most of these efforts concentrate on the proposal and development of approximate adders, multiplexers, and multipliers. Note that during the classification stage of a tree-based model, attribute tests are processed to determine the target path on the tree for a given value under classification, guiding the tree traversal. These tests are most frequently performed as comparisons, emphasizing the need for optimizations in comparator circuits. Improvements in the delay and power of such circuits can significantly reduce the hardware requirements on a decision tree synthesis flow. Thus, exploring approximation and optimization solutions for the design of comparator circuits becomes essential.

To the best of my knowledge, little research has been published on approximate comparator circuits. Generally, these researches explore designs at the architectural level, introducing approximations focused on delay optimizations for large input bit-widths, and are less suited for applications with higher energy restrictions (ZHOU et al., 2018). After my efforts in reviewing the literature, I believe this is the first prospective study on the circuit-level approximation of comparators tailored for tree-based models.

1.1 OBJECTIVES

The main goal of this work is to propose a set of novel approximate comparator circuits, alongside an automated workflow for evaluating the impact of their usage in tree-based classification models. The proposed design flow allows the evaluation of trade-offs between power and accuracy for continuous and mixed datasets and Decision Tree (DT) algorithms. Specifically, this work aims to:

- Propose approximate comparator architectures;
- Evaluate and describe the performance of DT classification models trained with the proposed comparators, observing the impact of each approximation;
- Compare obtained results from different approximation techniques and exact versions, as well as compare results with related work.

The secondary objectives of this work include:

1. analyzing and characterizing the electrical behavior of each proposed comparator;
2. evaluating the performance of each design in the complete operation of a DT application.

1.2 STRUCTURE OF THIS WORK

The remaining of this document is organized as follows. Chapter 2 presents the fundamental concepts of Decision Tree learning and Approximate Computing and electrical characterization with SPICE. Chapter 3 describes the related work on low-power comparator circuits, as well as the energy-efficient design of Decision Tree applications in hardware. Chapter 4 highlights some existing techniques for comparator design, and details the proposed approximations for $n-bit$ comparator circuits, along with the electrical and error behavior and characterization for each circuit. In Chapter 5 we discuss the developed workflow for evaluating the insertion of approximation in tree-based applications, focusing on DTs models. Finally, Chapter 6 presents a summary of the author's conclusions and some insights regarding possible paths for the continuation of this work.

2 BACKGROUND

The development of this project involves three main topics: approximate computing, decision trees, and nanometer technology. The following sections provide an overview of the theoretical background necessary to understand, develop and evaluate the approximate comparators proposed and their application in decision tree classifiers. To meet this goal, firstly, we introduce Approximate Computing in Section 2.1 and give an overview of some techniques for approximation of arithmetic operations at the circuit and gate level, which in this work are employed in the design of approximate comparators. Then, Section 2.2 discusses a Decision Tree Learning strategy based on entropy and information gain, then highlights the importance of the comparison operation in both the learning and usage of Decision Trees for numeric-valued attributes. Finally, Section 2.3 describes the methods for obtaining electrical information, such as power consumption and delay, for nanometer technology nodes using SPICE simulations. These methods are used throughout this work to estimate gains in the circuits proposed and are discussed here in detail.

2.1 APPROXIMATE COMPUTING

The specification of a computer application can be viewed as the set of outputs accepted as correct or reliable for given inputs. Many important computing applications are robust to noise or do not have exactness as the main requirement in their specification (STANLEY-MARBELL et al., 2020). Examples of applications include nondeterministic or probabilistic algorithms and applications where the outputs will be perceived by the human senses. Such design cases are generally termed *error-tolerant*, and thus their full reliability can be traded off for reduced resource usage.

In this context, the increase in nondeterministic applications and hardware-restricted devices has triggered the interest in techniques that take advantage of the flexibility in a system's specification to achieve resource savings. Such techniques have been termed in the literature as **Approximate Computing (AxC)**.

AxC includes techniques on various levels of design, ranging from programming languages and compilation techniques to hardware architectures that exploit nondeterminism exposed in the software or software hardware layer, to new dedicated hardware designs with correctness versus resource usage trade-offs in mind.

To better discuss AxC techniques, we provide the definitions used in this work for the terms *error*, *precision*, and *accuracy*, strongly inspired by the terminology defined by Stanley-Marbell et al. (2020).

An **error** consists of a failure of the system to produce the expected output. We note that errors may still be usable computations, in which case the system's specification is deemed error-tolerant.

Precision is defined here to mean the limits of possible exactness in a computing system. For instance, the double-precision floating point number system is a system of representation of real numbers in which each value is represented by $64 - bit$ arrays organized according to the IEEE 754 standard. In this sense, the precision of a computer system may be viewed as the degree of discretization of the data and operations used, in our example, numbers have $64 - bit$ precision.

Accuracy is defined as a measure of the distance between the outputs provided by the computing system and the results expected from the system's specification. Note that, in this work, we discuss two levels of accuracy: in the accuracy of the learned DT model and in the inexact arithmetic blocks. In DTs and ML in general, the accuracy of a learned model is the degree of correctness of the *model* itself in classification applications, this is the rate of the model's correct predictions over an input dataset. In AxC arithmetic, we can also use the term *accuracy* for the rate of successes or exact results of an approximated *operation.* To avoid confusion, here we only use *accuracy* in the context of ML, while for approximation of functions, we use a set of error metrics.

In recent years, research focused on AxC has been centered around four major areas: instruction processing, communication, hardware circuits and systems, and cloud computing. When tackling hardware and circuits approximation, in particular, most research efforts have been directed to the approximation of logic circuits, adders, multiplexers, multipliers, neural accelerators, and memories, with various techniques ranging from the architecture-level to gate and transistor-level approximation (BARUA; MONDAL, 2019). Since this work focuses on arithmetic operations – namely comparisons –, we shall now provide an overview of different techniques used in the design of approximate operators. The information provided here is focused on the methods of approximation used in this project, and the interested reader may find more detailed descriptions of these and other techniques in Sekanina, Vasicek, and Mrazek (2022).

Approximate versions of digital circuits are frequently obtained by **functional approximation**, in which we first start with an original exact circuit and modify its logic behavior to obtain the desired trade-off of quality and the electrical characteristics sought. These characteristics include, for example, power consumption, area, and delay. Further improvements on these circuits may be obtained by other power reduction techniques, such as voltage scaling or other technology-dependent methods. This method of functional approximation provides various levels of abstraction or application, and the designer may choose the level most applicable to the target technology. For example, if the target technology for a given system is ASIC, the design might employ transistor-level approximations, while for FPGA-based systems, gate-level or LUT-level approximations should be preferred.

In application-specific designs, a common and straightforward way to approximate an arithmetic operation is to perform **truncation**. This technique consists in discarding the h least significant bits and performing the exact operation in the remaining $n-h$ input bits. The resulting circuit has, thus, a reduced area, delay, and power consumption. Due to the simplicity and good results, along with well-understood error behavior, this technique is usually used as a baseline implementation and it is only suitable for smaller circuits. The technique used for approximation in this work is developed from the truncation method: one performs the functional approximation on a smaller circuit, which is then generalized to $n-bit$ versions.

2.1.1 Error assessment in AxC applications

However, to fully characterize the approximated circuit, it is not enough to evaluate the electrical gains from approximating the operation, which is necessary to establish metrics for error analysis. While, according to Sekanina, Vasicek, and Mrazek (2022), various researchers apply techniques from the formal verification of computer systems, in this work, we decide to simplify our analysis, using two error metrics: Error Rate (ER) and Error Distance (ED) (STANLEY-MARBELL et al., 2020). **Error Rate** is defined as the ratio of inputs that cause an inexact output out of all possible inputs, while **Error Distance** is the distance of the result to the correct output. In the case of circuits with a single Boolean output, such as comparators returning a single bit, the ED is trivially given by the total number of errors, and these two metrics become analogous. Despite this, we still provide the numbers for both, since it is helpful to understand the error behavior as both a percentage and an absolute number.

The calculation of ED for a binary function f and an approximation f_{ax} with $1-bit$ outputs and $n-bit$ input is defined in Eq. (1).

$$ED = \sum_{i=0}^{2^n} [f(i) \neq f_{ax}(i)] \tag{1}$$

The formula uses Iverson notation, as used by Knuth (1992), defined in Eq. (2).

$$[P] = \begin{cases} 1 & \text{if } P \text{ is true;} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Finally, we can define the ER in terms of ED, as shown in Eq. (3).

$$ER = \frac{ED}{2^n} \tag{3}$$

Finally, we reiterate that most AxC studies at the circuit level have been focused on more hardware-intensive arithmetic units than the comparator, i.e. multipliers and adders. That said, during the literature review, I found a few studies on the approximation of comparisons, at both the circuit and software levels. The techniques employed and results achieved will be further discussed in Chapter 3. Even so, it is important to highlight that these circuits are present in a variety of error-tolerant systems, and as in the case of hardware realizations of DTs, they are critical circuits to investigate power optimizations, including with AxC techniques.

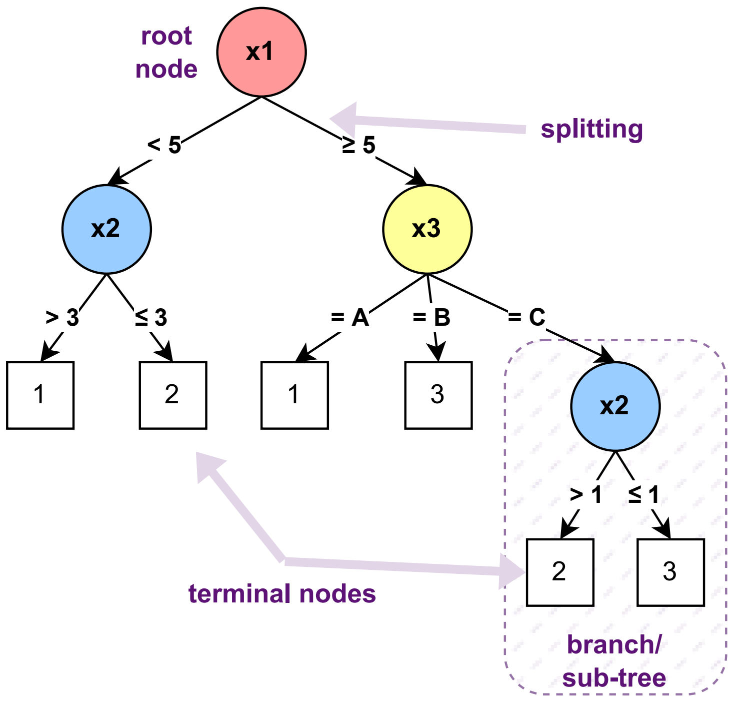
2.2 DECISION TREES

Decision Trees are one of the simplest yet most successful forms of machine learning models. Their operation is quite easy to comprehend since it is a natural way humans handle decision-making: normally, one gathers information around a certain goal or decision to be made, which is then tested against the expected characteristics of the target. For example, we may choose whether to watch a certain film by assessing the available options for specific traits such as being an action and adventure movie, being suitable for children, and having a specific runtime, among others. Similarly, one could then build a DT to filter the films at each node of the tree, classifying them on whether or not they would want to see the film.

A generic example of a Decision Tree is given in Figure 1. The tree traversal is defined by attribute tests at each decision node starting on the root node, and the classification is given at terminal nodes. Each decision node has n branches which may be either subtrees or terminal nodes, that guide classification based on information from the training data and learning method used. For a wide range of problems, the DT format yields satisfactory and concise results, while also operating at considerably lower computational cost when compared to more sophisticated models, such as Neural Networks (NNs) (RUSSELL; NORVIG, 2009) (ABREU; GRELLERT; BAMPI, 2020).

In Machine Learning, our main interest is the automatic construction, or learning, of efficient and accurate DTs from datasets, resulting in trees that can predict or categorize new data in the fewest attribute tests. The following paragraphs give an overview of this topic, though a more in-depth exploration may be found by the reader in Russell

Figure 1 – Example Decision Tree



Source: Pedro Silva, 2022.

and Norvig (2009). Throughout this text, we shall use the words *attribute* and *feature* interchangeably.

Understanding the process of DT learning benefits from basic knowledge of some concepts from Information Theory, specifically entropy and information gain. **Entropy**, represented by the Greek letter *eta* (H), is the fundamental quantity in Information Theory and is defined as the measure of the uncertainty inherent to a random variable's possible outcomes (SHANNON; WEAVER, 1949). In layman's terms, entropy quantifies how much knowledge one has of the variable in question. The entropy of a random variable X with possible values $\{x_1, ..., x_n\}$ and probability mass function $P(X)$ is defined in Equation (4). Note that the unit for entropy calculated with logarithm base 2 is *bits* (or *shannons*).

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \tag{4}$$

In DT learning, our goal is to minimize the entropy for the classes in the data set, finding the configuration of the tree that provides the largest amount of "knowledge" of the data set, that is, the one which correctly categorizes most of our data in the least amount of nodes traversed. In other words, the goal is to find a tree that is consistent with the examples provided as input and is as small as possible, which is an intractable problem, as it would require searching through all possible trees. Instead, we use some heuristics to find a satisfactory solution: Decision Tree Learning algorithms adopt a greedy divide and conquer strategy to find a small, but not necessarily the smallest, tree that is consistent with the input data. These algorithms recursively form sub-trees on each attribute test, always attempting to start from the attribute which makes the most difference to the classification of an example, until all relevant features have been identified and each test has been converted into a node. Algorithm 1 presents the Decision Tree Learning procedure with the heuristics discussed, as described by Russell and Norvig (2009). This is a simplified version of the algorithm capable only of dealing with categorical features. The function PluralityValue selects the most common output value in a set of examples, breaking ties randomly, and the function Importance will be detailed in the following paragraphs.

Algorithm 1 Decision tree learning algorithm.

```
1: function DecisionTreeLearning(examples, attributes, parentexamples) returns
   a tree
2:     if exemples is empty then return PluralityValue(parentexamples)
3:     else if all examples have the same classification then return the classification
4:     else if attributes is empty then return PluralityValue(examples)
5:     else
6:         $A \leftarrow \text{argmax}_{a \in attributes}$ Importance($a$, examples)
7:         tree ← a new decision tree with root test A
8:         for each value $v_k$ of A do
9:             $exs \leftarrow \{e : e \in examples$ and $e.A = v_k\}$
10:            subtree ← DecisionTreeLearning(exs, $attributes - A$, examples)
11:            add a branch to tree with label $(A = v_k)$ and subtree subtree
       return tree
```

Source: Russell and Norvig (2009).

There are two main challenges involved in building decision trees. The first is how to decide on the feature that will be picked to separate the classes of our data set into smaller partitions. In the algorithm presented, this is done in line 6 by using the Importance function, which computes the feature importance of the attribute being evaluated. By always beginning with the most "important" attributes, we hope that the resulting tree is *shallow*, meaning it will provide the correct classification with a small number of tests. Thus, the idea is to find the attributes that go the furthest way in providing an exact classification of the example data. Perfect features would divide the examples into sets consisting entirely of one category, while useless ones would leave the examples with roughly the same proportion of categories found before evaluating the split. The second challenge is deciding if a node should be split further. Perfect leaf nodes would contain instances of a single class, thus it may seem that we should continue splitting until this happens. The problem with this approach is that this is known to create trees that have very large depths, which ultimately causes overfitting. This is why most implementations of DT Learning algorithms will define thresholds to stop splitting even in cases where complete separation is not possible, as well as other mitigation strategies to avoid overfit trees.

To address the first challenge, our example implementation in Algorithm 1 uses the IMPORTANCE function to find the attribute on which the examples will be split. An implementation of this function should calculate a measurable quantity of the feature importance, for instance, the **information gain** from splitting the data set on a specific attribute. In the context of DTs, the information gained from splitting on a given attribute is the reduction in entropy compared to the previous state, when the data set was not yet split. To illustrate this, consider we have a data set S where each example is an array of values for a given set of attributes A. For each attribute $a \in A$, we divide the examples on the possible values of a and evaluate the entropy of the system considering the categories present on each subset. The difference in entropy before and after splitting on attribute a is the *information gain.* The attribute whose test most significantly minimizes entropy in the data set, in other words, the one with maximum IMPORTANCE, is the one that produces the largest information gain, and is the attribute that should be tested in the decision nodes, as shown in Fig. 1. Note that other metrics for importance may be used, such as the **Information Gain Ratio** or **Gini Impurity**.

Testing the split on discrete-valued attributes is straightforward, as the dataset can be trivially divided on each possible value or category. However, we can also build DTs with continuous or integer-valued attributes, which have an infinite set of possible values. Continuous or mixed attribute datasets are abundant, considering most financial and real-world data is continuous (RUSSELL; NORVIG, 2009). For these types of attributes, DT learning algorithms will generally attempt to find the best split value or threshold which yields the highest information gain, i.e. provide the best separation of the dataset on useful leaves.

For such cases, the Decision Node normally implements a comparison with the best split value. This comparison effectively reduces the problem of separating the example data into n possible classes to a binary problem – a given example can now be only one of two classes: lower or equal to the threshold, or greater than the threshold. A consequence of this fact is that the Decision Node for continuous attributes will always have two children, as shown in Fig. 1. Note that the only node in the example that has more than two children, $x3$, is a discrete-valued attribute, with three possible values, while the other two attributes are continuous.

An algorithm to find this threshold value is given in Algorithm 2, inspired by the implementation in the C5.0 classification model, a DT algorithm that will be further discussed below (QUINLAN, J. Ross, 1993), and the description given by Russell and Norvig (2009). Note that this is a simplified version of the algorithm, which is not optimized for large datasets.

Algorithm 2 Method for finding the best split in a continuous feature. The function CalculateInfo returns a measure of the reduction of entropy gained from the evaluated split, e.g. the *information gain*.

```
1: function BestSplit(examples: set, a: attribute) returns a value
2:     $step \leftarrow k | k \in \mathbb{R}$ and $k < \max_a examples$        ▷ define arbitrary step size
3:     $t \leftarrow \min_a examples + size$
4:     $best \leftarrow t$
5:     $info \leftarrow$ CalculateInfo($examples$, $a$)
6:     while $t < \max_a examples$ do
7:         $lower \leftarrow \{e : e.A \leq t\}$        ▷ split the examples with the current threshold $t$
8:         $higher \leftarrow \{e : e \notin lower\}$
9:         $newinfo \leftarrow$ CalculateInfo($lower$, $a$) $\oplus$ CalculateInfo($higher$, $a$)
10:        if $newinfo > info$ then
11:            $info \leftarrow newinfo$
12:            $best \leftarrow t$
13:        $t \leftarrow t + step$
       return $best$
```

Source: Pedro Silva, 2022.

Considering a dataset with the attribute *Temperature* in Celsius and the “yes/no” classification of *Snowing*, we might find that a split value below or close to $Temperature = 0$ provides the best information gain for this goal. Note that, while there are efficient methods for finding good split points, this is typically the most expensive part of DT learning algorithms.

There are various implementations of these ideas that differ mostly on the splitting criterion, the capacity to handle missing data, support for both classification and regression, and their overfitting mitigation criterion. Different versions also have diverging approaches to treating the overfitting discussed in the second challenge. Nowadays, two implementations are mostly used, notably the C5.0 Classification Model, developed by J. Ross Quinlan (1993) and Classification and Regression Trees (CART), developed by Breiman et al. (1984). A comparison of these models on the aspects highlighted is given in Table 1, along with the ID3 algorithm, an earlier DT algorithm proposed by Ross Quinlan, that has mostly historical importance (QUINLAN, J. R., 1986).

C5.0 is an open-source improved version of one of the most popular algorithms for training Decision Trees called C4.5, also developed by Ross Quinlan. On the other hand, CART is an algorithm popularized due to its ability to handle both numerical and categorical data, and known for its implementation in the `scikit-learn` Python library (PEDREGOSA et al., 2011). The version available in `scikit-learn` is an optimized implementation that currently has no native support for categorical attributes. This limitation is mostly overcome by using pre-processing workarounds, e.g. one-hot encoding.

Table 1 – Comparison of Decision Tree algorithms.

	ID3	**C5.0**	**CART**
Splitting criterion	Information Gain	Information Gain Ratio	Gini Impurity
Numeric attributes	No	Yes	Yes
Type of tree	Nonbinary	Nonbinary	Binary
Supports regression	No	No	Yes
Overfitting treatment	None	Post-training pruning	Post-training pruning

Source: Mateus Grellert, via `Jupyter nbviewer`.

