Efficient Random Forest Acceleration for Edge Computing Platforms with FPGA Technology

Cuong Pham-Quoc [1,2,*], Trung Pham-Dinh [1,2], Binh Kieu-Do-Nguyen [3]

[1] Department of Computer Engineering, Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam
[2] Department of Computer Engineering, Vietnam National University-Ho Chi Minh City (VNU-HCM), Thu Duc, Ho Chi Minh City, Vietnam
[3] Very Large-Scale Integration (VLSI) Lab, University of Electro-Communications (UEC), Tokyo, Japan
Email: cuongpham@hcmut.edu.vn (C.P.-Q.); trung.pham.ktmt@hcmut.edu.vn (T.P.-D.);
binh@vlsilab.ee.uec.ac.jp (B.K.-D.-N.)
*Corresponding author

***Abstract*—As one of the most successful instances of ensemble learning algorithms, Random Forest offers many advantages compared to other approaches. However, it is unsuitable for edge computing platforms due to its high computational power. In this paper, we present our proposed efficient architecture to perform random forest effectively for edge computing platforms based on Field Programmable Gate Array (FPGA) technology. The heart of the system is our Decision Tree Unit (DTU) architecture, which is mainly responsible for processing decision trees in the pipeline to achieve better performance. One of the biggest obstacles to decision tree implementation on hardware is the memory size. In this paper, we also propose a sufficient structure for storing decision trees' information for the execution of DTUs. Since we target edge computing platforms with limited resources and energy, the architecture supports the scalability of the number of DTUs in the system. Based on the available resources of the target platform, the system can be reconfigured accordingly. We implement our prototype version with the PYNQ Z2 FPGA edge computing board. We test the proposed system with the number of DTUs changed from 1 to 15. We conduct experiments and analysis with a certified dataset and compare with Intel core i7 and core i9 processors to show our efficiency and scalability. The experimental results show that we can achieve speed-ups by up to 19.96× compared to the Intel Core i7 desktop version and 12× compared to the Intel Core i9 high-performance computing version. Regarding energy consumption, we save up to 33.24× and 146.24× compared to the two processors.**

***Keywords*—Field Programmable Gate Array (FPGA) technology, decision tree, random forest acceleration, edge computing platforms**

I. Introduction

Random Forest (RF) is a successful example of supervised learning that has many applications in various fields, including finance and banking, e-commerce, and healthcare. It is particularly useful when dealing with datasets that have a large number of features but fewer samples [1], outperforming other machine learning models in classification tasks involving mixed data types [2]. However, RF's computation and resource requirements, especially for large datasets, make it challenging to implement on edge computing platforms where computing power, storage, and energy capacity are limited [3].

Field Programmable Gate Arrays (FPGAs) are a promising technology for edge computing, providing ample hardware resources for parallel processing [4]. They are also low-cost platforms that are well-suited for machine learning inference in IoT/edge computing applications [5]. To address the challenges of implementing RF on FPGA-based edge computing platforms, this paper proposes a scalable and efficient architecture. The proposed architecture includes multiple Decision Tree Units (DTUs) that can process decision trees in parallel, allowing for efficient processing of RF. The DTUs have a pipeline architecture to improve performance, and decision trees are divided into subsets for pipeline processing. To address storage requirements, an efficient memory structure that uses 4 bytes per node is proposed. The architecture is also configurable to meet specific requirements, such as maximizing performance, minimizing hardware resources, or reducing energy consumption.

The Xilinx PYNQ-Z2 edge computing platform with a Xilinx MPSoC Zynq FPGA device [6] is used to build a testing system based on the hardware accelerator paradigm [7]. The proposed architecture is implemented on the FPGA fabrics, while the ARM-hardwired processor is responsible for preprocessing data and determining the final results based on random forest computation. To assess scalability, the system's hardware resources, and power consumption are evaluated with various numbers of DTPs, ranging from 1 to 15. To test the efficiency of the proposed architecture and design, a certified dataset is used to evaluate the random forest system's performance with the number of DTUs ranging from 1 to 15 DTUs.

The experimental results demonstrate that with 15 DTUs, the system requires approximately 29K LUTs and

Manuscript received on June 25, 2023, revised July 11, 2023; accepted August 17, 2023; published February 5, 2024.

35K FF. The FPGA-based accelerator system with 15 DTUs achieves speed-ups of up to 19.96× and 12× when compared to Intel core i7 and Intel core i9 processors, respectively. In addition, the system also saves up to 33.24× and 146.24× energy consumption compared to the two processors.

