An Application-Specific Field-Programmable Tree Ensemble Architecture

Jan Kuehn and Yiannos Manoli
Department of Microsystems Engineering - IMTEK, University of Freiburg, Germany
{jan.kuehn, manoli}@imtek.uni-freiburg.de

***Abstract*—Supervised machine learning for data classification is increasingly implemented in hardware to be integrated close to the source of the data. The ability to update a trained machine learning model is the most important property any classification system must fulfill. This is often achieved by implementing the algorithm on reconfigurable hardware but some applications require speed, size, or power efficiency only application-specific integrated circuits (ASICs) can offer. Architectures that have proven to be very efficient on reconfigurable hardware are not always suited for custom ASIC designs. We therefore propose to integrate commonly used field-programmable technology in an application-specific architecture to allow updates of the trained model. This design pattern allows deep integration into full-custom ASICs while leveraging all advantages of reconfigurable hardware.**

I. Introduction

In the era of wireless sensor networks and pervasive computing, energy and speed requirements force sensor system designers to implement signal processing more efficiently and closer to the sensor. Data classification is acknowledged as a vital part of the signal processing, leading to an increased use of machine learning algorithms. Their ability to be trained for a variety of applications makes them extremely versatile but also calls for flexible implementations that can adapt to newly trained models. This flexibility is commonly achieved with reconfigurable hardware like FPGAs. However, in some applications it might be desired to implement machine learning directly on the sensor fabric in an application-specific integrated circuit (ASIC).

In this work, we focus on the implementation of decision tree ensembles like the random forest algorithm, originally proposed in [1]. Tree ensembles are popular because of their robustness and are used in a variety of applications reaching from network traffic analysis [2] to sensor signal processing [3]. They can be used for both classification and regression and are especially interesting for size-efficient implementations due to the simplicity of the underlying mathematical function.

The next section gives an overview of published tree ensemble architectures and discusses design considerations for the reconfigurable architecture we present in Section III.

II. Related Work

Published architectures for tree ensembles can be divided into two groups. Von Neumann-like architectures serially traverse the decision trees based on a trained model stored in the memory. Programming is easy and common techniques like pipelining and branch prediction can be applied. Parallel architectures compute all nodes at once and encode the tree structure and leaf outputs in a boolean function, essentially storing parts of the trained model in the circuitry.

A. FPGA Implementations

The serial architecture presented in [4] extensively uses pipelining to optimize throughput. Interestingly, the switching and interconnection blocks of the FPGA limit this design, not its logic resources. The reason is that the pipeline must support the intrinsic flexibility of the serial architecture which requires comprehensive multiplexing and wiring.

The authors of [5] propose to translate trained decision trees to a VHDL representation of the parallel architecture. This is combined with a novel tree aggregation unit in [2] to implement tree ensembles. It is important to notice, that the generated VHDL code describes a static implementation of a specific trained model. Synthesizing this for an ASIC would exclude the support of model updates. On an FPGA however, the architecture is size efficient and provides high throughput.

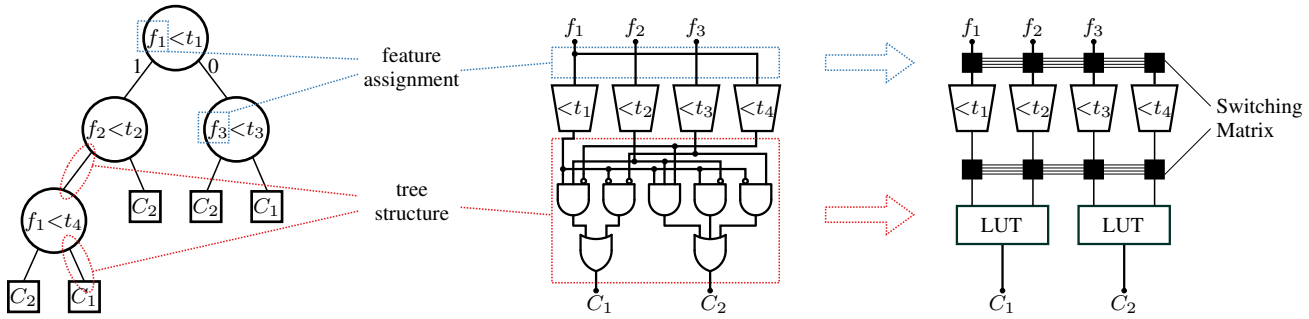
Intuitively, we expect serial architectures to be more size efficient than parallel ones. However, a size comparison of [4] and [2] based on a size prediction function proposed by [2] rules in favor of the parallel architecture. The serial architecture is inherently flexible and does not leverage reconfigurability. Thus, the pipeline structure becomes very complex making the architecture less size efficient. The parallel architecture on the other hand makes better use of the reconfigurable fabric since it is static and encodes parts of the trained model in the circuitry.