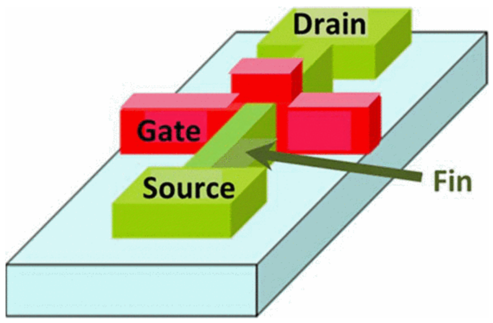
Some methods employ multiple trees, or ensembles, to improve the quality and reliability of tree-based predictors. The most notable ensemble methods are boosting and bagging, while a successful form of using multiple trees in parallel is the use of Random Tree Classifiers. These ensemble-based tree models highlight once again the importance of optimizing DT classifiers for both power and accuracy since their training and usage are much more resource intensive than single-tree models.

2.3 FINFET DEVICES AND ELECTRICAL CHARACTERIZATION WITH SPICE

As part of the efforts to propose approximate comparator architectures, this work describes and evaluates a set of circuits at the electrical level, due to this level being detailed enough to provide reliable data for a comparison between all circuits. Also, electrical level evaluation is less time-consuming and computationally costly than, for example, considering the layout level in a specific technology. It also enables us to use newer device technologies in a very straightforward manner, with simple alterations in the circuit description and different Predictive Technology Model (PTM).

For this evaluation, simulations in 7 nm Fin Field-Effect Transistor (FinFET) node were performed in the electrical simulator HSPICE® from Synopsys, using the ASAP7 Predictive Technology Model (CLARK et al., 2016). FinFETs are multigate non-planar transistors that have various advantages in contrast with the traditional planar transistors, specifically concerning short-channel performance, while allowing for very similar fabrication processes (YU et al., 2002). Since the 2010s, it has been widely used in nanoscale Complementary Metal-Oxide-Semiconductor (CMOS), due to its good characteristics in performance, scalability, and manufacturability, especially under 20 nm process nodes (Y. KAMAL, 2022). A diagram for a FinFET device is given in Fig. 2, where we can see the vertical structure called fin surrounded by the vertical gate. These vertical structures ensure better channel control, high mobility for electrons and holes, and better manufacturability for nanometer nodes.

Figure 2 – Doublegate FinFET transistor



Source: Irene Ringworm, CC BY-SA 3.0, via Wikimedia Commons.

Differently from planar CMOS devices, which have continuous sizing parameters, the sizing in FinFET transistors is discrete due to the vertical fin characteristics. Instead of varying the width of the channel, in these devices, we use several fins placed side-by-side and all covered by the same gate. The device acts electrically as a single gate and has greater drive strength and throughput than planar CMOS gates. Thus, when sizing FinFET transistor, we define only the number of fins present in each device. Specifically, the ASAP7 Process Design Kit (PDK) recommends utilizing the minimum recommended number of fins to ensure net routability, namely three fins per transistor (XU et al., 2017). In the simulations, we found that this recommendation was well suited for our use case, with good performance on all evaluated circuits. To illustrate, the description of an inverter circuit for HSPICE® using the ASAP7 PDK is shown in Listing 1. The nominal voltage for the 7 nm process node is 0.7 V, and the ASAP7 PDK device model chosen is considering the fast-fast corner for PFET and NFET devices.

Once the technology has been decided and the circuits designed, we move to the electrical characterization of the circuit. For the simulations to approximate reality, it is necessary to emulate a scenario in which the circuit is connected to other devices and logic cells. This is because, after the placement and routing of a circuit, there are other blocks connected to both inputs and outputs acting as resistors and capacitors, which interfere in the electrical behavior of the circuit of interest (WESTE; HARRIS, 2010). To

Listing 1 – Inverter circuit description for HSPICE® using the ASAP7 PDK.

```
* PTM and simulation parameters
.include 7nm_FF.pm

.param vdd = 0.7 V
.param len = 7 nm
.param n = 3
.option post = 2

* supply and source voltages
Vvdd vdd gnd vdd
Vin in gnd pulse(0 vdd 1.0n 0.1n 0.1n 1.0n 2.2n)

* circuit description
Mp vdd in out in pmos_rvt L=len nfins=n
Mn out in gnd in nmos_rvt L=len nfins=n

.tran 0.1ns 5.0ns
.end
```

Source: Pedro Silva, 2022.

this goal, a common approach in testing and characterization is to connect a chain of an even number of inverters 4× the minimum sizing to all inputs and outputs of the Device Under Test (DUT). This gives the entry inverter enough current to deal with the DUT input capacitance and provides enough charge for the DUT to be considered equivalent to the Fan-out of Four (FO4) rule, usually employed in combinational cell design. The FO4 rule is defined as the equivalent capacitance of a minimum-size inverter driving an inverter 4x larger than itself. For the simulations in this work, a different approach was used, where we connect the inputs and outputs of the DUT to 1 fF capacitors as load. This also gives an equivalent setup to the FO4 rule and is as realistic as a complex system, reducing the runtime of the electrical simulations.

After having defined the design, and obtained the description of each circuit, we must decide which electrical metrics should be obtained and evaluated. Since this project is focused on the reduction of energy consumption, it makes sense to calculate the metrics of power and delay, whose trade-offs are considered in the Power-Delay Product (PDP).

Firstly, we consider the calculation of power dissipation on the simulation tool used. To this end, HSPICE® only provides direct methods to measure the current, which can then be used to calculate the **average power dissipated** by the circuit in the simulation time frame. As shown in Equation (5), we compute the energy by taking the integral of the measured current from a source voltage $i(Vdd)$ over a time interval Δt, corresponding to the simulation duration.

$$E = \int_0^{\Delta t} i(Vdd)\,dt \tag{5}$$

With the total consumed energy, we apply Equation (6), determining the average power dissipated by the relation of energy over time, multiplied by the supply voltage. In all simulations for electrical characterization, we used the nominal supply voltage of $0.7\,V$, and $3ns$ duration.

$$P(Vdd) = Vdd \times \frac{E}{\Delta t} \tag{6}$$

Timing characterization is helpful to evaluate the performance of logic blocks and can be measured through their delays. For our purposes, only the propagation time of a single signal is considered in the calculation of delay and used in the computation of Power-Delay Product (PDP). The propagation delay indicates the time necessary for an output to change state when there is a single transition in the inputs. These times are measured from 50% of an input wave transition to 50% of the resulting output transition. These times may be from high-to-low when the output goes from a logic 1 to a 0, or low-to-high when the state changes in the opposite direction (WESTE; HARRIS, 2010). To characterize a circuit, we are interested in the largest propagation delay, or **critical delay**. In small circuits, it is possible to obtain this through exhaustive simulation, using input waves called *delay arcs* containing all relevant transitions.

Finally, with both the critical delay and the average power dissipation $P(Vdd)$, the PDP is trivial to compute from Equation (7). This metric is useful for comparing which circuit is more efficient by both metrics. A simplified physical interpretation for the PDP is the energy consumed in transitions of all devices in the design when stimulating the critical path.

$$PDP = Delay \times P(Vdd) \tag{7}$$

As the number of inputs of the circuits increases, this method of characterization becomes prohibitively costly, due to the difficulty in determining the delay arcs, and our strategy needs to be adapted. In this work, it was decided to characterize all logic gates and full adders used, and analytically find the critical propagation path in the circuit diagrams. Then, the critical delay of the logic blocks in the critical path is summed up to find an estimate of the critical delay of the entire circuit. This approach to characterization is only possible, of course, since our interest lies in how each proposed circuit compares with the others. Thus, because we use the same methods and metrics on all evaluated circuits, the values computed will be reliable enough for our use case.

3 RELATED WORK

Tree-based classification models are highly suitable for efficient hardware implementation. Compared to NNs, for example, DTs require the storage of a smaller number of coefficients and present linear computation (TORRES-ALVARADO et al., 2022). Thus, there is a long reported history of hardware implementations, including both Field-Programmable Gate Array (FPGA) designs and silicon integration in Application Specific Integrated Circuit (ASIC) (LI, Q.; BERMAK, 2011). There have been various techniques developed to improve hardware implementations of DTs, most of which require the use of comparators for attribute tests in continuous features.

Since this work is focused on optimizing comparator circuits specifically, the following sections will provide an overview of the state of the art of design and synthesis of DT accelerators as well as low-power and approximate comparator circuits. Firstly, Section 3.1 discusses recent projects advancing the synthesis flow for DT classifiers and highlights the efforts directed towards the approximate implementation of these systems. Section 3.2 discusses various techniques used on the low-power design of $2-bit$ and $n-bit$ comparators and comments on two papers proposing approximate versions of such circuits.

3.1 DECISION TREES AND HARDWARE ACCELERATION

The hardware implementation of DT models is mostly focused on applying these systems in low-resource environments. For instance, Qingzheng Li and Bermak (2011) develop an on-premise DT solution in ASIC for gas identification with 91% accuracy. Since there are various alternative architectures for DT classifiers, early work on this topic focused on evaluating these approaches.

J.R. Struharik (2011) provides insights on the topic of the hardware realization of both single-tree and multiple-tree models (STRUHARIK, R. J. R.; NOVAK, 2013). In both these works, the authors investigate several possible implementations of Decision Trees in hardware. Notably, they explore the fact that there is an equivalence between Decision Trees and a mathematical formulation named *threshold networks*. It is also highlighted that during the classification of an instance using any DT, only a subset of the nodes will be visited, meaning that if we have a single universal node that can evaluate every tree node, the number of hardware node modules required to implement this decision tree is equal to the depth of the tree. Going further, if the classification speed and throughput are not critical requisites, all trees can be implemented with a single universal node, drastically reducing the hardware cost of realizing the DT. These strategies have the downside of requiring much larger storage for the coefficients of the threshold network. According to the authors, the high throughput architecture proposed needs, on average, 56% fewer node modules than previously proposed architectures.

More recent projects have increasingly delved into improving energy consumption for embedded and mobile devices. García-Martín et al. (2021) develop a software-level approach to design energy-aware Very Fast Decision Tree (VFDT) for streaming algorithms requiring up to 31% less energy than the original VFDT while trading off around 1.7% of accuracy. Abreu, Grellert, and Bampi (2020) analyze the effects of model complexity on the power-accuracy trade-offs of hardware implementations, showing that quantization of inputs may have an advantageous impact on accuracy in contrast with precise cases, and evaluating the extent of the impact on power consumption when varying the maximum tree-depth.

Finally, there is some interesting work exploring approximate computing on the synthesis of DTs. Barbareschi, Barone, and Mazzocca (2021) explore a precision-scaling technique, building multiple tree variants and employing a multi-objective optimization algorithm to find optimal configurations targeting energy consumption, area occupation, and accuracy of the final model. Balaskas et al. (2022) have an approach similar to the one developed in this work, except targeting printed electronics. Using a dual-approximation strategy, they employ both precision scaling and comparison-level approximations. The comparators utilized in their architectures are "bespoke architectures", taking a single value as input and comparing it with hardwired coefficients. The precision scaling used is also applied to the threshold value, and since the size of the comparator varies with the hardwired value, they also approximate these thresholds to more hardware-friendly coefficients, which reduces the total area of each comparator.

Table 1 summarizes the similarities and differences between the studies discussed in this section. For each work, we identify the level of evaluation of the design, highlighting the target technology used. We also consider the target applications, if any, although most of the selected research is on general-purpose Decision Tree implementations. The Architecture column details what are the particularities of each design. Note that most works go further than we did, by implementing the entire DT in hardware, while we focus on software-level evaluation, as our objective is to characterize the impact of the comparators used. The AxC column includes any approximation techniques explored by the authors of all projects. Most of them deal with some level of approximation, particularly in the form of precision scaling. Here, our approach differs, as we investigate the approximation on the circuit level, altering the comparators themselves in a fixed precision scenario. Finally, the last column identifies which researches deal with comparisons and the techniques used by them. As discussed previously, out of the selected projects, the only research focusing on the approximation of comparisons is done by Balaskas et al. (2022), although their approach to approximation is on the coefficients used in bespoke comparators and the precision used, while our strategy concerns the comparator itself.

Table 2 – Summary of Decision Tree related work

Work	**Design**	**Target application**	**Architecture**	**AxC technique**	**Comparator techniques**
Qingzheng Li and Bermak (2011)	ASIC 180nm bulk	gas identification	threshold networks	None	None
García-Martín et al. (2021)	software	general purpose	VFTD	software level hyperparameter tuning	None
Abreu, Grellert, and Bampi (2020)	ASIC 64nm bulk	general purpose	automatic tree-to-VHDL translation	Precision scaling, depth scaling	None
Barbareschi, Barone, and Mazzocca (2021)	FPGA 28nm bulk	general purpose	automatic tree-to-VHDL translation	Precision scaling	None
Balaskas et al. (2022)	printed circuit	general purpose	bespoke classifier	Precision scaling, approximate coefficients	bespoke comparators (coefficients-specific design)
This work	**software, ASIC 7nm FinFET**	**general purpose**	**software-level evaluation**	**circuit-level comparator approximation**	**truncated comparator, copy strategy**

Source: Pedro Silva, 2022.

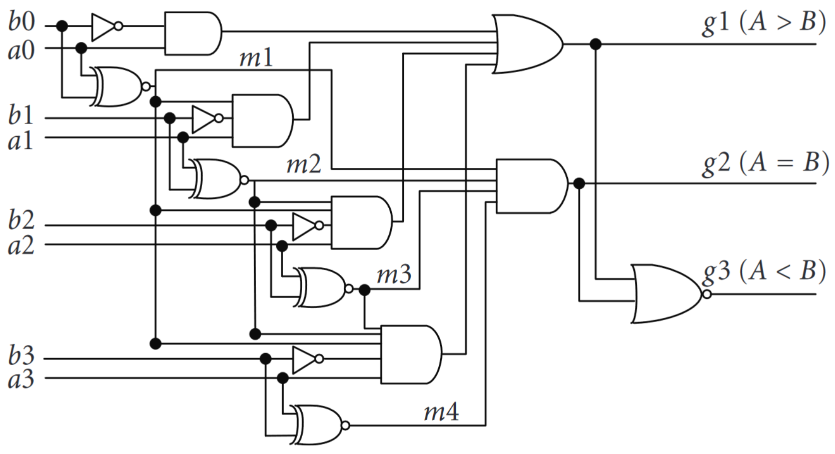
These projects highlight the efforts already taken in the design of DT applications using AxC techniques, in both software and hardware. They also present possible use cases of the approximate comparator approaches developed in this work, most notably in the usage of approximation in different levels of the design flow. For example, our approach could be combined with techniques developed in Balaskas et al. (2022) for the design of an ASIC implementation of a DT employing precision scaling, threshold approximation, and comparator circuit approximation.

3.2 LOW-POWER COMPARATOR DESIGN

The comparison operation is traditionally implemented in two main approaches: using Full Adders or dedicated circuits. Conventional comparators evaluate two $n-bit$ inputs and return three outputs: *greater than*, *equal to*, and *lesser than*, that may in turn be combined to form other desired outputs. Full Adder-based comparators are built using Full Adders (FAs) and traditional adder modules. The performance of FA-based comparators can be improved using architectures focused on power reduction or reduced delay. They also have the benefit of hardware reusability for applications that already require adder-subtractor modules, as well as good scalability for large inputs, by simply increasing the size of the adder used.

Despite the good performance in large bit widths and the incentive of hardware reuse in FA-based comparators, dedicated circuits are known to achieve further energy consumption and performance gains. The logic diagram for the conventional 3-output implementation of a dedicated comparator is given in Fig. 3.

Figure 3 – Logic diagram of a 3-output exact $4-bit$ dedicated comparator



Source: Younis and Zamli (2010).

Most work regarding low-power comparator design is centered on improving the electrical characteristics of a circuit without impacting its accuracy. In fact, despite the ubiquity of these logic blocks in digital design, to the best of the author's knowledge, the research on ultra-low-power comparators has been primarily concerned with applications requiring comparators as control blocks or for analog to digital converters.

There is a lot of interest in the design of 2-bit comparators, being the focus of most circuits the author encountered in the literature. These circuits can be chained to produce larger comparators since they provide all three possible outputs: equal to, greater than, and lesser than. Akash Gupta et al. (2017) propose a 2-bit low-power comparator topology developed in 90 nm CMOS process node by introducing what the authors refer to as a "swing domino logic with twisted transistors" to decrease dynamic power dissipation. This change results in savings of up to 60% in power consumption. Vallabhuni et al. (2020) design a comparator in 18 nm FinFET technology node, which is then analyzed on nominal and low voltage to evaluate transient response, as well as power and delay measures.

Moving to $n-bit$ comparators, Geetanjali Sharma, Nirmal, and Misra (2011) also propose an $8-bit$ Hybrid PTL comparator, removing the branches which compute lesser than and equal to and using low-power Full Adders (FAs) to provide energy savings in comparison with conventional dedicated comparators. This is possible due to the property that, given two numbers A and B, $\neg(A > B) \implies A \leq B$. Exploring this fact, Pranay

Singh and Jain (2018) and Munratiwar, Saikrishna, and Jyothi (2022) present very similar designs which simplify the logic required for computing the less than operation by reusing the branches responsible for the other two operations. This technique of reusing the same hardware to perform different comparisons with the help of additional logic is fairly common in $n-bit$ comparator design, and will also be used in our approach. Efstathiou, Agalioti, and Tsiatouhas (2022) propose a design based on dynamic logic, which shows good results in PDP in up to a 64-bit version. Another possibility in low-power $n-bit$ circuits is presented in Tripathy et al. (2015), where a hybrid logic consisting of Pass Transistor Logic (PTL), transmission gates, and static CMOS logic is used to explore larger levels of parallelism in comparator design.

Most research in this area is focused on developing low-power comparators with techniques that maintain accuracy, preserving the generality of novel topologies. However, there are some studies considering the design of approximate comparators. Most notably, the work developed by Kim, Yong Zhang, and Peng Li (2015) proposes a FA-based approximate comparator unit for large bit widths in Register-Transfer Level (RTL) that performs 18× more efficiently in terms of Energy Delay Product (EDP) than an equal bit width Ripple Carry Comparator (RCC) while presenting less than 0.1% ER. Building on this comparator, Zhou et al. (2018) design an architecture replacing the FAs in the prior design with comparator logic blocks. The resulting architecture is 1.5× more efficient in Energy-Delay-Error Rate Product (EDERP) than the previous version when both are implemented in TSMC 90 nm bulk CMOS.

The selected work in this section describes various techniques for the design of low-power comparators and is summarized in Table 3. We indicate the bit width of the proposed comparators, explicitly indicating the generic ones, i.e., the techniques extendable to $n-bits$. The column # Outputs summarizes if the proposed comparator circuit presents individual outputs for the three comparative functions, or if it is simplified to provide only one output function, e.g. only the greater-than operation. We also classified the comparators by the type of circuit: FA-based or dedicated. We include the techniques adopted in the conception of the circuits and the abstraction level of the design. In this column, the CMOS refers to the CMOS topology, with topological complementary networks.

In particular, the evaluation of the related work shows that for certain applications, it is unnecessary to compute all three outputs conventionally present in a comparator, since we can provide good results in the area, power dissipation, and delay by reducing the comparator circuit to a single branch that computes either the greater-than or lesser-than operation, and adopt the necessary comparisons accordingly. The designs dealing with

Table 3 – Summary of low-power comparator design related work

Work	Bitwidth	AxC	# Outputs	Type	Technique	Evaluation Level	Technology
Geetanjali Sharma, Nirmal, and Misra (2011)	8	no	1	FA-based	Hybrid PTL/CMOS	electrical	90 nm planar bulk CMOS
Akash Gupta et al. (2017)	2	no	3	dedicated	CMOS, logic coupling	electrical	90 nm planar bulk CMOS
Vallabhuni et al. (2020)	2	no	3	dedicated	basic CMOS	electrical	18 nm FinFET
Pranay Singh and Jain (2018)	4	no	3	dedicated	circuit level, CMOS	layout	180 nm planar bulk CMOS
Munratiwar, Saikrishna, and Jyothi (2022)	8	no	2	dedicated	logic simplification, RTL	FPGA/RTL	FPGA (Xilinx) †
Efstathiou, Agalioti, and Tsiatouhas (2022)	n	no	3	dedicated	dynamic logic	electrical	90 nm planar bulk CMOS
Tripathy et al. (2015)	8	no	3	dedicated	PTL, static CMOS, transmission gate	schematic, layout	45 nm planar bulk CMOS
Kim, Yong Zhang, and Peng Li (2015)	n	yes	1	FA-based	RTL	std cells, electrical	90 nm planar bulk CMOS
Zhou et al. (2018)	n	yes	1	dedicated	logic gate pruning, RTL	std cells, electrical	90 nm planar bulk CMOS
This work	**n, 8**	**yes**	**1**	**both**	**circuit level, CMOS**	**electrical**	**7 nm FinFET**

† Technology not informed by the authors.