The key contributions of the paper can be summarized as follows.

1) We propose an architecture for efficiently accelerate random forest on FPGA-based edge computing platforms. The proposed architecture is scalable when the number of DTUs can be reconfigured to quickly adapt to available resources of the target platforms.
2) We present our pipeline processing DTU's architecture with sufficient memory structure to store and perform decision tree's parameters with high-throughput and low resources required.
3) We test the system with a real FPGA-based edge computing platform and a certified dataset to report experimental results for future research reference.

The rest of the paper is organized as follows. Section II presents the background of our work and related work in the literature. We present our proposed architecture in Section III. The FPGA-based implementation of our prototype system is introduced in Section IV. We discuss our experiments with two different datasets in Section V. Finally, Section VI concludes our paper.

II. Background and Related Work

In this section, we present the background of random forests for designing the proposed architecture. We also introduce related work on accelerating random forest on FPGA.

A. Random Forest

The ensemble learning algorithm called Random Forest is utilized to address classification and regression problems. Initially introduced by Ho [8], it involves a collection of decision trees that contribute to the final prediction outcome. A decision tree is a binary tree comprising of internal and leaf nodes. Each internal node possesses two child nodes and executes a test on a sample's attribute to determine the subsequent branch to follow. This test compares a feature with a decision rule referred to as a threshold value. During the training process of a decision tree, appropriate thresholds are determined for each internal node.

In contrast, a leaf node lacks child nodes and instead holds a prediction value, which may be a class label for classification or a numeric value for regression. To make predictions using a decision tree, a sample is introduced at the root node and traverses through the tree until it reaches a leaf node. Decision trees are trained using the bootstrap aggregating (bagging) algorithm [9]. Ultimately, the prediction result of a random forest is obtained by calculating the mean or average (for regression) or majority (for classification) of the outcomes derived from the individual trees.

B. Related Work

To implement random forest, similar to other machine learning approaches, both training and inference phases are required, with the training phase typically performed offline. Consequently, significant research efforts have been dedicated to accelerating the inference phase to enhance processing performance. In Ref. [10], three architectures, namely memory-centric, comparator-centric, and synthesis-centric, were introduced. Zhao *et al.* [11] eliminated floating- point execution by pre-computing and storing floating-point values in local memory. Damiani and Sozzo *et al.* [12] employed a novel partial reconfiguration technique to update large random forest models. Jinguji *et al.* [13] and Ikeda *et al.* [14] optimized comparisons to accelerate random forest on FPGA. For decision trees, a 2-dimensional pipeline architecture was proposed by Qu and Prasanna [15], while a RISC-like architecture was introduced by Alcolea and Resano [16] to achieve higher performance in random forests. Furthermore, Oberg and Eguro *et al.* [17] aimed at enhancing random forests for computer vision applications.

These proposals specifically target modern and high-end FPGA platforms equipped with abundant hardware resources and nearly unlimited energy. In contrast, this work concentrates on edge computing platforms characterized by limited resources and energy constraints. Hence, a scalable architecture has been designed, allowing for the quick scaling up or down of the number of DTUs (Decision Tree Units) based on the available resources.

III. Proposed Architecture

In this section, we present our proposed architecture designed to accelerate random forest specifically for edge computing platforms. We begin by introducing our architecture, which aims to enhance the processing speed of random forest. Subsequently, we outline our Decision Tree Unit (DTU) architecture, which incorporates the pipeline technique to further optimize performance. Lastly, we describe the memory structure utilized for storing decision trees.

A. The Generic System Architecture

Fig. 1 depicts the FPGA-based generic architecture of our proposed random forest acceleration system, designed to be scalable and efficient for edge computing platforms. In this architecture, the responsibility of the software aspect of the random forest-based application, including data pre/post- processing, I/O management, and network communication, lies with a host processor. This host processor can be an embedded hardwired processor for MPSoC FPGA devices or a soft- processor for standard FPGA devices.

The host processor is connected to the FPGA fabrics and a main external memory, which is utilized for storing the application's data. Additionally, a Direct Memory Access (DMA) block is incorporated to facilitate data transfers between the main memory and the local memory in the programmable logic. These connections are established through a communication infrastructure, typically a bus-based interconnect.