We strongly believe that this way of storing information is the main asset of the static, parallel architecture. At the same time, this is the main weakness that prohibits us from implementing the architecture on an ASIC. The information stored in the circuitry would have to be modified when the tree ensemble is retrained.

B. ASIC Implementations

A partly configurable decision tree is implemented on an acoustic sensor ASIC in [3]. The thresholds and feature selection for each node can be configured. The structure of the tree, however, is fixed and rather small making this a more restrictive implementation.

The first and only ASIC implementation of a tree ensemble accelerator was introduced in [6]. It is based on mixed



(a) An example classification tree, (b) its corresponding static implementation, (c) and the reconfigurable architecture

Fig. 1. Comparators implement the tree nodes and are wired to the corresponding feature inputs by fixed or configurable interconnections. The boolean function encodes the tree structure and can be substituted by look-up-tables (LUTs) in the reconfigurable architecture.

signal in-memory computations using the readout amplifier to perform the comparisons. Although several trees are computed in parallel, the tree itself is traversed serially making this a Von Neumann architecture.

III. A Reconfigurable Architecture

Our goal is to combine application-specific and field-programmable technology to implement the parallel architecture on an ASIC in a way that supports model updates. The hardware will be designed specifically for tree ensembles and optimized for a particular application to gain maximum efficiency. This will leave us with an architecture that combines the efficiency and speed of ASICs with the flexibility of reconfigurable logic.

To reach this goal, we first have to understand which parts of the architecture must be reconfigurable and which parts can be static. When designing a tree ensemble for a specific application, hyperparameters like the number of trees, features, and classes usually stay fixed. Thus, parts of the architecture that relate to hyperparameters do not need to be reconfigurable.

Fig. 1a shows an example of a single classification tree after training. The feature ids f_i and threshold values t_i of the nodes, the class labels C_i of the leafs, and the general tree structure will change with every training. Fig. 1b illustrates how these parameters are statically implemented in parts of the parallel architecture. In an ASIC implementation these parts must remain reconfigurable.

We propose to substitute them by reconfigurable counterparts commonly used in FPGAs as depicted in Fig. 1c. The assignment of features to the nodes is achieved by a switching and interconnection network that wires the feature inputs to the appropriate node comparator. The boolean function that encodes the tree structure and computes the leaf outputs is replaced by look-up-tables (LUTs) as used in FPGAs logic blocks. Only the comparators stay fixed and their thresholds must be stored in registers. This allows the architecture to support model updates.

Of course, the number of nodes in a tree might change after training, although the tree growth is usually limited. Providing each tree unit with a sufficient number of comparators would solve this issue. However, here we should expand our focus from a single tree to the complete ensemble. The switching and interconnection blocks can span the ensemble and allow comparators to be shared between trees. We expect this to lead to a more size efficient architecture, considering that individual trees in the ensemble can have very different sizes.

In the same manner, the output function LUTs could be shared to change the number of trees or classes. Other enhancements are possible, often creating a trade-off between size and flexibility. Eventually, this will lead to field-programmable array of tree building blocks that can be configured to efficiently implement a range of tree ensembles.

IV. Conclusion

Based on our assessment of available tree ensemble architectures, we propose a design scheme that enables the ASIC implementation of a parallel architecture that is inherently restricted to reconfigurable devices. Doing so, we strive to answer the following questions:

- Is the parallel architecture still as efficient when we consider the reconfiguration overhead?
- Will the reconfiguration framework require less memory than a Von Neumann architecture to store the trained model inside the architectures circuitry?
- Can this design pattern be applied to other algorithm families as well?

References

[1] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[2] M. Barbareschi, S. Del Prete, F. Gargiulo, A. Mazzeo, and C. Sansone, *Decision Tree-Based Multiple Classifier Systems: An FPGA Perspective*. Cham: Springer International Publishing, 2015, pp. 194–205.

[3] K. Badami, S. Lauwereins, W. Meert, and M. Verhelst, "24.2 context-aware hierarchical information-sensing in a 6μW 90nm cmos voice activity detector," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, 2015, pp. 430–431.

[4] B. V. Essen, C. Macaraeg, M. Gokhale, and R. Prenger, "Accelerating a random forest classifier: Multi-core, GP-GPU, or FPGA?" in *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, 2012.

[5] F. Amato, M. Barbareschi, V. Casola, A. Mazzeo, and S. Romano, "Towards automatic generation of hardware classifiers," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2013, pp. 125–132.

[6] M. Kang, S. K. Gonugondla, and N. R. Shanbhag, "A 19.4 nj/decision 364k decisions/s in-memory random forest classifier in 6t sram array," in *ESSCIRC Conference 2017: 43rd European Solid-State Circuits Conference*, 2017, pp. 263–266.