Source: Pedro Silva, 2022.

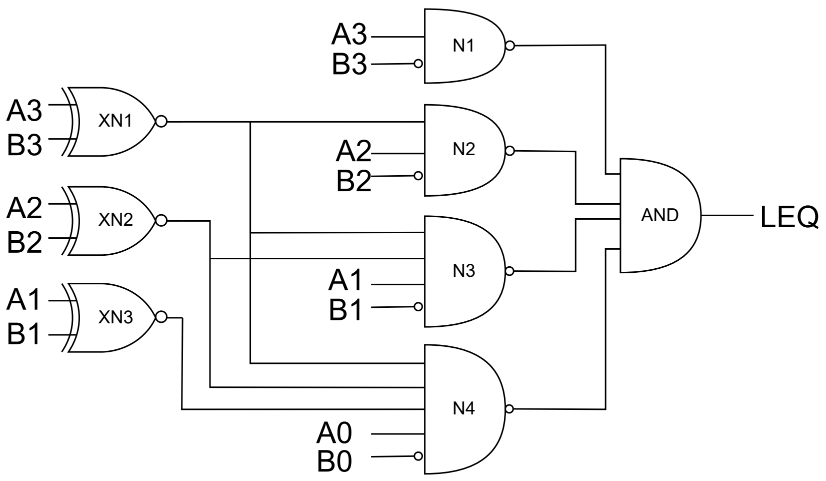
AxC demonstrate the growing need to explore the combination of conventional low-power design techniques with approximate computing methods in comparator design for error-tolerant applications. Among the techniques exploring AxC, our approach differs from the previous ones by proposing new dedicated circuits using the truncation and copy strategies and comparing them with FA-based comparators.

4 PROPOSED APPROXIMATE COMPARATORS

This work proposes new approximate comparator circuits. These circuits are reduced versions of a conventional dedicated comparator (HOLDSWORTH; WOODS, 2002), showed in Fig. 3. The proposed circuits are also compared with FA-based comparators.

In this work, the Exact Dedicated Comparator (EDC) was designed by removing the *lesser-than* and *equal-to* branches of a traditional comparator and inverting the *greater-than* output, resulting in an architecture that computes only *lesser-or-equal-than*. This comparator design is an adapted version of the architecture proposed by Geetanjali Sharma, Nirmal, and Misra (2011). This reduction targets improvements in the area and, consequently, power consumption, without any impact on the accuracy of the circuit. We note that the other comparison operations can still be implemented by switching the input order or inverting the output.

Figure 4 – 4-bit
Exact Dedicated Comparator (EDC)



Source: Pedro Silva, 2022.

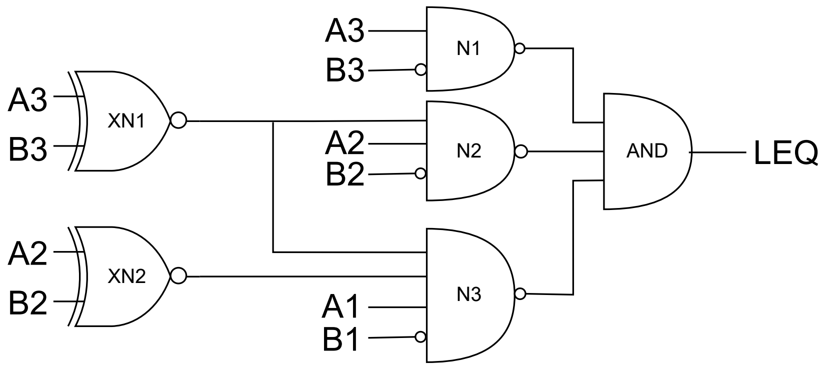
The general equation for $n - bits$ of the EDC is presented in Eq. (8), comprised of three parts. The first line performs an equality test for both inputs in the form of an XNOR operation, producing each EQ_i; the terms $\overline{G_i}$ represent a larger than test in each bit, accounting for all equality tests in more significant bits; and the last line computes the operation $A \leq B$ with an AND operation over G_i.

$$\begin{aligned} EQ_i &= A_i \odot B_i & , 0 < i < n \\ \overline{G_i} &= \overline{[\prod_{k=i+1}^{n-1} EQ_k] \cdot (A_i \cdot \overline{B_i})} & , 0 \leq i < n \\ A \leq B &= \prod_{i=0}^{n-1} \overline{G_i} \end{aligned} \tag{8}$$

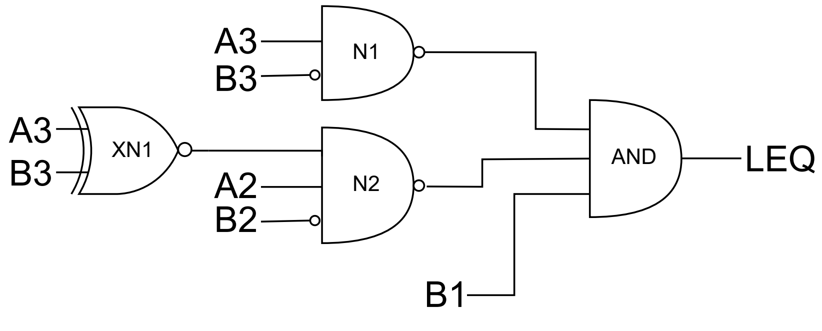
To exemplify, a $4-bit$ version of this exact dedicated comparator is presented in Fig. 4. The diagram structure follows the same idea as that of $n-bit$ comparators. This circuit serves as the starting point for the proposed dedicated approximated comparators.

Considering the EDC, we propose two architectures for Approximate Dedicated Comparators (AxDCs): the Approximate Dedicated Comparator 1 (AxDC1) and the Approximate Dedicated Comparator 2 (AxDC2). The functional approximation technique adopted removes logic gates from the EDC circuits. This approach is explored to improve power consumption while attempting to have little impact on the application accuracy. To illustrate, we present the $4-bit$ versions of both comparators in Fig. 5, as a simplification to introduce the approximations and to allow a direct comparison with the $4-bits$ EDC example.

Figure 5 – Approximate Dedicated Comparators (AxDCs)



(a) 25% inputs approximation (AxDC1)



(b) 50% inputs approximation (AxDC2)

Source: Pedro Silva, 2022.

The first approximate dedicated comparator we will discuss is the AxDC1. The general equation for $n - bits$ is presented in Eq. (9), where the difference from the Eq. (8) is that the range of operation is reduced to $\frac{3}{4}n$ in each input. In other words, the approximation was achieved by truncating the $\frac{n}{4}$ Least Significant Bits (LSBs). In the $4 - bit$ example, this corresponds to removing gates $N4$ and $XN3$ gates from the EDC.

$$[ht] \quad \begin{aligned} EQ_i &= A_i \odot B_i && , \tfrac{n}{4} < i < n \\ \overline{G_i} &= \overline{[\textstyle\prod_{k=i+1}^{n-1} EQ_k] \cdot (A_i \cdot \overline{B_i})} && , \tfrac{n}{4} \leq i < n \\ A \leq B &= \textstyle\prod_{i=\frac{n}{4}}^{n-1} \overline{G_i} \end{aligned} \tag{9}$$

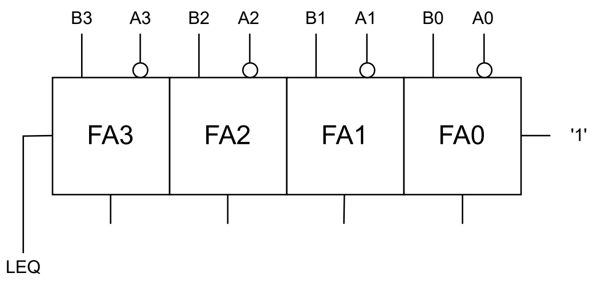
The AxDC2 implementation is shown in Eq. (10). This version limits the range of operation to the $\frac{n}{2}$ Most Significant Bits (MSBs) in each input, also ignoring the $\frac{n}{4}$ LSBs. On the $4 - bit$ example, the gates $XN2$ and $N3$ are bypassed and removed from the EDC circuit. The bit $B1$ was passed directly to the last AND gate, an approximation technique we refer to as the "copy strategy", that improved the error rate for this circuit in comparison with directly truncating 50% of the inputs. This resulted in further reduction in area and possibly power compared to the AxDC1.

$$\begin{aligned} EQ_i &= A_i \odot B_i && , \tfrac{n}{2} < i < n \\ \overline{G_i} &= \overline{[\textstyle\prod_{k=i+1}^{n-1} EQ_k] \cdot (A_i \cdot \overline{B_i})} && , \tfrac{n}{2} \leq i < n \\ A \leq B &= \textstyle\prod_{i=\frac{n}{4}}^{\frac{n}{2}-1} B_i \ \cdot \ \prod_{i=\frac{n}{2}}^{n-1} \overline{G_i} \end{aligned} \tag{10}$$

As generalized by the Eqs. (9) and (10), these approximations can be explored in $n - bit$ circuits, constructed by obeying the same conditions of approximation. For the experiments, let us consider $8 - bit$ versions of each circuit, analogous to those presented in the schematics for $4 - bits$. To do so, we use slices of 25% of the full bit width instead of individual bits. As an example, AxDC2 would ignore bits 0 and 1 from both inputs, and pass bits $B2$ and $B3$ directly to the last AND gate.

To set a comparative environment for the proposed approximate circuits, we also evaluated three FA-based approximate comparators. Let us consider a RCC. Beginning with an adder-subtractor, we can evaluate a comparison by using a subtraction and analyzing a specific output. For example, it is evident that if $A - B \geq 0$, then $A \geq B$. In two's complement arithmetic, that is equivalent to testing if the sign bit (the MSB in the output) is equal to 0. Conversely, performing $A < B$ is equivalent to the sign bit of the operation $A - B$ being equal to 1. To avoid the need for an inverter in lesser-or-equal comparisons, the most frequent type in our use case, we modify this FA-based approach: instead of evaluating $A - B$ and taking the MSB, we analyze the *carry out* for the MSB when calculating $B - A$. To illustrate, Fig. 6 provides an example of a Ripple Carry Comparator (RCC) for $A \leq B$.

Figure 6 – Ripple Carry Comparator (RCC)



Source: Pedro Silva, 2022.

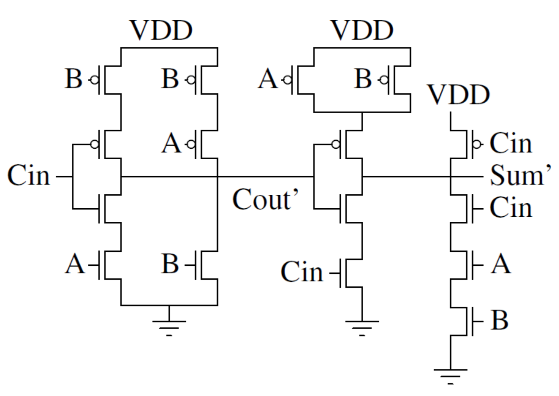
In these circuits, the approximation is trivially inserted by replacing exact FA cells with three different approximate topologies: the Simplified Mirror Adder (SMA), the Approximate Mirror Adder 1 (AMA1), and the Approximate Mirror Adder 2 (AMA2) (GUPTA, V. et al., 2011). The circuits of these approximate FAs are presented in Fig. 7. These AxC FAs are explored due to the good outcomes of lower error distance and power savings compared to the Mirror Adder (MA) (SILVA, P. A.; MEINHARDT, 2020). This approximation has the added benefit of allowing a fine-grained tuning of the level of approximation, by varying the number of bits that will use the AxC FAs and those that shall continue exact sums. For this analysis, we approximated 100% of the FAs in the design.

As for the dedicated comparators, some existing designs provide support for larger input bit widths. However, these are typically implemented following an architectural approach targeting timing optimization (ZHOU et al., 2018). For our purposes, we will consider approximated comparators topologies that have good scalability until 16 bits, insofar as larger bit widths justify analyzing other solutions that reuse hardware or use bespoke comparators with hardwired constants. Nonetheless, generalized $n-bit$ constructions are presented for the dedicated approximations that will be discussed.

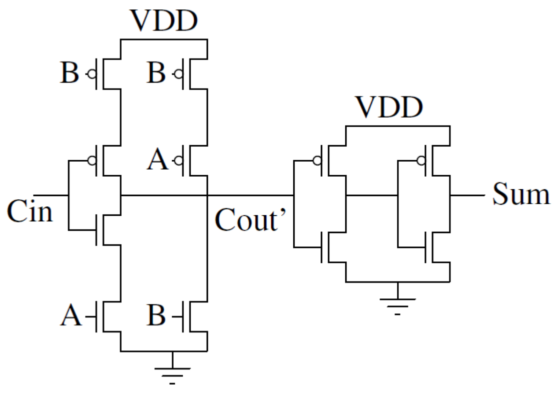
4.1 ERROR AND ELECTRIC EVALUATION OF PROPOSED COMPARATORS

To study the effects of approximation in each comparator, we developed an analysis of their error distribution, error rates, and error distances. This evaluation has the objective of understanding what is the expected behavior of the errors given the ranges of values

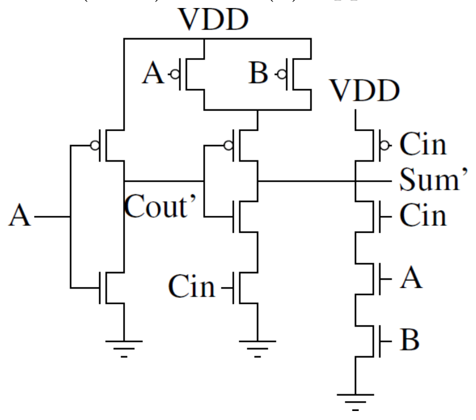
Figure 7 – Approximate FAs inspired by the exact Mirror Adder



(a) Simplified Mirror Adder (SMA)



(b) Approximate Mirror Adder 1 (AMA1)



(c) Approximate Mirror Adder 2 (AMA2)

Source: Vaibhav Gupta et al. (2011)

in a dataset, and how the features are distributed. It also aids in the investigation of the quality of approximations, allowing better-informed decision-making when opting for a specific architecture during the design of a DT system.

The conventional error metrics of Error Distance (ED) and Error Rate (ER), previously defined in Section 2.1, were calculated according to the Truth Tables of the $8-bit$ approximations. Since all comparators have the same bit width, it is more interesting to focus on the ER, though the ED gives a better perspective on the full impact of the approximation. The results for both metrics in all evaluated designs are presented in Table 4. The error distribution, on the other hand, is shown in Fig. 8 and Fig. 9 for the FA-based comparators, and Fig. 11 and Fig. 13 for the dedicated versions. These figures contain heatmaps where the X and Y axes consist of 8 uniformly distributed ranges of A and B values in the operation $A \leq B$. Each cell contains the number of errors for the range of A and B considered and the fraction of the total errors present in the cell. These graphs help us understand the patterns through which the error is distributed on all possible inputs.

Table 4 – Error Metrics for the $8-bit$ Comparator Circuits

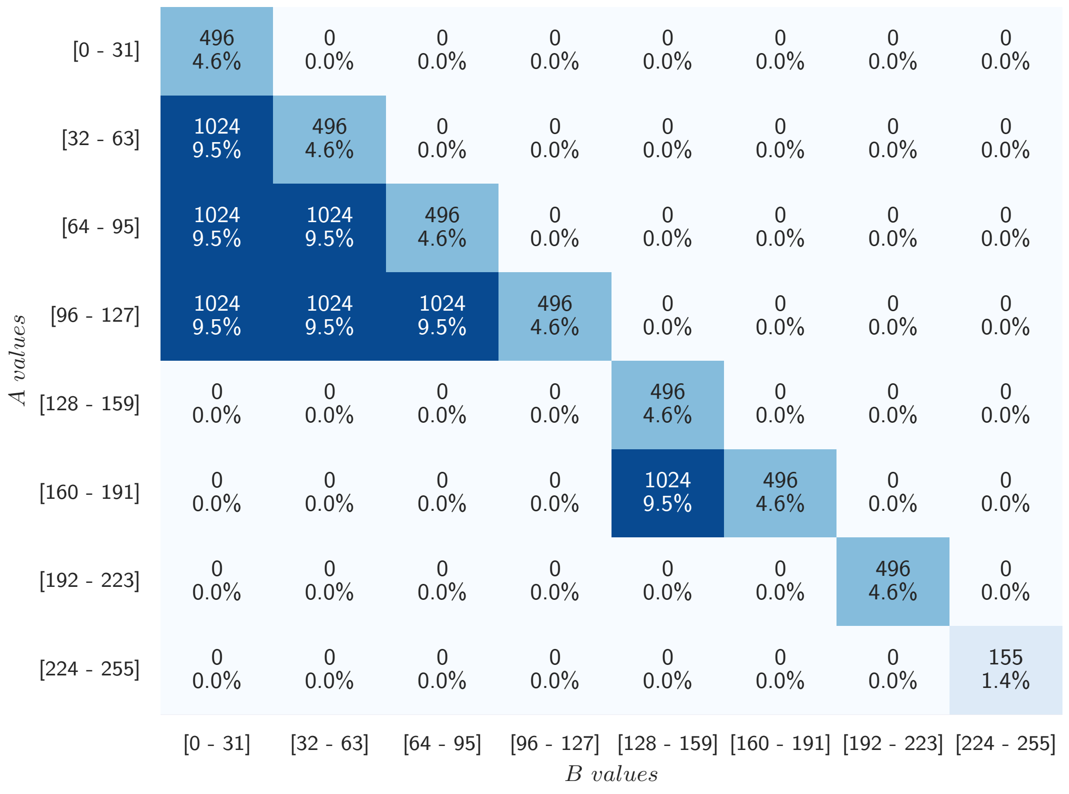
Circuit	**ED** †	**ER (%)**
AxDC1	384	0.59
AxDC2	24,384	37.21
SMA	10,795	16.47
AMA1	10,795	16.47
AMA2	16,384	25.00

† The maximum ED for $8-bit$ comparators is 65,536.

Source: Pedro Silva, 2022.

In Table 4, note that the AMA1 and SMA FA-based comparators present the same metrics, and thus the error heatmap in Fig. 8 is the same for both comparators. This behavior is caused by the fact that the FA-based comparator only depends on the part of the FA circuit that computes the carry-out, which is an identical transistor arrangement for both FA topologies, as seen in Fig. 7b and Fig. 7a.

Figure 8 – Error heatmap for the $8-bit$ AMA1 and SMA-based comparators

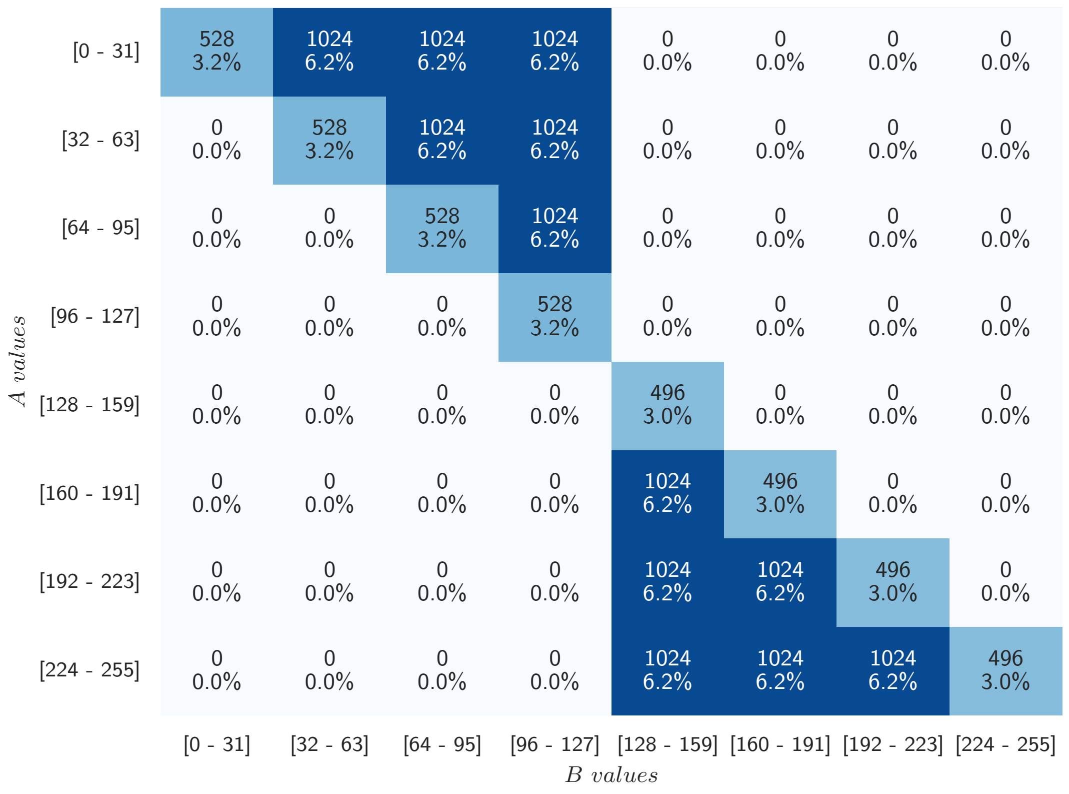


Source: Pedro Silva, 2022.