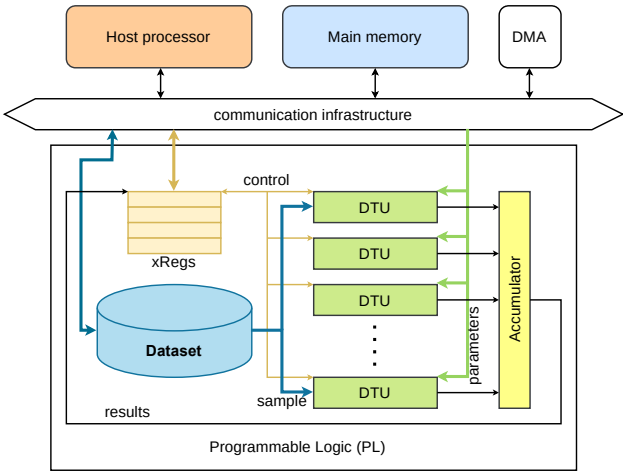

Fig. 1. The generic architecture of the proposed system.

The primary contribution of this research lies in the architecture implemented within the programmable logic (FPGA fabrics), which consists of multiple DTUs (Decision Tree Units) responsible for processing decision trees. The details of the DTU architecture are presented in the subsequent section. To facilitate the processing of DTUs, a set of exchange registers (xRegs) is employed for passing arguments and managing the initiation and completion of DTUs. Furthermore, a local buffer is constructed to store datasets (samples) processed by the DTUs, enabling decision-making operations. The Accumulator module computes the outputs generated by the DTUs for classification and regression techniques. Importantly, the number of DTUs can be reconfigured prior to synthesis, allowing the system to be adaptable to diverse FPGA-based edge computing platforms characterized by limited resources and energy constraints.

B. DTU Local Memory Structure

To process decision trees within random forests, the structures and parameters of the trees are stored in the local memory of the DTUs. Before delving into the proposed architecture of a DTU, this section outlines the structure of the DTU's local memory, which serves as the foundation for tree collection and processing. The memory structure of our DTU is illustrated in Fig. 2.

Given that our DTU follows a 5-stage pipeline model (re- quiring five cycles to process a node), decision trees assigned to a DTU are partitioned into five subsets. This allows five trees (one from each subset) to be processed concurrently within the pipeline. Thus, the initial five memory words of a DTU's memory store the addresses of each subset in memory (as shown in the subset addresses section of Fig. 2), with the last bit indicating whether it represents the final subset or not. As depicted in the figure, the trees are stored contiguously within each subset. Since the sizes of the trees (i.e., the number of nodes in a tree) may vary, instead of allocating fixed space for each tree, we store the relative address of the next tree in the leaf nodes. Each tree consists of two types of nodes, internal and leaf nodes, with each node requiring a 32-bit word, as indicated in the figure.

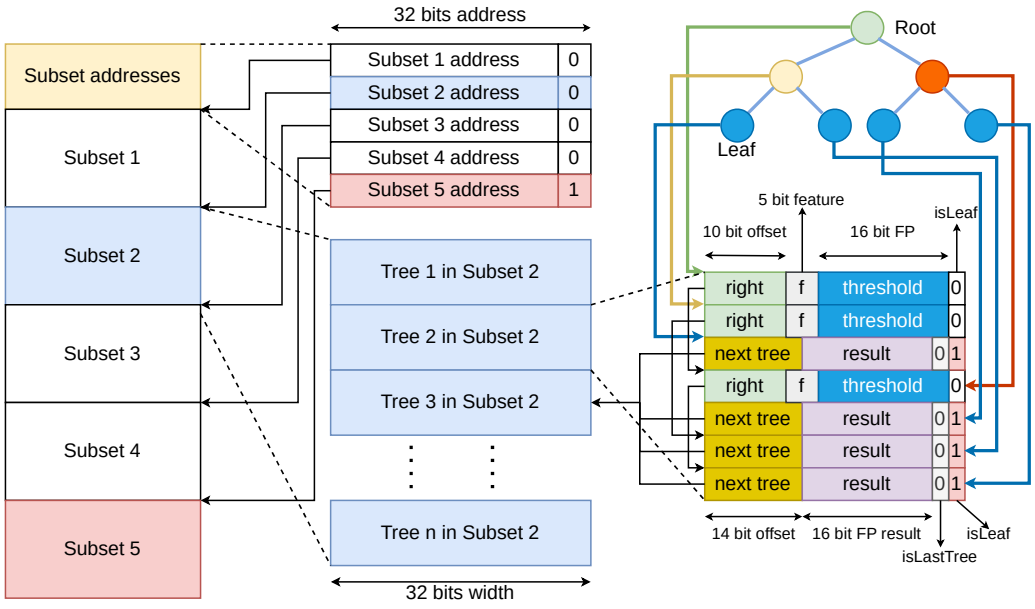

Fig. 2. The Block RAM-based memory structure of the proposed DTU.