The AMA2-based version has a problematic distribution of errors. Fig. 9 indicates that not only the AMA2 has the largest ER among the FA-based comparators, as the errors caused by this comparator are much more evenly distributed in contrast to the other FA-based comparators. For the SMA and AMA1 designs, the errors are mostly concentrated in lower values of A and B, while in the AMA2, the error distribution gives no indication of which range of values of a feature might incur in an incorrect classification.

As previously discussed, a fine-grained approximation strategy could be used to improve the error behavior in FA-based comparators, by limiting the approximate FAs to the LSBs and, for example, use an optimization algorithm to find the most appropriate number of approximated bits in a given application.

Figure 9 – Error heatmap for the $8-bit$ AMA2-based comparator

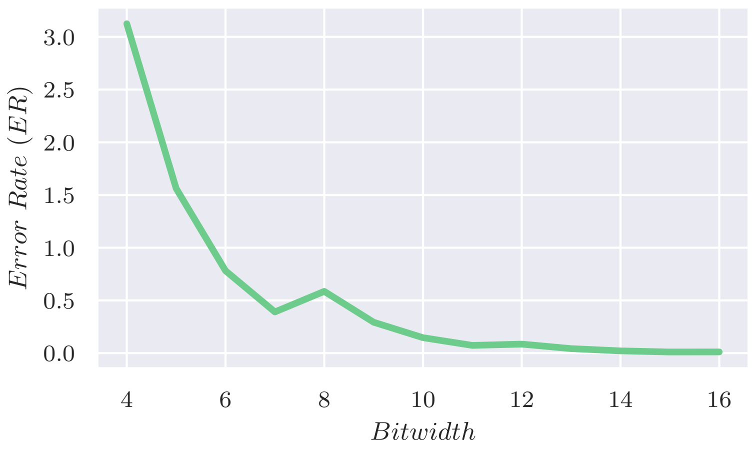


Source: Pedro Silva, 2022.

Another interesting fact evident in Table 4 is the remarkably low error rate in the AxDC1. Despite the design using truncation of 25% of input bits, the Error Rate (ER) is only 0.59%. See that the ER for this circuit decreases exponentially with the number of bits, as shown in 10.

In Fig. 11, we can also see that all the errors are concentrated in the principal diagonal of the heatmap matrix, that is, the region where A and B values are the closest. For DT applications, understanding this behavior is very desirable, thus a detailed heatmap for the ranges of [0,32] is given in Fig. 12. Note that all other ranges follow the same pattern. When considering an attribute test with this circuit, the incorrectly classified examples would only be the ones where values are the closest to the threshold. By contrast,

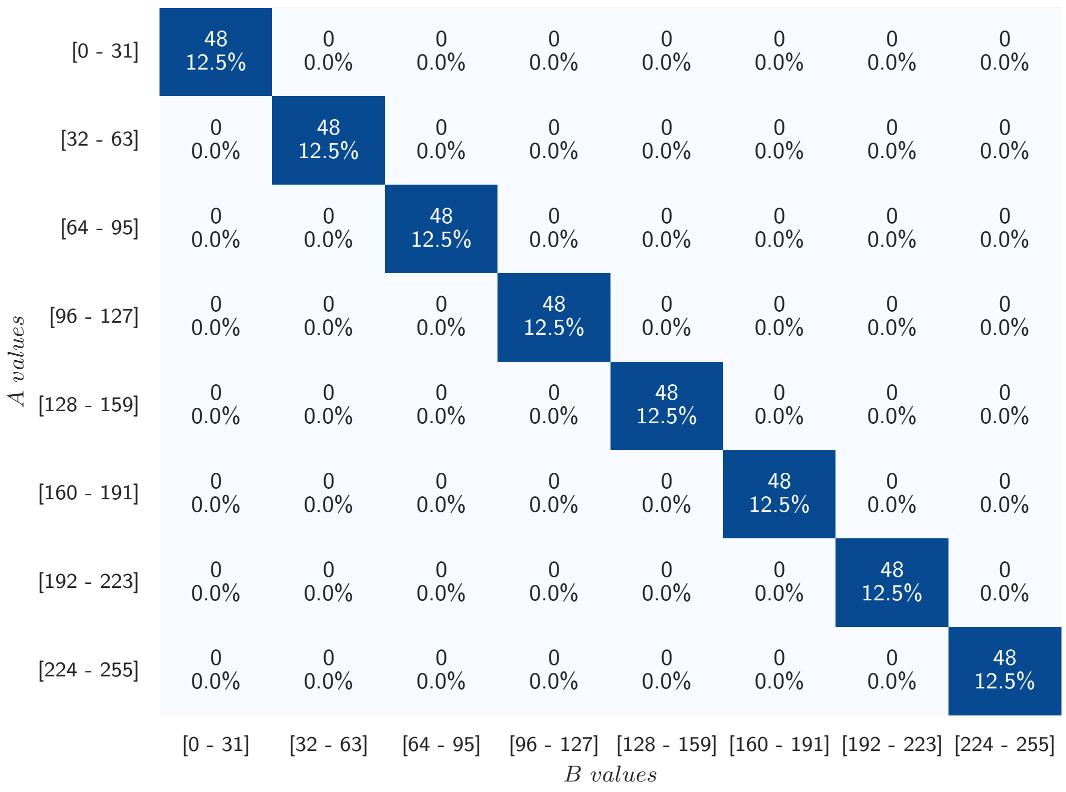
Figure 10 – Evolution of the error rates for the AxDC1 design



Source: Pedro Silva, 2022.

a designer would need to be careful that the target dataset has less frequent examples in the vicinity of the split to avoid an increased loss of accuracy from this comparator.

Figure 11 – Error heatmap for the $8 - bit$ AxDC1

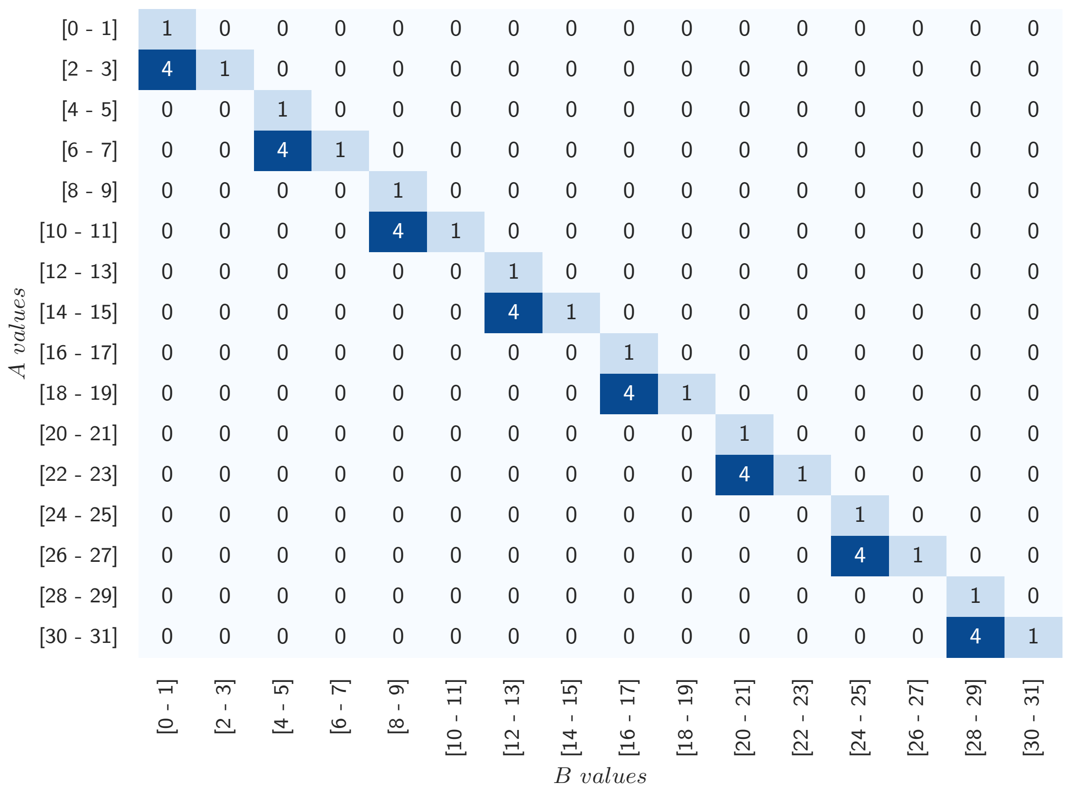


	[0 - 31]	[32 - 63]	[64 - 95]	[96 - 127]	[128 - 159]	[160 - 191]	[192 - 223]	[224 - 255]
[0 - 31]	48 12.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%
[32 - 63]	0 0.0%	48 12.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%
[64 - 95]	0 0.0%	0 0.0%	48 12.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%
[96 - 127]	0 0.0%	0 0.0%	0 0.0%	48 12.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%
[128 - 159]	0 0.0%	0 0.0%	0 0.0%	0 0.0%	48 12.5%	0 0.0%	0 0.0%	0 0.0%
[160 - 191]	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	48 12.5%	0 0.0%	0 0.0%
[192 - 223]	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	48 12.5%	0 0.0%
[224 - 255]	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	48 12.5%

Source: Pedro Silva, 2022.

Finally, we discuss the AxDC2 comparator. From Table 4, the large impact the truncation had in its error metrics is unmistakable, resulting in the worst figures in ER

Figure 12 – Error heatmap for the 8 − *bit* AxDC1 in the range [0,32]



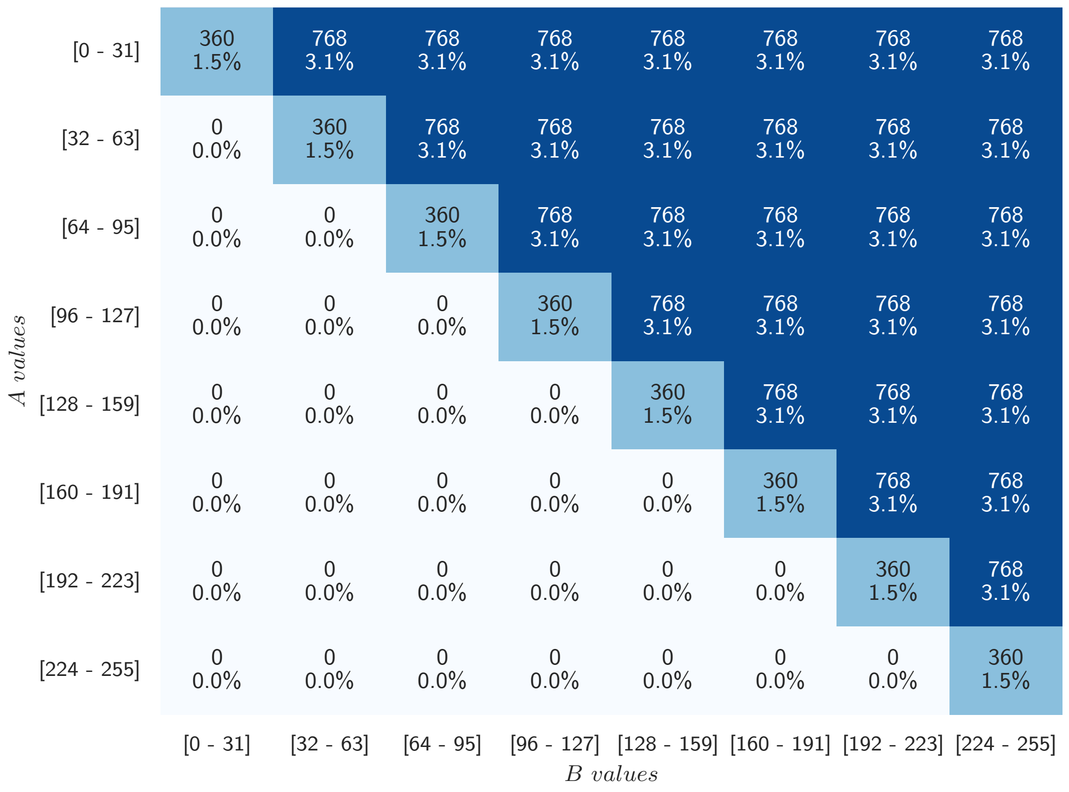
	[0 - 1]	[2 - 3]	[4 - 5]	[6 - 7]	[8 - 9]	[10 - 11]	[12 - 13]	[14 - 15]	[16 - 17]	[18 - 19]	[20 - 21]	[22 - 23]	[24 - 25]	[26 - 27]	[28 - 29]	[30 - 31]
[0 - 1]	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[2 - 3]	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[4 - 5]	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
[6 - 7]	0	0	4	1	0	0	0	0	0	0	0	0	0	0	0	0
[8 - 9]	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
[10 - 11]	0	0	0	0	4	1	0	0	0	0	0	0	0	0	0	0
[12 - 13]	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
[14 - 15]	0	0	0	0	0	0	4	1	0	0	0	0	0	0	0	0
[16 - 17]	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
[18 - 19]	0	0	0	0	0	0	0	0	4	1	0	0	0	0	0	0
[20 - 21]	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
[22 - 23]	0	0	0	0	0	0	0	0	0	0	4	1	0	0	0	0
[24 - 25]	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
[26 - 27]	0	0	0	0	0	0	0	0	0	0	0	0	4	1	0	0
[28 - 29]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
[30 - 31]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	1

Source: Pedro Silva, 2022.

and ED. Combined with the distribution of errors in Fig. 13, we expect this architecture to cause a large degradation of the classification accuracy. Still, the distribution is more well-behaved than in other adders, with most errors concentrated on larger figures of B, likely a result of the copy strategy used, where 25% of magnitude tests were replaced directly with the bits this input.

Moving on to the electrical evaluation of the comparator circuits, we start with the description of the EDC and the proposed circuits for 8 − *bit* extensions of the EDC and AxDCs, as well as the RCC with each approximated FA topology. As described in Section 2.2, the circuits were described at the transistor level using the ASAP7 7 nm FinFET PTM provided by Arizona State University (ASU) (CLARK et al., 2016). All devices use 3 fins, recommended as the minimum sizing for standard cell design. Due to the large number of inputs in 8 − *bit* comparators, the critical delay was analytically estimated, using the critical path in the circuit and each logic block delay characterization. The power was estimated by a worst-case analysis, taking the sum of the average power consumption measured for all logic blocks impacting the critical path in the design. For each cell, the delay and average power dissipation were extracted through exhaustive simulation with HSPICE® from Synopsys. The circuits were characterized under a nominal voltage of 0.7

Figure 13 – Error heatmap for the $8-bit$ AxDC2



Source: Pedro Silva, 2022.

V, utilizing a 1 fF capacitor as load, equivalent to a fan-out of 4 (FO4), to emulate a more realistic scenario.

Alongside the electrical characteristics discussed, another figure of merit is introduced in this evaluation, to give insight into the trade-offs between power dissipation, delay, and accuracy of the proposed circuits. The Power-Delay-Error Rate Product (PDERP) is calculated through Eq. (11), where P_{avg} is the average power dissipation.

$$PDERP = P_{avg} \times Delay \times ER \tag{11}$$

The idea behind this metric is that we desire the simultaneous reduction of all metrics: power, delay, and error rate. In the face of the impracticality of this goal, we become interested in minimizing their collective impact, since an approximated circuit is only energy-efficient if it has a good trade-off of power and delay, usually measured through the PDP, and is only usable if the impact in accuracy can be absorbed by the target application. Therefore, despite the rule-of-thumb of “the lower the PDERP the better” being somewhat reasonable, it is necessary to keep in mind that this analysis is not a substitute for a more thorough evaluation in the context of the application.

Table 5 presents the electrical characteristics of delay and power for the $8-bit$

comparator circuits evaluated in this work, alongside the trade-offs measured through the PDERP. It also includes the transistor count for the implementations used and recalls the ER to aid the analysis. The first line identifies the EDC, adopted as the baseline in the following discussions. The next two lines describe the results for the approximate dedicated versions (AxDC1 and AxDC2) proposed in this work. The last three lines are the results for the subtractor-based comparator circuits with approximation (SMA, AMA1 and AMA2).

Table 5 – Electrical Characteristics and Error Trade-Off for the $8-bit$ Comparator Circuits

Circuit	**# Devices**	**Delay** (ps)	**Power** (nW)	**ER** (%)	**PDERP** ($ps \times nW \times \%$)
EDC	150	79.45	3,421.23	N/A	N/A
AxDC1	102	60.25	3,109.73	0.59	1,097.75
AxDC2	66	44.05	1,700.38	37.21	27,868.74
SMA	176	104.56	4,319.44	16.47	74,393.80
AMA1	120	75.60	5,278.00	16.47	65,725.65
AMA2	136	69.76	3,346.48	25.00	58,362.61

Source: Pedro Silva, 2022

From Table 5, we note that the dedicated comparators have an overall better performance in most metrics when compared to the FA-based versions. Considering the delay, this is a consequence of the linear increase in delay for FA-based versions, since they depend on the evaluation of the entire carry chain. In contrast, the depth of the critical path in dedicated comparators, and consequently the delay, is mostly constant. Out of the evaluated architectures, the smallest delay was obtained in the AxDC2, due to the critical path depending only on the evaluation of $\frac{n}{2}$ bits. For a similar reason, the AxDC1 presents the second lowest delay time.

When evaluating the power consumption, the worst-case analysis performed also puts the dedicated comparators at an advantage. The reason for this is that they were designed by pruning all unnecessary circuits in this comparison, while the FAs used in the remaining designs have additional circuitry to compute the *Sum* output. Focusing on the dedicated designs, the gains in power consumption are consistent with what was expected. The AxDC2 dissipates about half as much power and is nearly twice as fast as the exact version. On the other hand, the AxDC1 provides a somewhat modest improvement in power, close to 10%, although it has almost 25% lower delay time. This is expected, since removing the last 25% LSBs, representing the AxDC1, results in a proportionately smaller number of logic gates removed than the 50% approximation applied to the other comparator. This is also observed in the number of devices in both cases: the AxDC1 had a reduction of 31% in the transistor count, while the AxDC2 uses 56% fewer transistors than the exact version. From the circuit diagrams in Fig. 4 and 5, it is clear that the logic is simplified in the AxDC2: the 5-bit NAND gate present in the EDC was only reduced to a 3-bit in the AxDC1. The AxDC2 removed it completely, impacting the static power dissipation as well.

Considering the FA-based comparators, note that despite having a lower transistor count, the AMA1 has a higher power consumption than the AMA2. Recalling on the diagrams in Fig. 7b and Fig. 7c, one finds that this is caused, in part, by the higher delay times measured for the AMA1 *Cout*, as this signal is used to drive more internal transistors compared to the AMA2 design. Since the power estimate given in Table 5 was calculated by adding n times the worst-case power dissipation for each FA cell, it is understandable that the AMA1 show a larger power consumption. As to why the AMA1 still has a larger power consumption than the SMA-based comparator, we elicit the possibility that the latter circuit has a smaller dynamic power consumption. This would still need to be further evaluated in a more in-depth analysis.