Suppose the current node is an internal or root node (identified by the least significant bit, denoted as `isLeaf` = 0). In this case, the 10 most significant bits represent the relative address of the right node, while the subsequent memory word stores the leaf node. The following 5 bits in this word indicate the types of features stored by the node, allowing our decision trees to accommodate up to 32 features. The next 16 bits hold the value of the corresponding parameter in this node, referred to as `threshold`. To cater to different applications, floating-point values are used for threshold values. Depending on the sample being processed, the next node can either be the left (following node) or the right node. If the right node is the subsequent node, the 10-bit right relative address is

added to the current address to access the right node. Otherwise, the next memory word is selected.

If the current node is a leaf node (indicated by the least significant bit, `isLeaf` = 1), the result of this tree is obtained from the 14-bit floating-point result field. The second least significant bit denotes whether this tree is the last tree in the subset (`isLast` value). If it is not the last tree (`isLast` = 0), the most significant 14 bits represent the relative address of the next tree within the same subset. Conversely, if this is the last tree of the subset, the DTU completes the processing for that subset.

C. *Decision Tree Unit (DTU)*

The proposed FPGA-based pipeline DTU (Decision Tree Unit) architecture for processing a decision tree is illustrated in Fig. 3. Our DTU operates within a five-stage pipeline model, as shown in the figure. It completes the calculations required for each node in five cycles, with two cycles allocated for reading Block-RAM and three cycles dedicated to comparisons. This division of pipeline processing into five stages facilitates the efficient processing of decision trees.

The DTU architecture incorporates a two-cycle pipeline for the Block-RAMs, which serve as the storage for parameters and the structure of the decision tree. The first ports of the Block-RAMs are connected to the communication infrastructure, allowing data to be received from the DMA (Direct Memory Access), while the second ports are utilized by the DTU. Despite each read operation requiring two cycles, the Block-RAMs support pipeline reading and writing, enabling the DTU to request and retrieve data every cycle.

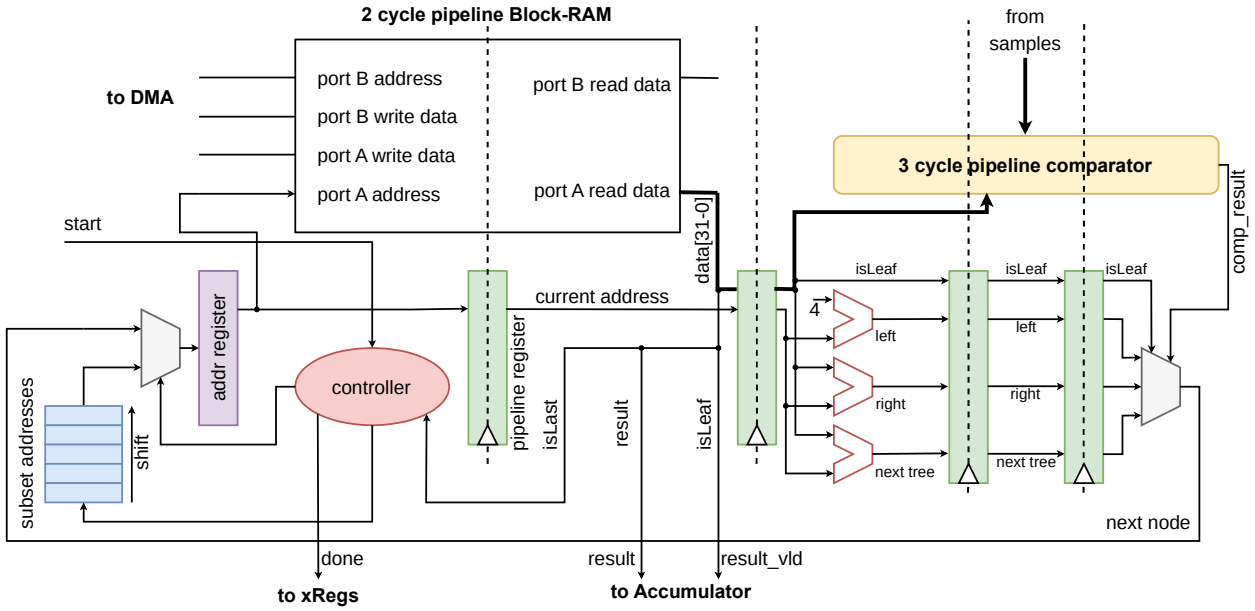

Fig. 3. The architecture of our proposed Decision Tree Unit (DTU).