Although it is important to properly understand both the error and electrical characteristics of approximated circuits, it is vital to provide a way to evaluate the trade-offs of the improvements in energy efficiency and the impact of approximation on the accuracy of the results. To this end, let us turn our attention to the last column presented in Table 5, referring to the PDERP. Once again, we observe that the dedicated circuits outshine the FA-based designs. However, in this case, this advantage is caused by the extremely low error in the AxDC1 and the drastic reduction in power dissipation in the AxDC2. Out of the FA-based versions, the AMA2-based comparator has the best results in PDERP, although an investigation as to whether the application tolerates the 25% ER is still of key importance.

The approximate comparators discussed in this chapter were designed using various techniques. The AxDC1 shows a rather conservative approximation, with minimal impact on the accuracy, while the remaining circuits are more aggressive in their error characteristics. The analysis provided is useful to guide the decision of which AxC comparator is best suited for an application, as well as to provide a basis for better understanding the results of employing them in a comparative evaluation, as shall be done over the next sections.

5 USING APPROXIMATE COMPARATORS IN DECISION TREES CLASSIFIERS

To evaluate the proposed approximate comparator circuits, this analysis uses the General Public License (GPL) version of the C5.0 Decision Tree implementation as a case study (QUINLAN, J. Ross, 1993). The GPL version of the C5.0 implementation is available at Rulequest Research, with the limitation of being single-threaded and lacking some optimizations done in the proprietary version. The C5.0 implementation is a well-known and widely used implementation of the C4.5 algorithm (QUINLAN, J. Ross, 1993). The main reason for selecting this implementation is that it is open source and available for free, independent of a larger package, unlike, for example, the `scikit-learn` tree-based classifiers. This allows us to easily understand and adapt the code to include modules describing the proposed comparators' behaviors and perform a software-level evaluation.

For continuous attributes, DTs algorithms typically compare their value with a threshold that is tuned in the training step. However, for categorical attributes, different implementations may have diverging philosophies. For instance, a common strategy when using the `scikit-learn` version of CART is mapping the categorical features into numerical values and testing them against a threshold. In this case, the comparator approximation developed in this work could be applied to all types of features present in the dataset. In contrast, C5.0 maps their value as indices to an array of sub-trees, forming the branches of a node. Therefore, in this study, our approximate operators are only applicable to continuous features.

The comparison operations performed in the classification step were modified in the C5.0 source code to analyze the behavior of the classifiers in the presence of approximate attribute tests, enabling both precise and approximate comparators. On the other hand, other operations such as the computation of the entropy and the information gain were not altered, resulting in an approximation limited to the classification step of the DT application. That is to say that the training step is not aware of the approximation, and the DT is trained with the original C5.0 implementation.

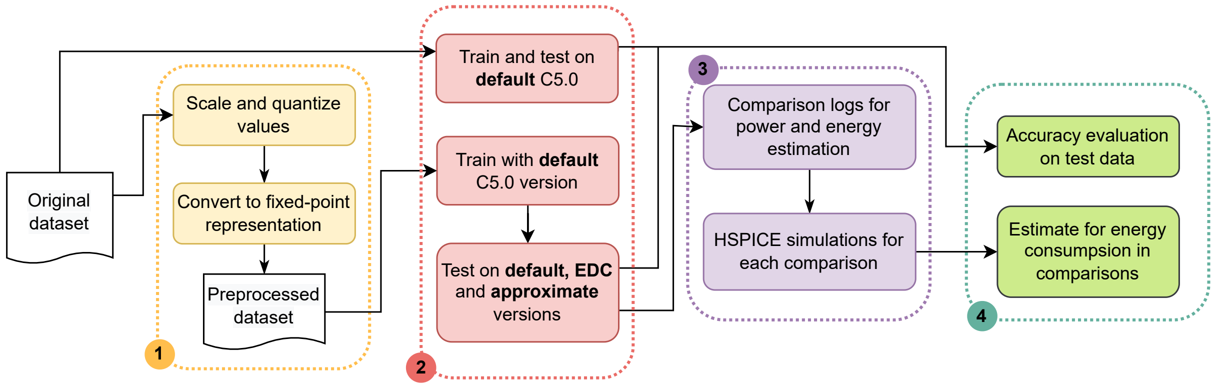
We developed an automated framework to study the utilization of approximate comparators in DT models, which evaluates the accuracy of all proposed comparators on various datasets and provides an estimate of the energy consumed and power dissipation per inference performed. This workflow gives the basis for an in-depth analysis of the trade-offs of power and accuracy in a given model, specifically the C5.0 classifier, using the various approximate comparators discussed proposed. However, we note that the proposed workflow and comparators are easily adaptable to other tree-based classification models.

In the following sections, we further discuss the evaluation of using the comparators in DT applications. Section 5.1 describes the workflow used to evaluate each inexact version of the C5.0 classifier and compare their trade-offs against the version using the EDC for attribute tests. In Section 5.2, we delve deeper into two quantization strategies investigated to convert the numerical features in the evaluated datasets to $8-bit$ values that can be used in our approximate comparators. Lastly, the results obtained from applying the methods in the workflow on different numerical and mixed-attribute datasets are presented in Section 5.3. Section 5.4 this chapter with an examination of the results and limitations of the proposed methods.

5.1 THE WORKFLOW

This experiment consisted in implementing a method for the evaluation of each architecture in the context of the C5.0 classification. The workflow is summarized in Fig. 14 was comprised of 4 stages: (1) pre-processing, (2) model training and classification, (3) power/energy estimation, and (4) summary and evaluation of results. Each step of the workflow developed is detailed in the following subsections.

Figure 14 – Workflow for the evaluation of AxC comparators evaluation in C5.0


Original dataset
Scale and quantize values
Convert to fixed-point representation
Preprocessed dataset
1
Train and test on **default** C5.0
Train with **default** C5.0 version
Test on **default, EDC** and **approximate** versions
2
3
Comparison logs for power and energy estimation
HSPICE simulations for each comparison
Accuracy evaluation on test data
Estimate for energy consumption in comparisons
4

Source: Pedro Silva, 2022.

Pre-processing

In this study, we employed $8-bit$ comparators, considering them large enough to maintain performance in accuracy while providing good results in energy savings. Due to the fixed bit-width in the comparators, a quantization pre-processing stage (step 1, in Fig. 14) becomes necessary. This step prepared the datasets for use as inputs of the unsigned $8-bit$ versions of the comparators proposed, by mapping the values in the dataset to the range $[0, 255]$ and then truncating them, a technique we refer to as *scaling*.

During the development of the flow, another pre-processing technique was considered. This technique is detailed in Section 5.2, confronting the impact on the accuracy with the scaling approach. For now, it suffices to say that our tests indicated that scaling was better suited to our use case. To guarantee that all other changes in accuracy were solely caused by approximation in the comparators, we decided to apply only the scaling in the pre-processing step.

Model training and classification

The model training and classification stage (step 2, in Fig. 14) consisted of running a full classification for each selected dataset. In C5.0, after the tree is constructed, it is validated on two separate sets of data. Firstly, the training set is used to validate the tree, and then the test set is used to validate the accuracy of the tree. This way, we can compare the accuracy of the original classifier with the accuracy of the approximated classifiers through the outputs obtained for the test set of each approximate version of the C5.0 model.

To approximate the C5.0 algorithm, we need to understand exactly how the decision nodes are evaluated. In the source code, the classification of a case is done by a function called `FindLeaf`, starting from the root node of the tree. The C code for this function is shown in Source code 2.

Here, when the node is found to be a leaf, the classification is performed, and the function returns with the class label. Otherwise, the function calls itself recursively, passing the next node to be evaluated, depending on the type of feature the decision node is testing and the result of the attribute test. In continuous attributes, the `FindLeaf` function calls the `Interpolate` function in line 12, which in turn is responsible for computing the weights of the branches, as shown in the rather convoluted Source code 3. The function receives the node to be evaluated and the value of the attribute to be tested and returns the weight of the branch that should be followed, depending on the value of the attribute. The weight of a branch is binary or the interpolation of the threshold value of the node with the value of the attribute. In our tests, we always used deterministic thresholds, thus it is known that the function always returns either zero or one.

Considering the implementation details discussed, we can approximate the classification of a constructed tree by replacing the comparisons made inside the `Interpolate` function body with the corresponding approximate attribute test in lines 2-4. The code for an approximate version of this function, using the $8-bit$ AxDC1 is shown in Source code 4, where `axdc1` is a C function that implements the same AxDC1 logic behavior described

Listing 2 – C5.0 continuous node evaluation

```c
void FindLeaf(DataRec Case, Tree T, Tree PT, float Fraction) {
    switch ( T->NodeType )
    {
    case 0:  /* leaf */
        LeafUpdate:
    ...
    case BrThresh:  /* test of continuous attribute */
        ...
        } else {
            /*  Find weights for <= and > branches, interpolating if
                  probabilistic thresholds are used  */
            BrWt[2] = Interpolate(T, CVal(Case, T->Tested));
            BrWt[3] = 1 - BrWt[2];

            ForEach(v, 2, 3) {
                if ( (NewFrac = Fraction * BrWt[v]) >= 0.01 )
                    FindLeaf(Case, T->Branch[v], T, NewFrac);
            }
        ...
        return;
```

Source: Adapted from Ross Quinlan, licensed through GPL via Rulequest Research.

Listing 3 – C5.0 continuous attribute test

```c
float Interpolate(Tree T, ContValue Val) {
    return ( Val <= T->Lower ? 1.0 : Val >= T->Upper ? 0.0 :
        Val <= T->Mid ? 1 - 0.5 * (Val - T->Lower) / (T->Mid - T->Lower + 1E-6) :
        0.5 - 0.5 * (Val - T->Mid) / (T->Upper - T->Mid + 1E-6) );
}
```

Source: Ross Quinlan, licensed through GPL via Rulequest Research.

in Eq. (9). Here, we unraveled the multiple ternary operators used in the original source code to improve code readability and replaced the comparisons with our approximation in lines 2, 4, and 6.

The procedure described for the approximation of the C5.0 classifiers was repeated for all 5 comparators proposed in this work, allowing us to compare the accuracy of the different designs in the DT models.

Energy consumption estimation

Along with the accuracy, two other aspects of the classifier are of key importance: the energy consumption and the number of comparison operations. The energy consump-

Listing 4 – Approximate C5.0 continuous node evaluation

```
float Interpolate(Tree T, ContValue Val) {
    if ( axdc1(Val, T->Lower, 8) )
        return 1.0;
    else if ( axdc1(T->Upper, Val, 8) )
        return 0.0;
    else if ( axdc1(Val, T->Mid, 8) )
        return 1 - 0.5 * (Val - T->Lower)
            / (T->Mid - T->Lower + 1E-6);
    else
        return 0.5 - 0.5 * (Val - T->Mid)
            / (T->Upper - T->Mid + 1E-6);
}
```

Source: Pedro Silva, 2022.

tion is the total amount of energy consumed during the execution of the classifier, which is important to determine the energy efficiency of the classifier. The number of comparison operations, on the other hand, serves as a diagnostic tool to determine how the approximations impacted the depth of the traversal in the tree. It is evident that the larger the number of operations, the larger the energy consumption. Thus, in addition to inserting the AxC comparators in the C5.0 classification state, we also implemented a log to record the number of operations performed by the classifier as well as the values used in each comparison.

Since our analysis is based on electrical simulations of the comparators, we have no way of accurately calculating the energy consumption of the entire classifier. Instead, we focus on the gains provided by the approximations in the comparators and compare them to the energy consumption of the comparators themselves. This way, we estimate the energy savings that can be achieved by using the approximations in the DT. The energy estimation constitutes the step in the workflow (step 3, in Fig. 14).

This estimate was done through the electrical simulation of each comparison recorded in the log in the previous step with their respective approximated comparators. The energy consumption of the comparators on the entire test dataset is then calculated by adding the energy measured for each comparison simulated. Note that this estimate was only done for quantized cases since we had no SPICE description of a full-precision floating-point comparator. All the operations were simulated with HSPICE®, with a duration of 2.5 ns under the nominal voltage of 0.7 V, measuring the energy consumption for each comparator circuit evaluated according to Eq. (5). We observe that to obtain a fully accurate energy estimation, it would be necessary to perform the synthesis of these circuits and the overall application, which is currently beyond the scope of this project. The SPICE simulation results obtained were a means to provide a concise analysis of the gains in each approximation compared to the baseline version.

Summary and evaluation

Finally, the last stage of the workflow (step 4, in Fig. 14) consisted in summarizing and understanding all produced metrics, to provide clearer insight into the characteristics and behavior of each approximation in a given dataset. We analyze the accuracy, total energy consumption for all comparison operations executed, and the number of operations needed. From these, another metric is calculated, measuring the average power consumption per inference case. This metric is calculated by dividing the total energy consumption by the duration of each simulation, and further dividing the result by the number of test cases. This way, we can better compare the energy efficiency of the different approximations of DTs.

The exact and approximate versions of C5.0 were assessed on 5 continuous and mixed-attribute datasets provided by the University of California Irvine (UCI) Machine Learning Repository (DUA; GRAFF, 2017), namely the *Heart disease*, *Arrhythmia*, *Adult*, *Iris* and *Forest fires* datasets. The results were then analyzed using the unaltered C5.0 version as a benchmark, focusing on the accuracy of the model on both training and test data, which corresponded respectively to 70% and 30% of each complete example set.

5.2 ANALYSIS OF PRE-PROCESSING TECHNIQUES: QUANTIZATION AND SCALING OF DATASETS

There are many alternatives to pre-process data according to the properties of the dataset and ML model. We propose one generic flow independent of the features of the dataset being classified, as a means to focus our analysis on the isolated impact of approximation, minimizing the effect of pre-processing on the accuracy. To find the best approach to this end, we observed two techniques, referred to as *pure quantization* and *scaling*. The impact of these techniques on the accuracy was analyzed when adopted in the exact version and together with the approximate comparators.

The *pure quantization* quantizes the dataset to 8 bits in fixed-point notation, without any scaling. Firstly, a correction was applied to remove the negative values, not applicable to the dedicated comparators. Then, we use an exhaustive search optimization strategy to find the best size for the fractional part of the bit array. Note that this is only possible due to the low number of bits in our target datasets.

The *scaling* maps the values in the dataset to the range $[0, 255]$ and then quantizes them to 8 bits. This technique was used to avoid the loss of information that occurs when using pure quantization. Thus, it is a more robust quantization method than the pure quantization approach.

In essence, the difference between both strategies is that in the case of pure quantization, if a value is greater than the maximum value in the dataset, it will be quantized to the maximum value, and if it is smaller than the minimum value, it will be quantized to the minimum value. On the other hand, in the case of scaling, the values are mapped to the range $[0, 255]$ and then quantized, which avoids the loss of information.

A comparison of the accuracy results on the test accuracy when using both approaches is presented in Table 6. Baseline results were generated by training and testing with the original C5.0 on three versions of the data set: raw, quantized, and scaled.

Together with the exact dedicated comparator, the approximate versions were trained with the default C5.0 version on the pre-processed datasets. Then, they were evaluated using the approximate comparators in each decision node. This way, we can determine how the approximation affects the accuracy of the classifier for all considered designs.

With the scaling technique, both the baseline and EDC represent pre-processing-only approximations, considering that these versions have the same accuracy on all datasets while showing an average deviation of only 0.01% to the classification with the raw dataset. It is then safe to assume this difference is solely caused by the quantization error. Still, it is curious that on the quantized data, the EDC performed considerably worse than the baseline.

From this table, we see that the scaling approach provides overall better results than the pure quantization approach, best seen in the Iris and Forest datasets. This difference is because pure quantization loses more information when quantizing the data. Scaling, on the other hand, allows the comparators to have a better resolution for the values of the dataset. Thus, scaling was used in the rest of the experiments.

5.3 IMPACT OF APPROXIMATION IN THE C5.0 CLASSIFICATION MODEL

This section will discuss the results obtained by applying the experimental workflow described to five different continuous and mixed-attribute datasets. Some of these results were presented in Pedro Aquino Silva, Grellert, and Meinhardt (2022), although the following discussion considers the most up-to-date data and more in-depth analysis. We first present the results for the accuracy of the classifiers on the testing data, followed

Table 6 – Comparison of quantization-only and scaling pre-processing on the accuracy over test examples

TEST ACCURACY (%)		**Iris**	**Forest**	**Adult**	**Heart Disease**	**Arrhythmia**
RAW	**baseline**	**93.3**	**96.4**	**86.6**	**63.3**	**64.4**
	default C5.0	93.3	96.4	86.0	61.7	65.6
	EDC	83.3	87.5	86.0	61.7	60.0
	AxDC2	56.7	85.7	83.3	56.7	31.1
QUANTIZED	AxDC6	30.0	39.3	23.5	51.7	5.6
	SMA	36.7	60.7	76.4	58.3	11.1
	AMA1	36.7	60.7	76.4	58.3	11.1
	AMA2	30.0	39.3	23.6	53.3	6.7
	default C5.0	93.3	98.2	86.4	63.3	63.3
	EDC	93.3	98.2	86.4	63.3	63.3
	AxDC2	93.3	98.2	85.9	63.3	63.3
SCALED	AxDC6	30.0	39.3	23.5	40.0	7.8
	SMA	73.3	78.6	77.6	55.0	16.7
	AMA1	73.3	78.6	77.6	55.0	16.7
	AMA2	33.3	60.7	24.2	46.7	6.7

Source: Pedro Silva, 2022.

by the energy consumption and the number of operations. Tables 7 and 8 summarize the results obtained for each dataset test case.

Table 7 – Evaluation of the Comparison Circuits on Decision Trees Classification Model – Mixed Attribute Datasets

	Heart disease Test samples = 60 Attributes = 13 (6 continuous)			**Arrhythmia** Test samples = 90 Attributes = 279 (206 continuous)			**Adult** Test samples = 16281 Attributes = 15 (6 continuous)		
Comparator circuit	Acc (%)	Energy (fJ)	# Ops	Acc (%)	Energy (fJ)	# Ops	Acc (%)	Energy (pJ)	# Ops
EDC	63.3	220.89	178	63.3	1,580.66	1198	86.4	132.01	118,087
AxDC1	63.3	190.89	178	63.3	1,391.39	1198	85.9	94.15	119,006
AxDC2	40.0	101.97	262	7.8	698.77	1795	23.5	118.45	331,039
SMA	55.0	767.90	319	16.7	3,691.76	1666	77.6	346.30	227,379
AMA1	55.0	225.90	141	16.7	530.36	468	77.6	73.84	109,292
AMA2	46.7	349.78	271	6.7	3,443.67	2367	24.2	456.99	342,376

Source: Pedro Silva, 2022.

In terms of prediction performance, it is observed that the adoption of approximated comparators on the classification of continuous datasets has a variable effect on the accuracy in contrast with the exact version. All approximated comparators were analyzed against the EDC, to account for both accuracy and energy performance, since we currently have no estimates for the energy consumption of full-precision comparison, as used in software-only classification.

Table 8 – Evaluation of the Comparison Circuits on Decision Trees Classification Model – Continuous Attribute Datasets

	Iris Test samples = 30 Attributes = 5			**Forest fires** Test samples = 56 Attributes = 13		
Comparator circuit	Acc (%)	Energy (fJ)	# Ops	Acc (%)	Energy (fJ)	# Ops
EDC	93.3	122.91	91	98.2	89.61	79
AxDC1	93.3	105.92	91	98.2	84.46	79
AxDC2	30.0	65.99	141	39.3	37.24	112
SMA	73.3	454.14	167	78.6	402.48	145
AMA1	73.3	118.56	76	78.6	129.28	66
AMA2	33.3	229.62	120	60.7	86.96	56

Source: Pedro Silva, 2022.

Out of the Approximate Dedicated Comparators, the AxDC1 showed the closest accuracy to the EDC. This result corroborates our expectations, since the AxDC1 is the simplest approximation, and thus, the one that is expected to have the least impact on the accuracy of the classifier. This was also validated through the error analysis presented in Section 4.1, where we observed that the AxDC1 introduced only 0.59% error in the comparison truth table. In terms of energy reduction, the estimates were also consistent with the characterization analysis, where we observed that the AxDC1 had a modest reduction of 9.10% in power dissipation. In classification, the AxDC1 has a larger gain on energy consumption, of around 15% on average.

Interestingly, the only case where the AxDC1 had a different accuracy and number of operations than the exact version was the Adult dataset, as shown in Table 7. However, it may appear counter-intuitive that this was still the largest reduction of energy consumption obtained by this design (28.7%), despite the increase in the number of operations. This is explained by the fact that, in this dataset, the approximation caused the tree traversal to differ, resulting in a change of the values evaluated, and since the power consumption is related to the values being compared, the energy consumption was also affected. We also highlight that this case represents the deepest tree out of the evaluated datasets, which increases the probability of the traversal diverging from the path followed in the classification with exact tests.

The AxDC2 reaches the highest energy reduction among all circuits. The largest impact on energy is observed on the Forest fires dataset (58.45%), in Table 8, even though the number of operations is almost 1.5$\times$ higher than the classification with the EDC. Once again, this is in line with the characterization analysis, where we observed that the AxDC2 had a reduction of 50.3% in power dissipation. In classification, this circuit has a slightly lower gain in energy consumption, of around 45% on average.

However, as the approximation with the highest ER, the AxDC2 also showed the worst performance in terms of accuracy, with an average reduction of accuracy of about 65%. Recalling the approximation technique and the error heatmaps discussed in Section 4.1, the results indicate that the only use case that would justify the use of the AxDC2 is for features that present values concentrated on the furthest extremes of the data range, resulting in the traversal path being less frequently affected by the approximation. Currently, we have little information about the distribution of the values in the features, and thus, we cannot make any more claims concerning the best use case for the AxDC2.

Focusing on the FA-based comparators, it was seen in Section 4.1 that the SMA and AMA1 result in the same approximation, causing the accuracy of both classifiers to be equal. These designs showed the second-best results in terms of accuracy, with an average reduction of 28% accuracy, despite only the AMA1 presenting a net positive in energy reduction.

While the AMA2 has the most promising results in the electrical evaluation, the energy estimate was significantly impacted by the steep increase in the number of operations for this comparator, which resulted in a net negative in both energy reduction and accuracy. In the worst case, the number of comparisons achieved nearly $3\times$ that of the exact version, whereas the energy consumption increased by a factor of 3.5.

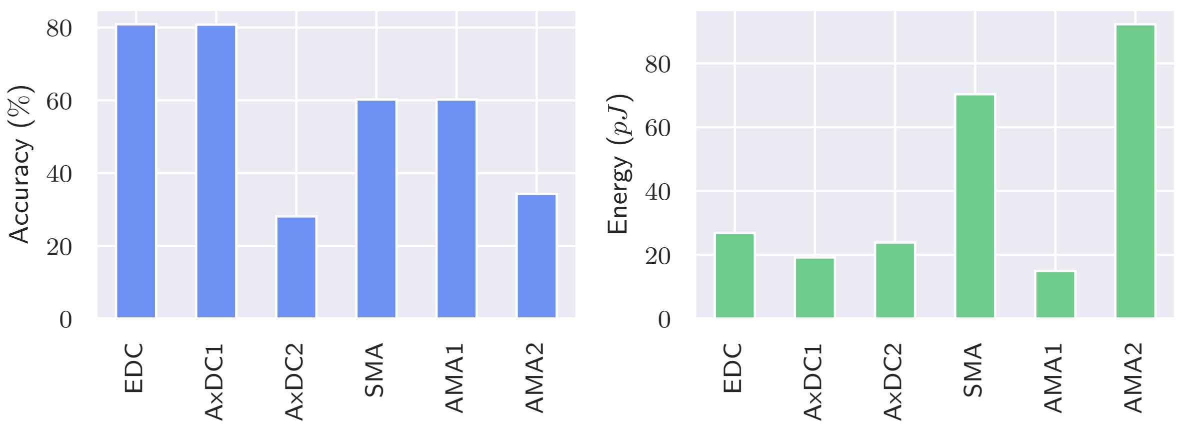
Fortunately, the FA-based circuits allow for a straightforward optimization by limiting the number of FAs approximated. This also enlarges the space to include strategies for automatically searching for the best approximation range, which is currently beyond the scope of this project. However, the energy results in the error and electric characterization indicate that the AMA1 and AMA2 are promising candidates for future work on configurable approximation.

5.4 DISCUSSION

Considering the main concern of this analysis is evaluating the energy-accuracy trade-off, Fig. 15 shows the average energy estimation and accuracy over all datasets, as a means of condensing the information previously discussed. Here, we observe that the AxDC1 is the most promising candidate as an EDC replacement, as it is very conservative in the classification accuracy while providing good energy reduction. Nonetheless, the AMA1-based comparator also appears as an apt candidate, especially considering it has the lowest average energy consumption and, being an FA-based adder, has a large flexibility in terms of the approximation range. On the other hand, the extent of approximation in the AxDC2 led to the worst accuracy in the classification tests, despite its attractive results in energy consumption. In fact, due to the diverging paths and increased number of operations caused by using the AxDC2, the AxDC1 provides better results even for energy consumption.

In the graphs presented in Fig. 15, it is vital to keep in mind that the energy estimation depends on the number of operations performed. For example, in the Adult dataset, the number of comparisons in each case is of the order of 10^5, as shown in Table 7. Due to these large values, the average case is highly influenced by the performance of the comparator in this dataset. Even so, this measure is still representative when confronting all comparators, since they all have many comparisons of a similar order of magnitude for most datasets. Furthermore, evaluating a case with large volumes of data is highly desirable, as it is more representative of some real-world applications than the smaller research-oriented test cases.

Figure 15 – Average accuracy and energy consumption for each evaluated comparator



Source: Pedro Silva, 2022.

Regarding the poor results in accuracy for most comparators, we emphasize that our intervention in the `Interpolate` function was only enough to consider the approximation of already constructed trees. Recalling Section 2.2, this approximation could be improved upon by adapting the construction algorithm itself to use the approximate attribute test when calculating the threshold for each numerical feature. In our efforts to implement an approximation-aware training algorithm, this alteration proved the need for a more thorough understanding of the C5.0 algorithm and its implementation, which was not possible in the time available for this project, leaving this implementation as future work. Instead, we decided to focus on the approximation of the constructed trees, which is still a valid approach for the use of the approximated comparators.

A known limitation of our current approach is the fact that the comparators are constrained to fixed bit widths. That is, the precision of the comparators cannot be changed during the workflow. This problem could be overcome by using a configurable comparator, such as the FA-based circuits, as well as developing a tool to automatically create SPICE descriptions for $n-bit$ versions of the dedicated comparators, following the equations presented in Chapter 4. By removing this restriction, more advanced power-accuracy optimization strategies could be implemented, including the automatic search for the best approximation range for FA-based adders, and the use of the precision-scaling technique to identify the best bit width for each feature in a dataset with any approximation configuration.

Finally, while this analysis focused on the C5.0 classifier, the automated workflow is generic enough to be adapted with minimal intervention to any other classifier that employs comparator-based testing. This includes the CART implementations in the `scikit-learn` framework and the `rpart` package. The `scikit-learn` version, in particular, is a very interesting candidate for future work, as it is a popular and widely used framework for machine learning in Python, and it is also open-source, allowing for an in-depth investigation of the implementation. As previously stated, since its implementation does not currently support categorical data, a commonly used technique is the encoding of the categorical features as numerical features. Exploring this encoding could enable the usage of the approximated comparators developed here for categorical as well as continuous attributes. Thus, the evaluation of other classifiers is also a promising avenue for future work, as it would allow for a more thorough analysis of the impact of the approximation on the classification accuracy, as well as the possibility of comparing the results with other approximation techniques.

6 CONCLUSIONS

This work explored two approaches for approximation in comparator circuit design, employing FA-based and dedicated architectures, and investigated the impact of their usage in tree-based classification models. In this sense, the efforts detailed here led to an approximation of already-constructed Decision Trees, where the numerical attribute test was changed from an exact comparison with a threshold value to five different inexact versions.

While the related work regarding the use of AxC in DT applications focused mostly on approximating the threshold used for attribute testing or precision scaling, the approach developed in this study consisted of approximating the circuits themselves while maintaining the threshold values computed in the training stage, and using scaling and quantization only as pre-processing techniques. As noted, the development of a method for the approximation-aware construction of DTs is ongoing work.

The two approaches evaluated represent the most commonly used techniques for the design of comparator circuits. The dedicated versions are capable of providing significant improvements in power consumption, number of transistors, and delay characteristics, while the FA-based designs are highly configurable and scalable. Here, approximation was inserted using three main techniques: 25% input truncation, 50% bit truncation and copy strategy, and FA replacement, representing the AxDC1, AxDC2 and FA-based comparators respectively. For an electric evaluation, the circuits were contrasted with a single-output Exact Dedicated Comparator (EDC), and 7 nm FinFET netlists for SPICE simulations were produced for $8-bit$ extensions of all designs.

The AxDC1 had the lowest level of approximation considered, incurring in an Error Rate of only 0.59% in the $8-bit$ version, which is further reduced up to 0.1% in $n-bit$ extensions. On the other hand, the most intensive approximation was done on the AxDC2 circuit, which had an ER of $\sim$ 37%. The FA-based comparators had a variable performance on error metrics, though the impact is most noticeable on the AMA2. It is important to highlight that the error seen in FA-based circuits is more easily controlled, since the approximation may be restricted to any fraction of the LSBs – here the extreme case of 100% approximation was considered.

Following the analysis of the error behavior of the $8-bit$ AxC comparators implemented, the circuits were evaluated in a DT classification model. To do so, I used the C5.0 classification algorithm as a case study, which employs comparisons in the decision nodes testing numerical features. In essence, what was tested is the performance of each architecture as a *replacement* of the EDC comparator. Here, the results were especially promising for the AxDC1 design, since it had a minimal impact on the accuracy of the trained trees while providing consistent energy reduction compared to the EDC. The AMA1 also comes across as a good alternative, mainly considering the good results in energy consumption and the possibility of tuning the level of approximation to reduce the impact of errors in the traversal of the tree.

The AxDC2 comparator had a low performance in most tests, due to its large ER causing the tree traversal to follow a deeper path. This impacted both the accuracy of the model and the energy consumption of each inference, as it increased the number of attribute tests performed. It became clear that this circuit is not suitable to be a replacement for the exact comparator in a hardware implementation, and verifying if it can be used for any specific features in a dataset depending on their value distribution is left as future work.

The results here presented help to clarify the extent of approximation acceptable in single-tree DT models. Nonetheless, several possible improvements and avenues for future work were identified during the development. These ideas include, for instance, the implementation of a mechanism to explore configurability in FA-based comparators, exploiting the fact that these circuits allow for finer tuning of approximation parameters. Through this method, one could find the exact number of FAs that should be replaced with approximate versions and which cells are the most critical to be maintained as exact versions to improve the power-accuracy trade-offs. Another possible branch for future research is the inclusion of an optimization algorithm to implement precision scaling, tuning the bit width for any given approximate comparator and dataset.

Recalling the related work discussed in the literature review, several projects explore software-level approximation in tree-based models, opening yet another path of investigation on the possibility of combining different levels of approximation to further reduce the energy consumption of these models in a hardware implementation context.

Considering the cross-level nature of the work described in this text, I also identified a few improvements on the circuits and systems perspective that should be given thought in any further efforts. For example, the comparator circuits were described using mostly a complementary MOS logic style, with only the XNOR gates using low-power design techniques, namely PTL. Here, I highlight the opportunity for the inclusion of more state-of-the-art low-power logic styles, i.e. dynamic CMOS, further use of PTL, transistor pruning, etc.

In conclusion, the efforts constituting this work serve as first steps in a thorough investigation of the role of comparators as decision-making circuits, and ways to improve their energy efficiency in a world that increasingly requires solutions to process and analyze huge amounts of data. The tools here developed will aid in the future in-depth explorations of the diverse research paths encountered when analyzing our results, and in advancing our knowledge of approximation techniques and their impact on tree-based classification systems.

6.1 PUBLICATIONS

The development of this work, starting in early 2021, has been documented and published in events of regional and international impact. Early results, when we were starting to get accustomed to the C5.0 source code, and developed the first iterations of comparators presented here, were published in the *36º Simpósio Sul de Microeletrônica*, held in 2021, evaluating six different approximations for dedicated comparators in an early version of the C5.0 workflow (SILVA, P.; GRELLERT; MEINHARDT, 2021). At this point, we believed C5.0 to internally map the categorical features into numerical values, thus using comparators for all features in mixed-attribute and categorical-only datasets. On subsequent investigation, we noted that this was not the case, and adapted the workflow accordingly. From these results, we also refined the approximations, evolving them to the two dedicated circuits presented in this work.

After more extensive tests and improvements in our understanding of Decision Tree Classifiers and their implementation, as well as the design of comparator circuits, the new efforts of approximating the attribute test of continuous features only resulted in the presentation in the same symposium the following year (SILVA, P.; GRELLERT; MEINHARDT, 2022b). At this point, our evaluation workflow evolved to a form that justified their presentation in an international forum, leading to the publication of a complete paper in the 2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC) (SILVA, P. A.; GRELLERT; MEINHARDT, 2022). Since our efforts had already begun indicating the need to include the approximation in the learning algorithm to improve the accuracy of the approximate classifiers, an extended abstract was presented at the same conference, detailing the objective of achieving a complete approximation-aware workflow for Decision Tree applications (SILVA, P.; GRELLERT; MEINHARDT, 2022a).

Finally, the most recent results are being prepared for submission to an international journal, yet to be defined. This paper will consider the effects of the approximation-aware training on the accuracy and energy efficiency of the comparators presented here and will be submitted after obtaining such experiment results.

Below, we present the complete list of publications from this project to date:

1. Exploring $4 - bit$ Approximated Comparators on a Decision Tree Classification Model, 36^{o} Simpósio Sul de Microeletrônica, 2021 (SILVA, P.; GRELLERT; MEINHARDT, 2021),
2. Energy-Efficient Approximate Comparators in Decision Tree Applications, 37^{o} Simpósio Sul de Microeletrônica, 2022 (SILVA, P.; GRELLERT; MEINHARDT, 2022b),
3. Exploring Approximate Comparator Circuits on Power Efficient Design of Decision Trees, IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC), 2022 (SILVA, P. A.; GRELLERT; MEINHARDT, 2022),
4. Approximation Workflow for Energy-Efficient Comparators in Decision Tree Applications, IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC), **Student Forum**, 2022 (SILVA, P.; GRELLERT; MEINHARDT, 2022a).

The source code for the altered C5.0 implementation and the comparator circuits described in this work are publicly available at the following link: `https://github.com/phaquinosilva/axc-dt`. Results and figures presented in this work are also available in the same repository.

REFERENCES

ABREU, Brunno A.; GRELLERT, Mateus; BAMPI, Sergio. VLSI Design of Tree-Based Inference for Low-Power Learning Applications. In: 2020 IEEE International Symposium on Circuits and Systems (ISCAS). [S.l.: s.n.], 2020. P. 1–5.

BALASKAS, Konstantinos; ZERVAKIS, Georgios; SIOZIOS, Kostas; TAHOORI, Mehdi B.; HENKEL, Jorg. Approximate Decision Trees For Machine Learning Classification on Tiny Printed Circuits. In: 2022 23rd International Symposium on Quality Electronic Design (ISQED). [S.l.]: IEEE, Apr. 2022. P. 1–6.

BARBARESCHI, Mario; BARONE, Salvatore; MAZZOCCA, Nicola. Advancing synthesis of decision tree-based multiple classifier systems: an approximate computing case study. **Knowledge and Information Systems**, v. 63, p. 1577–1596, 6 June 2021. ISSN 02193116.

BARUA, Hrishav Bakul; CHANDRA MONDAL, Kartick. Green Data Mining using Approximate Computing: An experimental analysis with Rule Mining. In: 2018 International Conference on Computing, Power and Communication Technologies (GUCON). [S.l.: s.n.], 2018. P. 115–120.

BARUA, Hrishav Bakul; MONDAL, Kartick Chandra. Approximate Computing: A Survey of Recent Trends—Bringing Greenness to Computing and Communication. **Journal of The Institution of Engineers (India): Series B**, v. 100, p. 619–626, 6 Dec. 2019. ISSN 22502114.

BREIMAN, Leo; FRIEDMAN, Jerome; STONE, Charles J.; OLSHEN, R. A. **Classification and Regression Trees**. Andover, England, UK: Taylor & Francis, 1984. ISBN 978-0-41204841-8.

CLARK, Lawrence T.; VASHISHTHA, Vinay; SHIFREN, Lucian; GUJJA, Aditya; SINHA, Saurabh; CLINE, Brian; RAMAMURTHY, Chandarasekaran; YERIC, Greg. ASAP7: A 7-nm finFET predictive process design kit. English (US). **Microelectronics**, Elsevier Limited, v. 53, p. 105–115, July 2016. Publisher Copyright: © 2016 The Authors. ISSN 0026-2692.

DUA, Dheeru; GRAFF, Casey. **UCI Machine Learning Repository**. [S.l.: s.n.], 2017. Available from: `http://archive.ics.uci.edu/ml`.

EFSTATHIOU, Constantinos; AGALIOTI, Laura; TSIATOUHAS, Yiorgos. Efficient Dynamic Logic Magnitude Comparators. In: 2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC). [S.l.: s.n.], 2022. P. 1–5.

GARCÍA-MARTÍN, Eva; LAVESSON, Niklas; GRAHN, Håkan; CASALICCHIO, Emiliano; BOEVA, Veselka. Energy-aware very fast decision tree. **International Journal of Data Science and Analytics**, v. 11, p. 105–126, 2 Mar. 2021. ISSN 23644168.

GOEL, Abhinav; TUNG, Caleb; LU, Yung Hsiang; THIRUVATHUKAL, George K. A Survey of Methods for Low-Power Deep Learning and Computer Vision. In: IEEE World Forum on Internet of Things, WF-IoT 2020 - Symposium Proceedings. [S.l.: s.n.], Mar. 2020. arXiv: `2003.11066`.

GUPTA, Akash; KHATRI, Manohar; RAJPUT, Sachin Kumar; MEHRA, Anu; BATHLA, Shikha. Design of low power magnitude comparator. In: 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence. [S.l.]: IEEE, Jan. 2017. P. 754–758.

GUPTA, Vaibhav; MOHAPATRA, Debabrata; PARK, Sang Phill; RAGHUNATHAN, Anand; ROY, Kaushik. IMPACT: IMPrecise adders for low-power approximate computing. In: IEEE/ACM International Symposium on Low Power Electronics and Design. [S.l.: s.n.], 2011. P. 409–414.

HAN, Jie. Introduction to approximate computing. In: 2016 IEEE 34th VLSI Test Symposium (VTS). [S.l.: s.n.], 2016. P. 1–1.

HOLDSWORTH, B.; WOODS, R.C. 5 - Combinational logic design with MSI circuits. In: HOLDSWORTH, B.; WOODS, R.C. (Eds.). **Digital Logic Design (Fourth Edition)**. Fourth Edition. Oxford: Newnes, 2002. P. 105–141. ISBN 978-0-7506-4582-9.

AL-JARRAH, Omar Y.; YOO, Paul D.; MUHAIDAT, Sami; KARAGIANNIDIS, George K.; TAHA, Kamal. Efficient Machine Learning for Big Data: A Review. **Big Data Research**, v. 2, p. 87–93, 3 Sept. 2015. ISSN 22145796.

KIM, Yongtae; ZHANG, Yong; LI, Peng. Energy Efficient Approximate Arithmetic for Error Resilient Neuromorphic Computing. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 23, n. 11, p. 2733–2737, 2015.

KNUTH, Donald E. Two Notes on Notation. **The American Mathematical Monthly**, Mathematical Association of America, v. 99, n. 5, p. 403–422, 1992. ISSN 00029890, 19300972.

KUMAR, Ashish; GOYAL, Saurabh; VARMA, Manik. Resource-Efficient Machine Learning in 2 KB RAM for the Internet of Things. In: PROCEEDINGS of the 34th International Conference on Machine Learning - Volume 70. Sydney, NSW, Australia: JMLR.org, 2017. (ICML'17), p. 1935–1944.

LI, Qingzheng; BERMAK, Amine. A low-power hardware-friendly binary decision tree classifier for gas identification. **Journal of Low Power Electronics and Applications**, v. 1, p. 45–58, 1 Mar. 2011. ISSN 20799268.

MARWAHA, Damini; SHARMA, Anurag. A review on approximate computing and some of the associated techniques for energy reduction in IOT. In: 2018 2nd International Conference on Inventive Systems and Control (ICISC). [S.l.: s.n.], 2018. P. 319–323.

MOREAU, Thierry; SAMPSON, Adrian; CEZE, Luis. Approximate Computing: Making mobile systems more efficient. **IEEE Pervasive Computing**, v. 14, n. 2, p. 9–13, Apr. 2015. ISSN 15361268.