For comparing values between samples and node parameters, a three-cycle pipeline comparator is employed. The comparator accepts data inputs and produces results every cycle, similar to the Block-RAM storage. During the first two stages of the pipeline, the DTU reads a node's parameter from the storage to fetch it to the comparator if the node is internal or sends it to the Accumulator if the node is a leaf node (activated when the `isLeaf` signal is active).

In the first stage, the Controller block determines the addresses required to access the Block-RAM storage. As mentioned previously, the DTU's processing is divided into five stages, and decision trees are assigned to each DTU, partitioned into subsets containing either n or n+1 trees per subset. These subsets are stored in the Block-RAM storage based on the memory structure of the DTU and are processed within the pipeline. During the initial five cycles, the Controller selects subset addresses to initiate the processing of the first tree in each subset. Subsequently, the address of the next node is determined by the comparator, which can be either the left node or the right node, depending on the comparison result. If a leaf node is reached, the next tree within the same subset is selected when the current tree is not the last tree (as indicated by the inactive state of the `isLast` signal). Processing of a subset is considered complete when either the last tree is reached or the `isLast` signal is active. The DTU's processing is considered finished when all subsets' `isLast` values are active.

IV. FPGA-Based Edge Computing Platform Implementation

To validate the proposed architecture of the generic system, DTU, and memory structure, we implemented the system using the Xilinx PYNQ-Z2 edge computing platform [18]. This platform features a Xilinx MPSoC FPGA Zynq 7000 xc7z020 device, which offers 53.2K LUTs (Lookup Tables), 106.4K FF (Registers), and 140 Block-RAMs (4.9 Mbit). The device also incorporates a 2-core ARM Cortex-A9 Application Unit Processor that serves as the host processor in our implementation.

The proposed architecture for the system and DTU is implemented using parameterized SystemVerilog, allowing for easy configuration of the number of DTUs. The Xilinx AXI- lite bus is utilized as the communication infrastructure to facilitate data transfer between the main memory and the local memories of the DTUs. Block-RAM IP cores are employed to store the parameters of decision trees, as per the structure described earlier.

In this implementation, we developed the Accumulator module, which supports both regression and classification techniques. To evaluate the proposed system, we utilized the California Housing Price application published by Kaggle [19] with the Scikit-learn California housing regression dataset [20], which consists of eight features. For testing purposes, we constructed 100 decision trees with a maximum depth of nine, dividing them into five subsets. The dataset used contains floating-point numbers representing both the features and the results.

We employed Xilinx Vivado 2022 [21] to perform synthesis and build the implemented system on the Xilinx PYNQ- Z2 platform. To assess scalability, we synthesized the system using various numbers of DTUs, ranging from 1 to 15 units. Furthermore, in order to evaluate efficiency in terms of performance and energy consumption, we processed the aforementioned datasets using 1, 5, 10, and 15 DTUs.

V. Experiments

This section focuses on the experimental validation and assessment of the acceleration capability of the aforementioned system. Firstly, we present the results of our synthesis process, considering different numbers of DTUs ranging from 1 to 15. Subsequently, we conduct evaluations to analyze the performance and energy consumption of the proposed system, aiming to validate its efficiency.

A. Synthesis Results

Following the description provided earlier, we synthesized the system utilizing a range of DTU quantities, specifically from 1 to 15. Table I displays the utilization of hardware resources, working frequency, and estimated power consumption for each system configuration. The synthesis process was conducted automatically without any imposed area constraints or manual optimization.

The table reveals that our system, when deployed with 15 DTUs, utilizes a maximum of 70% of the available Block- RAMs for storing decision trees' parameters and structure, and 56.33% of the computing resources (LUTs and FFs) on the chip. This indicates that there is still room to scale up the system by incorporating additional DTUs. However, it is worth noting that as more DTUs are added, the power consumption of the system also increases.

In terms of the working frequency, the system achieves the highest frequency when a moderate number of DTUs, such as 9, 7, 6, or 5, are employed. The frequency tends to decrease when the number of DTUs is either small or large, primarily due to longer physical paths required for routing hardware resources. This issue can potentially be mitigated by applying area constraints during the placement and routing process. However, addressing this matter falls beyond the scope of this paper.

B. Performance Analysis

In order to evaluate the efficiency of the proposed system, we compare it with the Intel Core i7-8565U 1.8 GHz (desktop version) and Core i9-9820X 3.30 GHz (high-performance version) processors. When running on CPUs, all cores of the CPUs are utilized to process the dataset.

As previously mentioned, we employ the Scikit-learn California housing regression dataset to assess the system using the regression technique. Our system processes one hundred decision trees using 1 to 15 DTUs. Additionally, we conduct the same testing with Intel CPUs to obtain the execution time of the Intel Core i7 and Core i9 CPUs when utilizing all cores. The execution time of our system as well as the CPUs is presented in Table I.

As indicated in the table, the execution time of our system does not exhibit a linear scaling pattern based on the number of DTUs employed. This is primarily because the system involves the transfer of data from the main memory to the local memories of the DTUs, and this data movement time is not scalable.

Fig. 4 illustrates a comparison of the speed-ups achieved by our system, utilizing different numbers of DTUs, with the Intel Core i7 (desktop version) and Core i9 (high-performance version) processors. As depicted in the figure, we observe speed-ups of up to 19.96× and 12.01× in comparison to the two Intel processors, respectively. Notably, the speed-up obtained with respect to the Core i9 processor is comparatively lower than that achieved with the Core i7 processor, given that the Core i9 processor belongs to the high-performance computing category, while the Core i7 processor is a desktop version.

TABLE I. Experiment Results (Synthesis, Execution Time) with Various Number of DTUs Used for the Proposed System and the Execution Time of Reference Intel Processors

HW	Number of Decision Tree Units (DTUs)										
	15	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
LUTs	29,968	20,885	19,775	18,436	16,784	15,543	13,429	12,243	10,887	9,396	7,932
FFs	35,651	25,308	24,155	22,133	20,716	19,194	17,204	15,644	14,017	12,426	10,779
BRAMs	98.0	83.0	93.5	92.0	85.0	86.0	84.0	85.5	86.5	84.5	82.5
Freq. (MHz)	125		167	143		167			143		125
Power (W)	2.204	1.877	2.089	1.946	1.978	1.855	1.824	1.696	1.663	1.66	1.588
Exec. time (ms)	1.340	1.716	1.652	1.832	1.648	1.975	2.040	2.572	3.265	4.330	8.841
Core i7 (ms)						26.750					
Core i9 (ms)						16.086					

C. Energy Consumption Analysis

The energy consumption is calculated by considering the execution time and power consumption of the Intel Core i7 and Core i9 processors, which are measured at 3.67 W and 26.849 W, respectively, during the dataset processing. With respect to energy consumption, the reconfigurable technology employed in our system enables significant energy savings. Specifically, our system can achieve energy savings of up to 146.24× and 33.24× compared to the Intel Core i9 and Core i7 processors, respectively. This outcome highlights the efficiency of our system, particularly for edge computing platforms. Although the speed-ups achieved with the Core i9 processor are comparatively lower than those with the Core i7 processor, our system still manages to save more energy compared to the Core i9. This discrepancy arises because the Core i9 processor is primarily designed for high-performance computing, which demands a considerable amount of power consumption. Fig. 5 presents the comparisons for energy consumption of our system and of Intel processors. As depicted in the figure, we manage to save up to 146.24× and 33.24× when compared to Intel Core i7 and Core i9, respectively.

In summary, with all the experiments and analysis presented above, we proved that our system is suitable for edge computing platforms where the computational ability is low with the energy limitation. However, our proposed system offers better system performance compared to Intel processors. In other words, our system is much more energy-efficient than traditional processors.

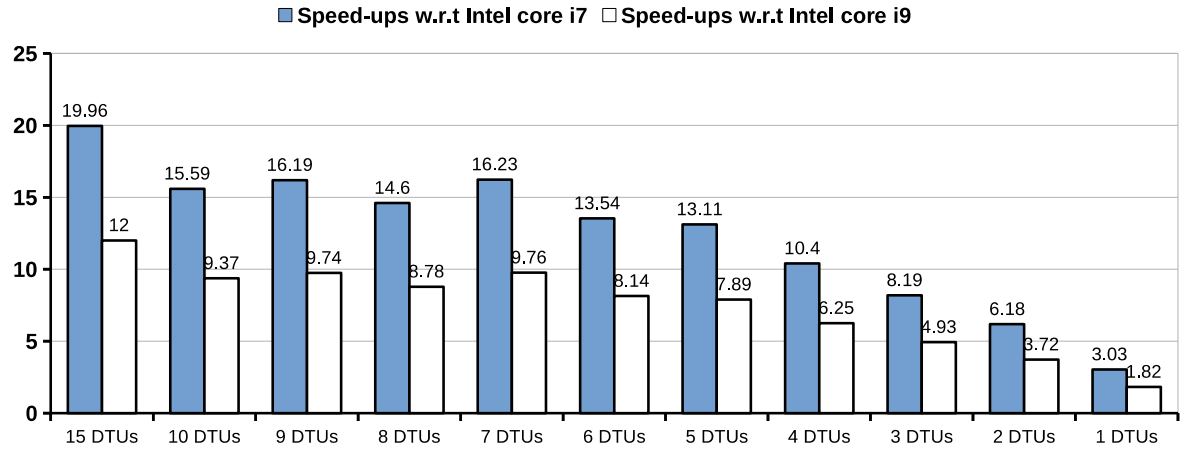

Fig. 4. Speed-ups of our system with respected to the Intel Core i7 and Intel Core i9 when processing the dataset.

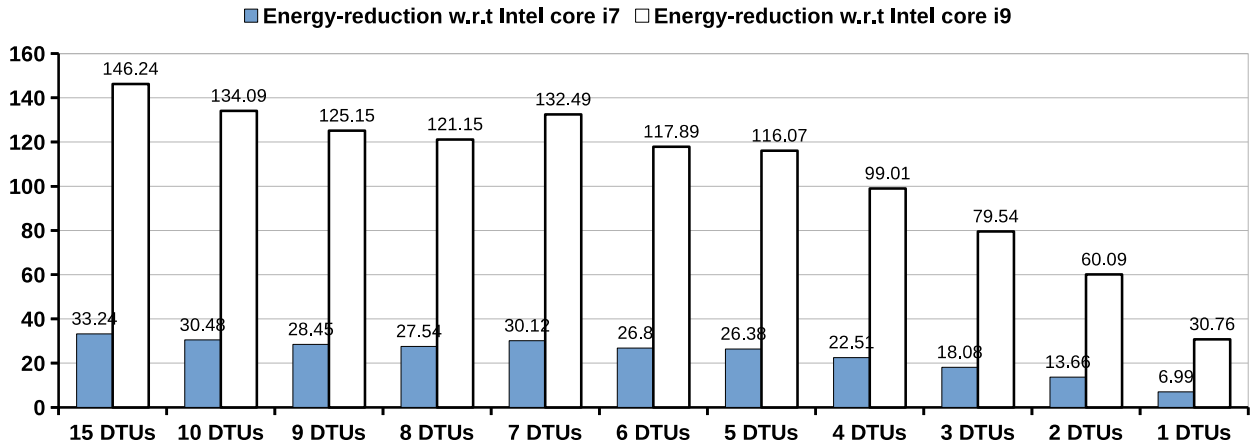

Fig. 5. Energy reduction of our system with respected to the Intel Core i7 and Intel Core i9 when processing the dataset.

VI. Conclusions

This paper presents a scalable and efficient architecture designed to accelerate random forest on FPGA-based edge computing platforms. The proposed architecture includes the Decision Tree Units (DTUs), where the pipeline technique is applied to enhance performance. To accommodate the decision tree's parameters effectively, we introduce the memory structure of the DTUs, which comprises five subsets for efficient pipeline processing. The architecture is implemented using SystemVerilog, enabling the scalability of the system by adjusting the number of DTUs. A prototype system is developed on a Xilinx Zynq device as an edge computing platform.

In the conducted experiments, the proposed system is com- pared with Intel Core i7 (desktop version) and Core i9 (high- performance computing version) processors. The experimental results demonstrate significant improvements achieved by our system. With a certified

dataset, our system outperforms the Core i7 and Core i9 processors by 19.96× and 12.01× in terms of execution time, respectively. Furthermore, our system exhibits energy savings of 146.24× and 33.24× compared to the Core i7 and Core i9 processors, respectively. These results validate the scalability and efficiency of our system, confirming its suitability for edge computing applications.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

C.P.-Q. proposed the idea and wrote the paper; T.P.-D. implemented and evaluated the system; B.K.-D.-N. tested the system and proofread the paper. All authors had approved the final version.

ACKNOWLEDGMENT

We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

REFERENCES

[1] P. Yang, Y. H. Yang, B. B. Zhou, and A. Y. Zomaya, "A review of ensemble methods in bioinformatics," *Current Bioinformatics*, vol. 5, no. 4, pp. 296–308, 2010.

[2] A. M. Prasad, L. R. Iverson, and A. Liaw, "Newer classification and regression tree techniques: Bagging and random forests for ecological prediction," *Ecosystems*, vol. 9, no. 2, pp. 181–199, 2006.

[3] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog *et al.*: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[4] S. Biookaghazadeh, M. Zhao, and F. Ren, "Are FPGAs suitable for edge computing?" in *Proc. USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, Boston, MA: USENIX Association, Jul. 2018.

[5] R. Chen, T. Wu, Y. Zheng, and M. Ling, "MLOF: Machine learning accelerators for the low-cost FPGA platforms," *Applied Sciences*, vol. 12, no. 1, 2022.

[6] L. Crockett, D. Northcote, and C. Ramsay, *Exploring Zynq MPSoC: with PYNQ and Machine Learning Applications*, Strathclyde Academic Media, 2019.

[7] C. Pham-Quoc, J. Heisswolf, S. Werner, Z. Al-Ars, J. Becker, and K. Bertels, "Hybrid interconnect design for heterogeneous hardware accelerators," in *Proc. 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2013, pp. 843–846.

[8] T. K. Ho, "Random decision forests," in *Proc. the 3rd International Conference on Document Analysis and Recognition*, 1995, vol. 1, pp. 278–282.

[9] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] X. Lin, R. S. Blanton, and D. E. Thomas, "Random forest architectures on FPGA for multiple applications," in *Proc. the Symposium on VLSI 2017, ser. GLSVLSI '17*, New York, NY, USA: Association for Computing Machinery, 2017, pp. 415–418.

[11] S. Zhao, Y. Sun, and S. Chen, "A discretization method for floating-point number in FPGA-based decision tree accelerator," in *Proc. the 2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, 2018, pp. 2698–2703.

[12] A. Damiani, E. D. Sozzo, and M. D. Santambrogio, "Large forests and where to 'partially' fit them," in *Proc. the 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022, pp. 550–555.

[13] A. Jinguji, S. Sato, and H. Nakahara, "An FPGA realization of a random forest with k-means clustering using a high-level synthesis design," *IEICE Transactions on Information and Systems*, vol. E101.D, pp. 354–362, 2018.

[14] T. Ikeda, K. Sakurada, A. Nakamura, M. Motomura, and S. Takamaeda-Yamazaki, "Hardware/algorithm co-optimization for fully-parallelized compact decision tree ensembles on FPGAs," in *Proc. the International Symposium on Applied Reconfigurable Computing, ARC 2020*, 2020, pp. 345–357.

[15] Y. R. Qu and V. K. Prasanna, "Scalable and dynamically updatable lookup engine for decision-trees on FPGA," in *Proc. the 2014 IEEE High Performance Extreme Computing Conference (HPEC)*, 2014, pp. 1–6.

[16] A. Alcolea and J. Resano, "FPGA accelerator for gradient boosting decision trees," *Electronics*, vol. 10, no. 3, 2021.

[17] J. Oberg, K. Eguro, R. Bittner, and A. Forin, "Random decision tree body part recognition using FPGAs," in *Proc. the 22nd International Conference on Field Programmable Logic and Applications (FPL)*, 2012, pp. 330–337.

[18] PYNQ: Python productivity. [Online]. Available: http://www.pynq.io/

[19] Kaggle. California housing prices-median house prices for California districts derived from the 1990 census. [Online]. Available: https://www.kaggle.com/datasets/camnugent/california-housing-prices

[20] Scikit learn. Scikit-learn California housing dataset. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetchcaliforniahousing.html

[21] AMD Xilinx. Vivado overview. [Online]. Available: https://www.xilinx.com/products/design-tools/vivado.html