MUNRATIWAR, Shubham; SAIKRISHNA, Y; JYOTHI, V. Design of High Speed $8-bit$ Magnitude Comparator for Security Application. In: 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE). [S.l.]: IEEE, Apr. 2022. P. 1–6.

OSTA, M.; IBRAHIM, A.; CHIBLE, H.; VALLE, M. Approximate Multipliers Based on Inexact Adders for Energy Efficient Data Processing. In: 2017 New Generation of CAS (NGCAS). [S.l.: s.n.], Sept. 2017. P. 125–128.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

QUINLAN, J. R. Induction of Decision Trees. **Mach. Learn.**, Kluwer Academic Publishers, USA, v. 1, n. 1, p. 81–106, Mar. 1986. ISSN 0885-6125.

QUINLAN, J. Ross. **C4.5 - Programs for Machine Learning**. [S.l.]: Morgan Kaufmann Publishers Inc., 1993. ISBN 1558602380.

REAGEN, Brandon; WHATMOUGH, Paul; ADOLF, Robert; RAMA, Saketh; LEE, Hyunkwang; LEE, Sae Kyu; HERNÁNDEZ-LOBATO, José Miguel; WEI, Gu-Yeon; BROOKS, David. Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators. In: 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). [S.l.: s.n.], 2016. P. 267–278.

RUSSELL, Stuart J.; NORVIG, Peter. **Artificial Intelligence: a modern approach**. 3. ed. [S.l.]: Pearson, 2009.

SEKANINA, Lukas; VASICEK, Zdenek; MRAZEK, Vojtech. Inexact Arithmetic Operators. In: **Approximate Computing Techniques: From Component- to Application-Level**. Ed. by Alberto Bosio, Daniel Ménard and Olivier Sentieys. Cham: Springer International Publishing, 2022. P. 81–107. ISBN 978-3-030-94705-7.

SHANNON, Claude E.; WEAVER, Warren. **The Mathematical Theory of Communication**. Urbana, IL: University of Illinois Press, 1949. ISBN 978-0-252-72548-7.

SHARMA, Geetanjali; NIRMAL, Umar; MISRA, Yogesh. A Low Power $8-bit$ Magnitude Comparator with Small Transistor Count using Hybrid PTL/CMOS Logic. **International Journal of Computational Engineering and Management**, v. 12, p. 110–115, 2011.

SILVA, Pedro; GRELLERT, Mateus; MEINHARDT, Cristina. Approximation Workflow for Energy-Efficient Comparators in Decision Tree Applications. In: 2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC). [S.l.: s.n.], 2022. P. 1–2.

SILVA, Pedro; GRELLERT, Mateus; MEINHARDT, Cristina. Energy-Efficient Approximate Comparators in Decision Tree Applications. In: 37^o Simpósio Sul de Microeletrônica. [S.l.: s.n.], 2022. P. 1–4.

SILVA, Pedro; GRELLERT, Mateus; MEINHARDT, Cristina. Exploring $4-bit$ Approximated Comparators on a Decision Tree Classification Model. In: 36^o Simpósio Sul de Microeletrônica. [S.l.: s.n.], 2021. P. 1–4.

SILVA, Pedro Aquino; GRELLERT, Mateus; MEINHARDT, Cristina. Exploring Approximate Comparator Circuits on Power Efficient Design of Decision Trees. In: 2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC). [S.l.: s.n.], 2022. P. 1–6.

SILVA, Pedro Aquino; MEINHARDT, Cristina. Energy-Efficient Design of Approximated Full Adders. In: 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS). [S.l.: s.n.], 2020. P. 1–4.

SINGH, Pranay; JAIN, Pramod Kumar. Design and Analysis of Low Power, High Speed 4 - Bit Magnitude Comparator. In: 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE). [S.l.: s.n.], 2018. P. 1680–1683.

SINGH, Shivam. Green computing strategies & challenges. In: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT). [S.l.: s.n.], 2015. P. 758–760.

STANLEY-MARBELL, Phillip et al. Exploiting Errors for Efficiency: A Survey from Circuits to Applications. **ACM Computing Surveys**, v. 53, 3 July 2020. ISSN 15577341.

STROLLO, Antonio G. M.; ESPOSITO, Darjn. Approximate computing in the nanoscale era. In: 2018 International Conference on IC Design & Technology (ICICDT). [S.l.: s.n.], 2018. P. 21–24.

STRUHARIK, J.R. Implementing decision trees in hardware. In: 2011 IEEE 9th International Symposium on Intelligent Systems and Informatics. [S.l.: s.n.], 2011. P. 41–46.

STRUHARIK, Rastislav J. R.; NOVAK, Ladislav A. Hardware Implementation of Decision Tree Ensembles. **Journal of Circuits, Systems and Computers**, v. 22, p. 1350032, 05 June 2013. ISSN 0218-1266.

TORRES-ALVARADO, Alan; MORALES-ROSALES, Luis Alberto; ALGREDO-BADILLO, Ignacio; LÓPEZ-HUERTA, Francisco; LOBATO-BAEZ, Mariana; LÓPEZ-PIMENTEL, Juan Carlos. Trade-Off Analysis of Hardware Architectures for Channel-Quality Classification Models. **Sensors**, v. 22, p. 2497, 7 Mar. 2022. ISSN 1424-8220.

TRIPATHY, Suryasnata; MANDAL, Sushanta K.; PATRO, B. Shivalal; OMPRAKASH, L. B. Low Power, High Speed $8-bit$ Magnitude Comparator in 45nm Technology for Signal Processing Application. **Indian Journal of Science and Technology**, v. 8, p. 1–10, 1 Jan. 2015. ISSN 09746846.

VALLABHUNI, Rajeev Ratna; SRAVYA, D.V.L.; SHALINI, M. Sree; MAHESHWARARAO, G.Uma. Design of Comparator using 18nm FinFET Technology for Analog to Digital Converters. In: 2020 7th International Conference on Smart Structures and Systems (ICSSS). [S.l.]: IEEE, July 2020. P. 1–6.

WESTE, Neil; HARRIS, David. **CMOS VLSI Design: A Circuits and Systems Perspective**. 4th. USA: Addison-Wesley Publishing Company, 2010. ISBN 0321547748.

XU, Xiaoqing; SHAH, Nishi; EVANS, Andrew; SINHA, Saurabh; CLINE, Brian; YERIC, Greg. Standard cell library design and optimization methodology for ASAP7 PDK: (Invited paper). In: 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). [S.l.: s.n.], 2017. P. 999–1004.

Y. KAMAL, Kamal. The Silicon Age: Trends in Semiconductor Devices Industry. **Journal of Engineering Science and Technology Review**, v. 15, p. 110–115, May 2022.

YOUNIS, Mohammed; ZAMLI, Kamal. A Strategy for Automatic Quality Signing and Verification Processes for Hardware and Software Testing. **Advances in Software Engineering**, v. 2010, Jan. 2010.

YU, Bin et al. FinFET scaling to 10 nm gate length. In: DIGEST. International Electron Devices Meeting, [s.l.: s.n.], 2002. P. 251–254.

ZHANG, Boyu; DAVOODI, Azadeh; HU, Yu Hen. Exploring Energy and Accuracy Tradeoff in Structure Simplification of Trained Deep Neural Networks. **IEEE Journal on Emerging and Selected Topics in Circuits and Systems**, v. 8, n. 4, p. 836–848, 2018.

ZHOU, Yangcan; LIN, Jun; WANG, Jichen; WANG, Zhongfeng. Approximate Comparator: Design and Analysis. In: 2018 IEEE International Workshop on Signal Processing Systems (SiPS). [S.l.: s.n.], 2018. P. 1–5.

Exploring Approximate Comparator Circuits on Power Efficient Design of Decision Trees

Pedro Aquino Silva, Mateus Grellert, Cristina Meinhardt
Departamento de Informática e Estatística - PPGCC
Universidade Federal de Santa Catarina (UFSC), Brazil
pedro.aquino@grad.ufsc.br, {mateus.grellert, cristina.meinhardt}@ufsc.br

***Abstract*—In recent years, Approximate Computing has been gaining space as a technique for tackling energy requirements in error-resilient applications, while the usage of Machine Learning systems has steadily increased. This work explores two different approaches for approximation in comparator circuits and their impact on Decision Tree applications, observing power and accuracy metrics. Two gate-level architectures are proposed for dedicated comparators, approximating 25% or 50% of least significant bits using different techniques. The circuits were described in 7 nm FinFET technology. The approximate comparators were then evaluated in a Decision Tree classification model using five continuous and mixed attribute datasets. The 25% LSB approximate comparator proposed improves the energy efficiency in Decision Tree applications, reducing from 12% up to 84% the power per inference while presenting minor deviations in accuracy compared to the exact baseline.**

***Index Terms*—approximate computing, decision trees, energy efficiency, comparators**

I. Introduction

Nowadays, there is an increasing demand for energy-efficient designs, as battery life has become a significant factor in embedded and Internet of Things (IoT) devices. Moreover, there is growing concern regarding energy savings of applications in accordance with Green Computing practices [1] [2]. In this context, dedicated hardware solutions are demanded, as they can be associated with architectures optimized for energy efficiency. With the massive amounts of data currently created, many Machine Learning (ML) applications are available and inserted in the our daily life. This also brings the necessity for energy efficient ML systems [3], recalling on the demand for dedicated hardware solutions. ML algorithms generally involve a large number of operations that are independent, which enlarges the design space to include hardware optimization. These applications are also very resilient to errors, making Approximate Computing (AxC) [4] [5] a suitable approach.

AxC is an emerging research area that exploits the fact that many applications have soft constraints in terms of accuracy, trading the exactness of operations for significant energy savings [4] [6]. AxC techniques have been explored in both hardware and software in different contexts, such as Internet of Things (IoT) devices, video and audio processing, Machine Learning and other error-tolerant environments [7] [8] [9].

Recent projects have looked into developing low-power approximate solutions for ML applications, focusing particularly on Neural Networks (NNs) [10] [11]. However, the usage of NNs may still be costly in energy-restricted environments, for example due to the large number of multiplication operations required. In this scenario, simpler and less costly learning models might be preferred over NNs. For instance, Decision Trees (DTs) can provide satisfactory and concise results for a large number of inference problems [5] [12].

During the classification stage of Decision Trees, one of the most frequently executed operations is the comparison. These operations are processed to determine the target path on the tree for the value under classification, guiding the tree traversal. Optimizations on delay and power of comparator circuits can significantly reduce the resource requirements on a decision tree synthesis. Thus, optimizations and improvements in comparator circuits are critical for tree-based models, including functional approximation.

While most studies evaluating the usage of AxC techniques in ML models investigate architectural and software approaches, such as [13] [14], there are also some works investigating AxC arithmetic blocks [15]. However, most of these works concentrate on the proposal and development of approximate adders and multipliers, with little to no literature investigating AxC approaches in the design of energy-efficient comparators. To the best of our knowledge, this is the first prospective study about the design of power efficient comparator circuits for decision trees.

This work investigates the design of approximated comparator circuits and their usage in Decision Trees classification models, focusing on the gains in power efficiency within acceptable accuracy constraints. Since many ML applications specifically require the inference (or classification) step to operate in low-power environments [5], we focused on the classification operation of pre-trained trees. The main contributions of this work are: 1) proposing two energy efficient approximate comparator dedicated circuits; 2) exploring approximation in classification with Decision Trees; and 3) introducing a workflow for analyzing the energy savings of AxC comparators in tree-based applications of continuous and mixed datasets.

The remainder of this paper is organized as follows: Section II introduces the approximate comparator circuits designed. The approximation workflow for decision trees classification are detailed in Section III. Section IV discusses the evaluation of the approximated comparators on decision trees. Finally, the main conclusions are presented in Section V.

II. Proposed Approximate Comparator Circuits

The comparison operation is traditionally implemented in two main forms: with dedicated circuits or with subtractors. Traditional dedicated comparators are designed to perform greater, equal or lesser-than computations for a given number of bits, and may be chained to accommodate desired bit widths for the inputs. These architectures are generally used for unsigned operands. Some designs for AxC comparator blocks provide support for larger bit width, however, they are typically implemented following a architectural approach targeting timing optimization [16].

We propose an exact comparator targeting reduction in area and, consequently, power consumption, named Exact Dedicated Comparator (EDC). This circuit was designed by reducing a CMOS unsigned comparator circuit to perform only the *lesser-or-equal than* function. We note that other comparison operations can be trivially implemented by switching the input order or inverting the output. In this circuit, the branches that computed equal and less-than were removed, and the greater-than branch was fed through an inverter.

The general equation for n-bits of the EDC is presented in Eq. 1, comprised of three parts. The first line performs an equality test for both inputs, producing each EQ_i; the terms $\overline{G_i}$ represent a larger than test in each bit, accounting for all equality tests in more significant bits; and the last line computes the operation $A \leq B$.

For the sake of compactness, a 4-bit version of this exact dedicated comparator is presented in Fig. 1. The diagram structure follows the same idea as the n-bit comparators. This circuit is taken as the starting point and baseline for the proposed dedicated approximated comparators.

$$\begin{aligned} EQ_i &= \overline{A_i \oplus B_i} && , \ 0 < i < n \\ \overline{G_i} &= \overline{[\prod_{k=i+1}^{n-1} EQ_k] \cdot (A_i \cdot \overline{B_i})} && , \ 0 \leq i < n \\ A \leq B &= \prod_{i=0}^{n-1} \overline{G_i} \end{aligned} \tag{1}$$

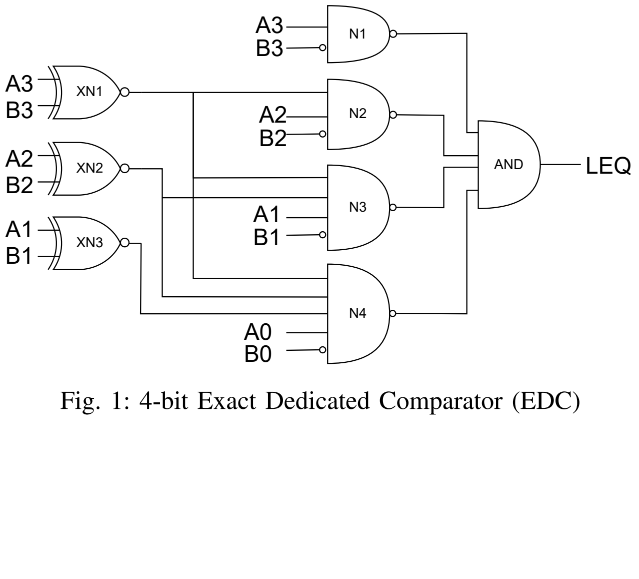


Fig. 1: 4-bit Exact Dedicated Comparator (EDC)

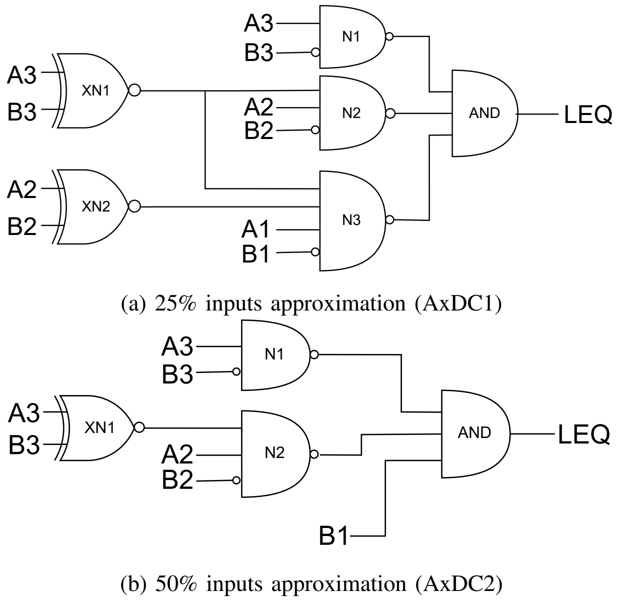
A. Dedicated Approximate Comparators

We propose two architectures for Approximate Dedicated Comparators (AxDCs): the AxDC1 and the AxDC2. The functional approximation technique involved removing logic gates from the EDC circuits while truncating the inputs. This approach is employed to improve power consumption while having little impact on the application accuracy. The for 4-bit versions of both comparators are shown in Fig. 2, as a simplification to introduce the idea, and to allow a direct comparison with the 4-bit EDC example.

The first dedicated approximate comparator is the AxDC1. The general equation to n-bit is presented in Eq. 2, in which the difference from the Eq. 1 is that the range for the operations is reduced to $n/4$ of the n more significant bits (MSB). The approximation was achieved by truncation of the Least Significant Bits (LSBs). Thus, in the 4-bit example, the NAND4 and XNOR3 gates were removed from the EDC circuit.

$$\begin{aligned} EQ_i &= \overline{A_i \oplus B_i} && , \ \tfrac{n}{4} < i < n \\ \overline{G_i} &= \overline{[\prod_{k=i+1}^{n-1} EQ_k] \cdot (A_i \cdot \overline{B_i})} && , \ \tfrac{n}{4} \leq i < n \\ A \leq B &= \prod_{i=\frac{n}{4}}^{n-1} \overline{G_i} \end{aligned} \tag{2}$$

The AxDC2 implementation is shown in Eq. 3. This version limits the range of exact operations to $n/2$ of the n more significant bits (MSB), also truncating the $n/4$ LSBs. For the 4-bit version, the gates XNOR2 and NAND3 were also removed from the EDC circuit, and the bit 1 from the second input (B) was routed directly to the last AND gate. This resulted in an approximation of 50% of input bits and further reduction in area.



(a) 25% inputs approximation (AxDC1)

(b) 50% inputs approximation (AxDC2)

Fig. 2: Approximate Dedicated Comparators (AxDCs)

$$\begin{aligned} EQ_i &= \overline{A_i \oplus B_i} & , \tfrac{n}{2} < i < n \\ \overline{G_i} &= \overline{[\prod_{k=i+1}^{n-1} EQ_k] \cdot (A_i \cdot \overline{B_i})} & , \tfrac{n}{2} \leq i < n \\ A \leq B &= \prod_{i=\frac{n}{4}}^{\frac{n}{2}-1} B_i \ \cdot \ \prod_{i=\frac{n}{2}}^{n-1} \overline{G_i} \end{aligned} \tag{3}$$

These approximations can be explored in n-bits circuits, which are constructed by obeying the same conditions of approximation shown in Eqs. 2 and 3. For these experiments, we employed 8-bit versions of the approximate circuits, analogous to those presented in the schematics for 4-bits. For example, the AxDC2 would ignore bits 0 and 1 from both inputs, and pass bits B2 and B3 to the last AND gate.

B. Subtraction-based Comparators

In this work, the proposed dedicated approximate comparators are confronted with comparators based on subtraction. These circuits use full adders (FAs) to perform subtraction and analyze a specific output for the operation required, for either unsigned or signed inputs. In the case of unsigned operands and the lesser-or-equal-than operation, we evaluate the carry-out of the most significant bit (MSB) full adder (FA) as the desired output. This approach also presents the benefit of easy scalability, achieved simply by increasing the size of the subtractor implemented.

For our case study, the subtraction-based comparators were designed with a simple Ripple Carry Adder (RCA). In these circuits, the approximation was achieved by substituting exact FA cells with three different approximate topologies: the Simplified Mirror Adder (SMA), the Approximate Mirror Adder 1 (AMA1), and the Approximate Mirror Adder 2 (AMA2) [17]. The circuits of these approximate FAs are presented in Fig. 3. These AxC FAs are explored due to the good outcomes of lower error distance and power savings compared to the Mirror and SERF Full Adders [18].

While dedicated circuits tend to be faster and more energy efficient, the downsides of subtraction-based comparators can be minimized using specialized low-power or faster architectures, as the RCA design used generates longer carry chains which result in increased delay.

III. Electrical Evaluation of the Comparators

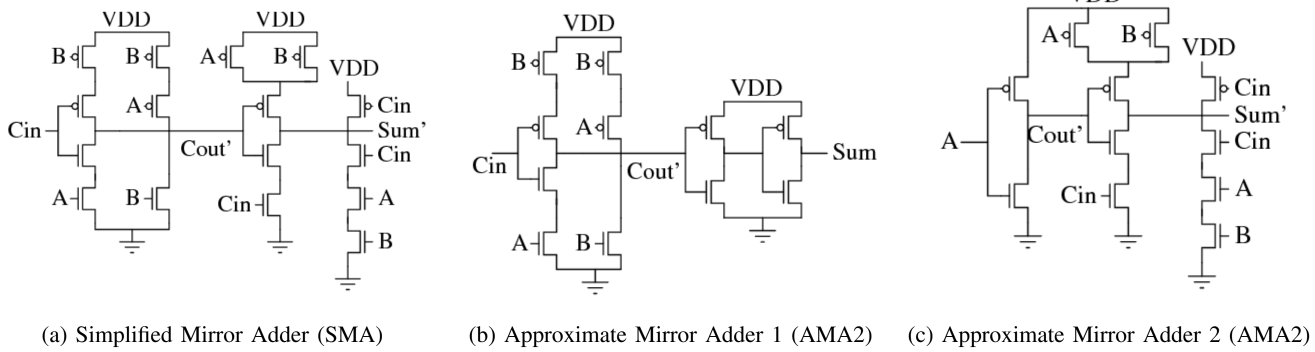
The electrical evaluation of the comparator circuits begins with the description of the Exact Dedicated Comparator and each proposed circuits for 8-bit extensions of the EDC and AxDCs, as well as the RCA with each approximated FA topologies for the subtractor-based comparators. We note that the proposed approaches can be easily adapted to larger inputs according to the range of the data in the datasets, but for our case study, 8-bit versions were justified by little impact in accuracy from quantization of the data.

The circuits were described adopting the ASAP7 7 nm FinFET PTM provided by Arizona State University (ASU) [19]. All devices use 3 fins, recommended as the minimum sizing for standard cell design [19]. Delay and power consumption were extracted for each circuit by simulation with Synopsys HSPICE. The circuits were electrically characterized under nominal voltage of 0.7 V, utilizing a 1 fF capacitor as load, equivalent to a fan-out of 4, to emulate a more realistic scenario.

Due to the large number of inputs of 8-bit comparators, the critical delay and power were analytically estimated, using the critical path and logic cell electrical characterization. Finally, the Total Error Distances (ED) and Error Rates (ER) were also calculated through the Truth Tables of the 8-bit approximations and considered in the analysis.

Table I presents the electrical characteristics of delay and power for the 8-bit comparator circuits evaluated in this work, as well as the ED and ER for each circuit. The first line describes the EDC. The next two entries present the results for the approximate dedicated versions (AxDC1 and AxDC2) proposed in this work. Finally, the remaining lines contain the results for the subtractor-based comparators with approximation (SMA, AMA1 and AMA2).

The AxDC1 shows the best results for error metrics among all the circuits evaluated, while presenting 24% of delay reduction and 11% of power savings. Higher improvements on power and delay are achieved with the version 2 of the proposed approximations, AxDC2, which reduces over 44%



(a) Simplified Mirror Adder (SMA) (b) Approximate Mirror Adder 1 (AMA2) (c) Approximate Mirror Adder 2 (AMA2)

Fig. 3: Approximate FAs inspired by the Exact Mirror Adder (EMA) [17] [18]

TABLE I: Electrical Characteristics and Error Metrics for the 8-bit Comparator Circuits

Circuit	Delay (ps)	Power (nW)	ED [†]	ER (%)
EDC	79.45	1040.73	0	0
AxDC1	60.25	916.21	384	0.59
AxDC2	44.05	720.75	24,384	37.21
SMA	104.56	4,319.44	10,795	16.47
AMA1	75.60	5,278.00	10,795	16.47
AMA2	69.76	3,346.48	16,384	25.00

[†] The maximum ED for 8 bit comparators is 65,536.

and 30% for delay and power respectively. However, this circuit has the largest ED due the 50% approximation of the inputs.

The Mirror Adder inspired approximations explored in the subtrator-based comparators show delay reduction for the AMA versions. However, all subtractor comparators evaluated had increased power consumption compared to the EDC, and presented ER superior to 16%. These results indicate that the dedicated approach is more advantageous for limited energy environments and the approximations proposed in this work can elevate the power efficiency, observing the quality restrictions of the target application, while subtractor-based comparators are more suitable for applications which already provide approximate adder/subtractor modules.

IV. Approximation Workflow for Decision Trees Classification

As a case study for this analysis, we used the General Public License (GPL) version of the C5.0 Decision Tree implementation [20]. C5.0 is an improved version of one of the most popular algorithms for training Decision Trees developed by Ross Quinlan, called C4.5. The C5.0 version contains improvements in terms of memory and computing resources, as well as extended features like boosting. For continuous attributes, C5.0 compares their value with a threshold that is tuned in the classification step. For categorical features, C5.0 maps their value as indices to an array of sub-trees (the branches of a node). Therefore, our approximate operators only work on continuous attributes. The comparison operations performed in the classification step were modified in the C5.0 source code in order to enable precise and approximate comparators.

This experiment consisted in implementing a method for evaluation of each architecture in the context of C5.0 classification. Our workflow, summarized in Fig. 4 was comprised of 4 stages: (1) pre-processing, (2) model training and classification, (3) power/energy estimation, and (4) comparison of results.

In this study, we employed 8-bit comparators, as they were large enough to maintain performance in accuracy, and provide good results in energy savings. Due to the fixed bit width in the datasets, a quantization pre-processing stage (1) was necessary. This step prepared the datasets for use as inputs of the unsigned 8-bit versions of the comparators proposed. The quantization consisted of two steps: a scaling step, fitting the continuous-attribute data in the range of 0 and $2^8 - 1$; then truncating the values to remove fractional information that went beyond the limits of representation.

The model training and classification stage (2) consisted of running classification for each tested dataset. The baseline results are generated training and testing the original dataset on the default C5.0. The approximate versions are trained with the default C5.0 version adopting the results from the pre-processing performed in the Stage 1. Thus, the test of the classification adopts the EDC and approximated versions of the comparator on the C5.0, emulating the results of using each approximation circuit in the classification step.

The approximate versions of C5.0 were trained on 5 continuous and mixed attribute datasets provided by the University of California Irvine (UCI) Machine Learning Repository [21]. The results were then analyzed using the unaltered C5.0 version as benchmark, focusing on the error rates (ER) on both training and test data, which correspond respectively to 80% and 20% of each complete dataset. The impact on the quality of the Decision Tree output was analyzed using the accuracy obtained on each approximation.

The third stage includes an electrical evaluation of all comparisons done on the approximate operations added to

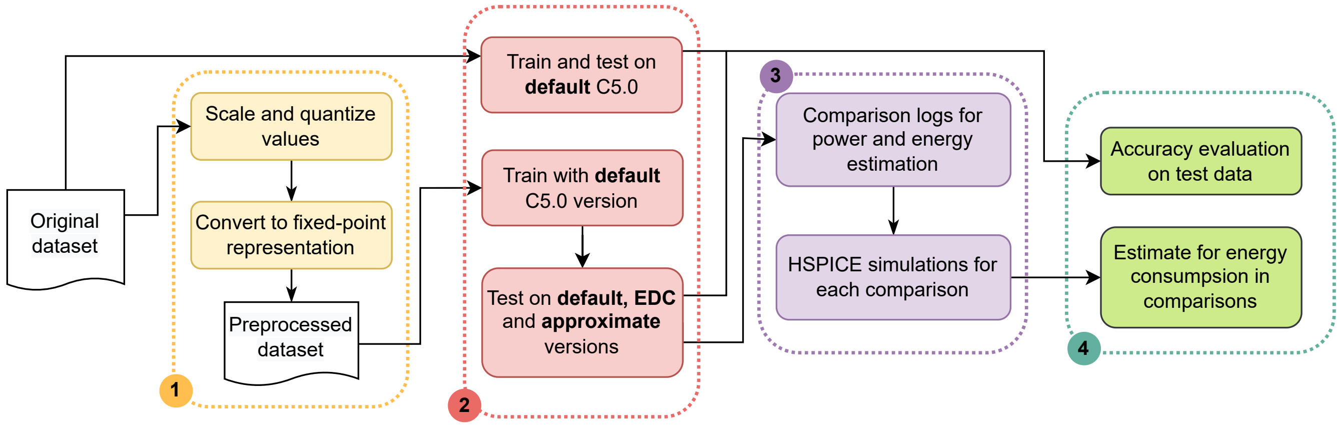

Original dataset
Scale and quantize values
Convert to fixed-point representation
Preprocessed dataset
1
Train and test on default C5.0
Train with default C5.0 version
Test on default, EDC and approximate versions
2
3
Comparison logs for power and energy estimation
HSPICE simulations for each comparison
Accuracy evaluation on test data
Estimate for energy consumption in comparisons
4

Fig. 4: AxC comparators evaluation in C5.0 workflow

the C5.0 source code in stage 2. A log of all continuous attribute test comparisons performed in the second stage is recorded. These logs saved during classification were used to generate sources for electrical simulation using the circuit descriptions generated for electrical characterization. All the operations were then simulated with HSPICE, with duration of 2.5 ns under nominal voltage (0.7 V). The energy data was collected from the simulations, which were summed to provide an estimate of the total energy consumption for each comparator in the entire test set classification.

The last stage (4) was the evaluation of all produced metrics, to provide clearer insight into the characteristics and behavior of each approximation in a given dataset. We analyze accuracy, total energy consumption for all comparison operations executed, and the number of operations performed. With these values, we can also calculate the average power consumption of the comparators per inference.

V. Evaluation on Decision Trees

Table II summarizes the results obtained for each dataset employed, presenting the number of test samples and attributes in each case. Furthermore, the table includes the accuracy (Acc) achieved in classification, the energy estimate calculated and the number of comparison operations (#Ops) done in continuous attributes. Note that a larger number of continuous attributes will incur in greater energy consumption.

In terms of prediction performance, we can observe that the adoption of approximated comparators on the classification of continuous datasets has varying effects on the accuracy compared to the exact version. The EDC presents a very similar behavior to the default C5.0 version, with an average deviation of only 0.01%, caused by the quantization error. All approximated comparators were thus analyzed against the EDC, to account for both accuracy and energy performance. For the approximated dedicated comparators, the AxDC1 showed the closest accuracy to the EDC, with a reduction on accuracy of only 0.12% on average for evaluated datasets. This is mostly due to the nearly negligible error rate of this comparator.

The AxDC2 reaches the higher energy reduction with more than 51% of savings compared to the EDC circuit for all datasets evaluated. The highest impact on energy is observed on the Iris dataset (86.64%) despite the fact that this approximate circuit increases the number of comparison operations executed in 81.8%. Despite reductions in resource consumption, the accuracy for this circuit was significantly worse than most other evaluated comparators, mostly due to its large error metrics.

The variable number of operations seen in Tab. II is explained by the different paths taken in the traversal of the trees when performing classification with each approximation. This variation is directly related to the energy consumption of each inference performed. In fact, the AxDC1 shows an average reduction of 21.2% in the number of comparisons, while the AxDC2 has an increase of 51.1%. This explains the similar energy reductions obtained in both AxDC1 and 2, of 51% and 46.3% respectively, especially relevant considering the much more accurate results of the AxDC1. Notably, this effect could be minimized by also applying the approximations the construction of the DTs, in which the threshold for continuous tests would be tuned for each different circuit.

Observing the experiment results by the metric of the power per inference, we highlight the fact that the power consumption calculated in this workflow is impacted directly by the number of continuous attributes which are present in the target dataset. For example, the Arrhythmia dataset shows the highest power per inference due to the presence of a total of 279 attributes, of which 206 were continuous. With the SMA-based comparator each inference in this dataset consumes 10.08 μW, while the adoption of the AxDC2 significantly reduces the power per inference to 1.77 μW. This difference reinforces how the approximate comparator proposed in this work can be applied in the DT classification, achieving high power efficiency with little to no interference on the accuracy of the application. We can also note that the AMA1-based comparator also proves to be a good choice, with an average reduction of 32% in power per inference when compared to the EDC. The AxDC1 provided reduction of 12.5% up to 84.95% in power per inference, with an average of 46.25% on all datasets.

VI. Conclusions

This work proposed two architectures for dedicated approximate comparators, evaluating their usage in the classification stage of Decision Trees. Along with this, we also evaluated another approach utilizing subtractors and approximated FAs. All circuits were described and studied in 8-bit versions, applied to the classification stage of the a Decision Tree Classifier. The proposed dedicated circuits circuits showed good overall results in electrical characteristics, with reduction of up to 31% in power consumption in relation to the exact baseline comparator. However, the AxDC2 provided significantly worse accuracy, compared to the remaining comparators. We also

TABLE II: Evaluation of the Comparison Circuits on Decision Trees Classification Model

	Heart disease Test samples = 60 Attributes = 13 (6 continuous)			**Iris** Test samples = 30 Attributes = 5 (4 continuous)			**Arrhythmia** Test samples = 90 Attributes = 279 (206 continuous)			**Adult** Test samples = 16281 Attributes = 15 (6 continuous)			**Forest fires** Test samples = 56 Attributes = 13 (10 continuous)		
Comparator circuit	Acc (%)	Energy (fJ)	# Ops	Acc (%)	Energy (fJ)	# Ops	Acc (%)	Energy (fJ)	# Ops	Acc (%)	Energy (pJ)	# Ops	Acc (%)	Energy (fJ)	# Ops
EDC	63.3	160.7	120	93.3	35.8	88	63.3	1,003.1	1,195	86.4	73.7	115,704	98.2	94.4	184
AxDC1	63.3	140.6	127	93.3	5.4	75	63.3	725.8	949	85.9	21.2	107,532	98.2	61.4	56
AxDC2	40.0	76.8	138	30.0	4.8	160	7.8	398.8	1,734	23.5	58.5	176,441	39.3	58.4	296
SMA	55.0	601.7	141	73.3	95.4	121	16.7	2,267.5	1,653	77.6	185.7	211,991	78.6	389.5	129
AMA1	55.0	174.2	21	73.3	16.1	30	16.7	334.2	456	77.6	29.5	96,287	78.6	106.1	102
AMA2	46.7	253.5	138	33.3	105.9	160	6.7	1,610.5	1,740	24.2	344.5	192,574	60.7	390.6	296

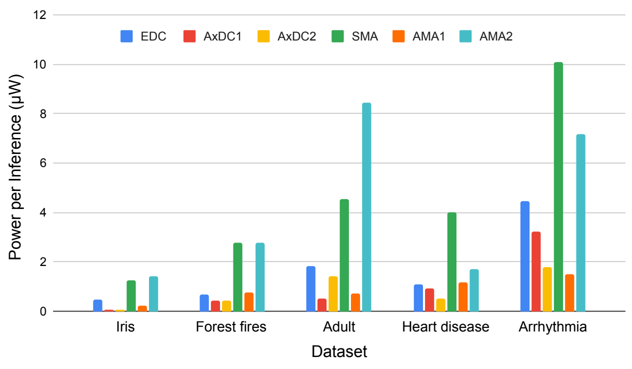


Fig. 5: Evaluation of the Power Consumption per Inference

investigated the effects of using subtractor-based approximated comparators, which provide greater design usability in applications where AxC arithmetic blocks are present, with the caveat of worse performance when compared to the dedicated approach. Overall, the AxDC1 showed the best results in error metrics, just of 0.59% error rate.

In general, the dedicated circuits show more advantages in accuracy and energy savings compared to the versions based on subtractors exploring approximations based on Mirror Adder. From the FA-based comparators, the AMA1 was especially competitive, providing gains of 32% in energy and at the cost of 27.7% in accuracy, by decreasing the number of operations by more than 54%, the largest reduction in number of comparisons between evaluated options. The advantage of this approach could be in an approximate environment, where there is the demand for an approximate adder/subtractor and the hardware can be reused. However, for dedicated hardware design, the AxDC1 was still considered the best option regarding the power-accuracy trade-off.

The proposed workflow has demonstrated to be able to deal with mixed datasets, and data larger than the 8-bit representation limits used. Next steps include to include approximation in the decision tree construction, and also combine other energy efficient approaches together with the proposed comparator circuits in a configurable environment.

Acknowledgment

This work was financed in part by National Council for Scientific and Technological Development – CNPq and the Propesq/UFSC.

References

[1] S. Singh, "Green computing strategies & challenges," in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*. IEEE, 10 2015, pp. 758–760. [Online]. Available: http://ieeexplore.ieee.org/document/7380564/

[2] H. B. Barua and K. Chandra Mondal, "Green data mining using approximate computing: An experimental analysis with rule mining," in *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, 2018, pp. 115–120.

[3] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, "Efficient machine learning for big data: A review," *Big Data Research*, vol. 2, pp. 87–93, 9 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2214579615000271

[4] J. Han, "Introduction to approximate computing," in *Proceedings of the IEEE VLSI Test Symposium*, vol. 2016-May. IEEE, apr 2016, pp. 1–1. [Online]. Available: http://ieeexplore.ieee.org/document/7477305/

[5] B. A. Abreu, M. Grellert, and S. Bampi, "Vlsi design of tree-based inference for low-power learning applications," in *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 2020-Octob. IEEE, 10 2020.

[6] H. B. Barua and K. C. Mondal, "Approximate computing: A survey of recent trends—bringing greenness to computing and communication," *Journal of The Institution of Engineers (India): Series B*, vol. 100, pp. 619–626, 12 2019. [Online]. Available: http://link.springer.com/10.1007/s40031-019-00418-8

[7] T. Moreau, A. Sampson, and L. Ceze, "Approximate Computing: Making mobile systems more efficient," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 9–13, apr 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7093019/

[8] A. G. Strollo and D. Esposito, "Approximate computing in the nanoscale era," in *ICICDT 2018 - International Conference on IC Design and Technology, Proceedings*. IEEE, 6 2018, pp. 21–24. [Online]. Available: https://ieeexplore.ieee.org/document/8399746/

[9] D. Marwaha and A. Sharma, "A review on approximate computing and some of the associated techniques for energy reduction in iot," in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. IEEE, 1 2018, pp. 319–323. [Online]. Available: https://ieeexplore.ieee.org/document/8399087/

[10] B. Zhang, A. Davoodi, and Y. H. Hu, "Exploring energy and accuracy tradeoff in structure simplification of trained deep neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, pp. 836–848, 12 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8354792/

[11] A. Goel, C. Tung, Y. H. Lu, and G. K. Thiruvathukal, "A Survey of Methods for Low-Power Deep Learning and Computer Vision," in *IEEE World Forum on Internet of Things, WF-IoT 2020 - Symposium Proceedings*, mar 2020. [Online]. Available: http://arxiv.org/abs/2003.11066

[12] Q. Li and A. Bermak, "A low-power hardware-friendly binary decision tree classifier for gas identification," *Journal of Low Power Electronics and Applications*, vol. 1, pp. 45–58, 3 2011. [Online]. Available: http://www.mdpi.com/2079-9268/1/1/45

[13] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient machine learning in 2 kb ram for the internet of things," in *34th International Conference on Machine Learning, ICML 2017*, vol. 4. Springer New York, 2017, pp. 3062–3071. [Online]. Available: http://link.springer.com/10.1007/978-1-4614-7138-7

[14] E. García-Martín, N. Lavesson, H. Grahn, E. Casalicchio, and V. Boeva, "Energy-aware very fast decision tree," *International Journal of Data Science and Analytics*, vol. 11, pp. 105–126, 3 2021. [Online]. Available: https://link.springer.com/10.1007/s41060-021-00246-4

[15] M. Osta, A. Ibrahim, H. Chible, and M. Valle, "Approximate multipliers based on inexact adders for energy efficient data processing," in *2017 New Generation of CAS (NGCAS)*, Sep. 2017, pp. 125–128.

[16] Y. Zhou, J. Lin, J. Wang, and Z. Wang, "Approximate comparator: Design and analysis," in *IEEE Workshop on Signal Processing Systems, SiPS: Design and Implementation*, vol. 2018-Octob. IEEE, 10 2018, pp. 129–133. [Online]. Available: https://ieeexplore.ieee.org/document/8598366/

[17] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: Imprecise adders for low-power approximate computing," in *Proceedings of the International Symposium on Low Power Electronics and Design*. IEEE, 8 2011, pp. 409–414. [Online]. Available: http://ieeexplore.ieee.org/document/5993675/

[18] P. A. Silva and C. Meinhardt, "Energy-efficient design of approximated full adders," in *ICECS 2020 - 27th IEEE International Conference on Electronics, Circuits and Systems, Proceedings*. IEEE, 11 2020.

[19] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 7 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S002626921630026X

[20] J. R. Quinlan, *C4.5 - Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[21] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml