Hardware acceleration of encrypted TCP / IP traffic classification using Machine Learning techniques

Samuel Picallo Martínez, Pedro Pérez Carballo, Antonio Núñez Ordóñez and Sonia Raquel León Martín
IUMA, Institute for Applied Microelectronics, University of Las Palmas de Gran Canaria, Spain
{spicallo,carballo,nunez,slmartin}@iuma.ulpgc.es

***Abstract*—Encryption methods adopted in TCP/IP based communications represent a challenge to traffic classification. This scenario has encouraged an evolution from classic techniques towards statistical flow-based approaches. In this work it's proposed a solution based on a C5.0 Machine Learning Algorithm, as well as a hardware acceleration for its prediction stage. The accelerator is composed of parallel processing nodes capable of processing data from a streaming source in a pipelined fashion to use resources more efficiently and increase the achievable throughput. The results show that this system achieves a throughput 3 orders of magnitude higher than original C5.0 prediction software implementation.**

Keywords- Encrypted traffic classification; Machine Learning; C5.0; Hardware acceleration; FPGA

I. INTRODUCTION

Network traffic classification is an important but challenging task for network management and security. One of its main obstacles has been the proliferation of encryption systems in the different layers of the TCP / IP model. This has made ineffective the usual classic TCP / IP traffic classification techniques [1], [2]. One of the techniques widely used is DPI, based on the analysis of the payload of individual packets. This information is usually contained in the layer to be encrypted and has produced a change in the strategies used for traffic classification. In order to solve this problem, ML (Machine Learning) based techniques have emerged. These rely on flow statistical features.

One of the main ML algorithms employed for network classification tasks is C5.0. This decision tree type algorithm has shown good performance when facing encrypted traffic [3].

The application of ML in networks substantially increase the need for computation, in order to maintain the high transmission rates achieved in current networks without affecting transmission latencies. Therefore, it is practically mandatory to apply hardware acceleration techniques. MPSoC FPGA based platforms applied to network classification purposes have demonstrated to achieve significant performance gains in terms of time delay and bandwidth [4]. FPGA architecture possibilities suite well for decision tree implementations, achieving an outperform when compared with multicore platforms [5].

C5.0 algorithm-based classification is performed in two stages. The first stage generates a decision tree from a previously classified (ground truth) data set. The second stage consists of doing the predictions using the previous generated decision tree. Prediction stage determines the classification time, so it is of interest its implementation into a MPSoC FPGA platform in order to minimize the total classification time.

II. METHODOLOGY

This section explains in detail the design flow used in this work, represented in Figure 1.

A. Generating the Classification Model

In order to generate a reference decision tree to implement, we used a dataset including encrypted network flows from [6]. First, a filter is applied to obtain a version which contains the instances of the seven applications with more presence and 76 flow features of each instance, leaving out features that are prone to be out of date rapidly such as client port and server IP.

Using C5.0 R environment, an experiment based on the genetic algorithm [7] is designed to analyze the potential of the algorithm for the dataset. Then, a sequential forward selection is performed in order to identify the features that provide more information at the time of classifying. At this step, 4-fold cross validation is used to evaluate the results of each combination. As the classes have a great imbalance, the Cohen's kappa value [8] is used as the quality metric. Once the target features have been identified, it is performed a grid search in order to achieve a fine adjustment of the algorithm setting parameters. The reference classification model is generated using all instances present in the data set and the optimal configuration resulting from the experiments proposed.

B. Hardware Acceleration

Once the reference decision tree has been generated, an initial Python model is designed in an agile manner. In order to achieve a synthesizable model, it is translated into its C++ equivalent model. At the hardware design stage, High-Level Design Methodology [9] was employed. Compared with traditional RTL methodology, more time is spent at higher levels of abstraction, where verification times are the fastest and productivity gains are the greatest. Xilinx Vivado HLS tool [10] allows to develop and verify decision tree algorithm at the C-level and then synthesizes it into its RTL equivalent model. At this stage, a developed Python script is used to parse and gather the attributes of each of the nodes present in the decision tree.

Accelerator performance is optimized using Vivado HLS dedicated pragmas. The correct operation of the RTL model is checked in an initial verification using the C / RTL co-simulation functionality of the tool. The next step is the integration of the system conforming a platform, including data movers between ARM-based Processing System (PS) and custom hardware accelerator (PL). As result of the integration process the programming bitstream and the hardware description file are generated.

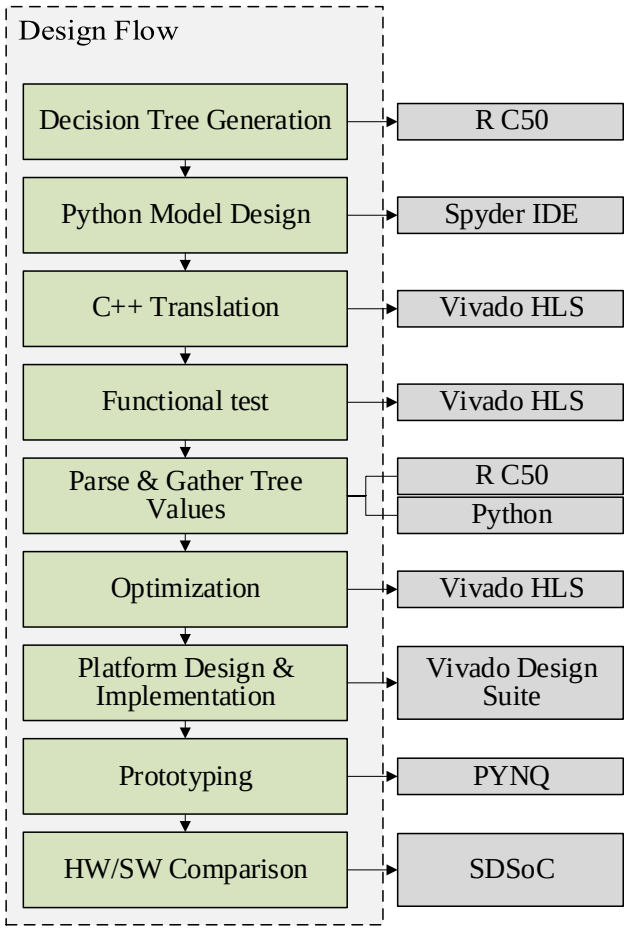

Figure 1. Design Flow.

Finally, the system is verified on a prototype using Xilinx PYNQ methodology. The embedded software is developed in Python. Also, we have used some Integrated Logic Analyzer cores for debugging purpose.

Hardware accelerator performance is compared with its equivalent software executed in ARM-based target device processing system. SDSoC dedicated libraries are used to measure software execution time using device's independent cycle counter.

III. RESULTS

A. Classification Model

The feature reduction experiment identified a set of six features that achieve even higher quality models than those generated with all of the features present in the dataset. Besides, it manages to reduce generation time and complexity in terms of the number of nodes in the decision tree. This emphasizes the importance of taking into account the appropriate flow features to achieve an effective and efficient model [11].

On the other hand, tuning of the algorithm settings shows a trade-off between complexity and quality. In the analyzed range, reducing the number of nodes in about half penalizes the Cohen's kappa metric by 1.3 points. This is especially important when performing hardware implementation, since a smaller number of nodes will increase accelerator performance and decrease its resource utilization.

In the 4-fold cross validation evaluation, models with a number of nodes between 26047 and 55189 obtain a kappa value in the range between 70.6% and 71.9%. The 26047 nodes model is chosen because it presents the best relationship between quality and complexity. Table I shows the precision and recall values of the chosen reference model. It is observed that recall values are especially penalized for those classes that have fewer instances.

TABLE I. PRECISION AND RECALL VALUES FOR EACH OF THE SEVEN CLASSES OF THE RESPONSE VARIABLE IN THE EVALUATION OF THE C5.0 MODEL WITH REDUCTION OF PREDICTOR VARIABLES AND FINE ADJUSTMENT OF ALGORITHM SETTING PARAMETERS.

Metric	**Class**						
	Amazon	***Facebook***	***Google***	***Microsoft***	***Skype***	***Windows Update***	***Youtube***
Precision	83.75%	91.94%	87.63%	79.77%	79.39%	90.91%	83.25%
Recall	75.20%	88.29%	96.72%	63.40%	56.54%	86.86%	50.18%

B. Hardware Acceleration

Designed hardware accelerator is capable of processing data from a streaming source in a pipelined fashion, achieving an initiation interval of one cycle between each data input. Thus, the rate of classifications per second is determined by the operating frequency of the block. Latency is committed to the product of the number of stages in the pipeline and the clock period. FPGA concurrency allows to reduce the classification latency by one order of magnitude and increase the rate of classifications per second by three orders of magnitude with respect to the original software application executed in the ARM-based processing system. As shown in Table II, smallest decision tree implementation on XC7Z020 device achieves a classification rate 1943.7 times higher for the hardware version compared to the best software version.

TABLE II. HW/SW EXECUTION TIME COMPARISON FOR SMALLEST DT ON XC7Z020 DEVICE.

	Hardware implementation		***Software implementation for ARM A9***		***Original software application executed in ARM A9***	
	Latency	**Throughput**	**Latency**	**Throughput**	**Latency**	**Throughput**
Frequency	160.3 MHz		666 MHz			
Cycles	182	1	8076	8076	8113	8113
Period	6.24 ns		1.50 ns		1.50 ns	
Results	1135 ns	160.3 MCPS	12114 ns	82.47 kCPS	12169 ns	82.09 kCPS

kCPS: Kilo Classifications Per Second – MCPS: Mega Classifications Per Second

For the implementation of the proposed block, three classification models of different complexity have been analyzed on three different technologies: Artix-7 (XC7Z020), Kintex-7 (XC7Z045) and UltraScale+ (XCZU9EG). It is

observed that the number of utilized memory blocks remains constant regardless of the device and the target frequency, with the number of nodes that form the tree determining the amount of BRAM resources needed (Figure 2).

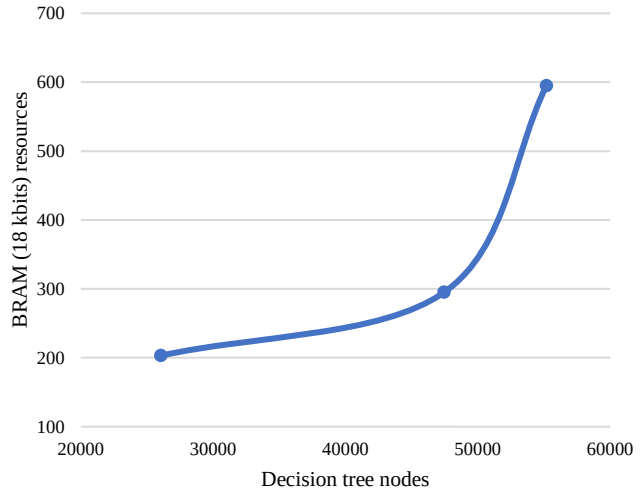

Figure 2. BRAM (18kbits) resources utilization in function of implemented decision tree complexity in terms of its number of nodes.

This is a critical factor, since the amount of available resources of this type on the target device will limit the complexity of the model to be implemented. Regarding the rate of classifications, it will be the technology of the target device that determines the limit of the frequency of operation, with a low impact of the complexity of the model on it, as shown in Figure 3 and Figure 4. Figure 5 shows platform implementation floorplan for FPGA device.

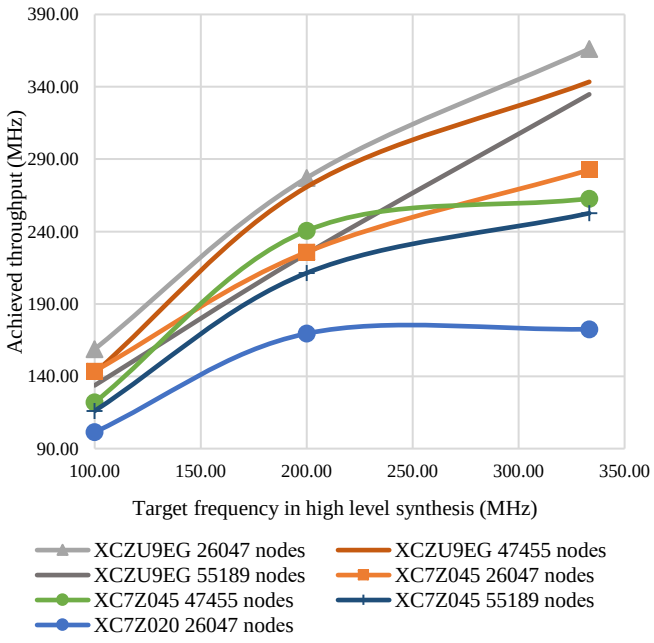

Figure 3. Achieved implemented throughput in function of target frequency at the high level synthesis stage.

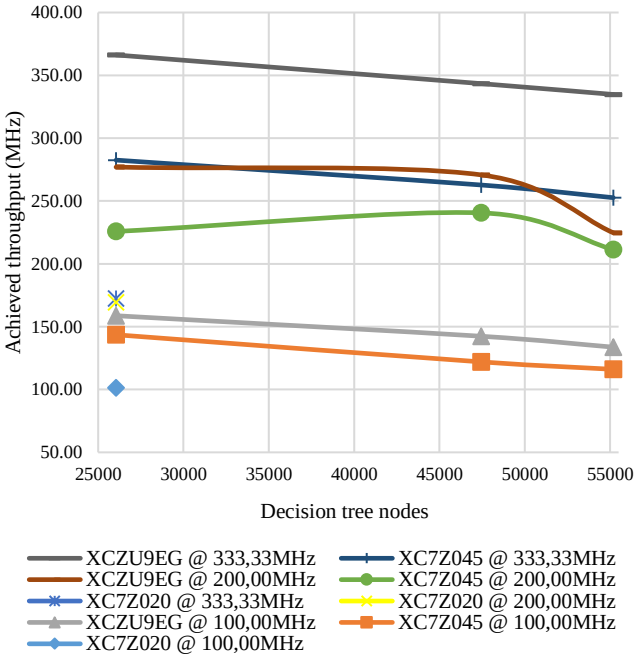

Figure 4. Achieved implemented throughput in function of.implemented decision tree complexity in terms of its number of nodes.

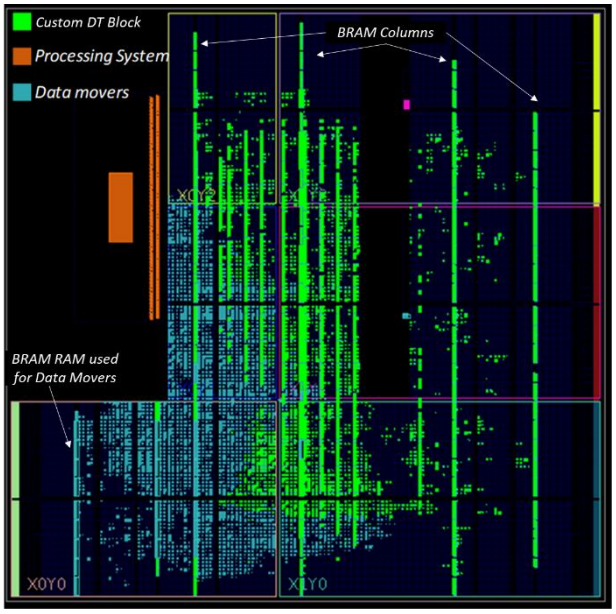

Figure 5. Viewing implementation placement of smallest DT on XC7Z020 @100MHz platform.

Proposed solution is compared with state-of-the-art decision tree hardware implementations attending to factors such as achieved frequency, decision tree nodes and FPGA technology. Figure of merit is calculated by equation (1), as shown in Table III. The main difference between this work and the analyzed ones lies in the greater complexity of the implemented tree, which makes it necessary to use BRAM memories instead of distributed ones, affecting the maximum frequency.

$$F = \log_{10}(Nodes)\frac{Achieved\ Frequency}{Technology\ coeficient} \quad (1)$$

TABLE III. STATE-OF-THE-ART DECISION TREE IMPLEMENTATION COMPARISON.

	Methodology	*Technology*	*Nodes*	*Achieved frequency (MHz)*	*F*
D. Tong et al. [5]	RTL	Virtex UltraScale	200	896.00	*981.77*
R. Kułaga & M. Gorgoń [12]	HLS	Artix-7	1958	88.00	*289.68*
J. R. Struharik [13]	RTL	Virtex-5	-	251.95	-
This work	HLS	Artix-7	26047	172.50	*761.72*
This work	HLS	Kintex-7	55189	252.65	*748.77*
This work	HLS	UltraScale+	55189	334.78	*755.94*

IV. CONCLUSIONS

C5.0 ML algorithm-based solution proposed in this work can classify encrypted network traffic, where flow parameters and number of instances of each class will determine the quality of the classification model. The design flow used allows the implementation to be carried out again in a highly automated way when the tree is updated. Regarding its hardware acceleration, the technology integrated in the target MPSoC FPGA device has a high impact on the maximum classification rate, achieving a throughput 3 orders of magnitude higher than original C5.0 prediction software.

REFERENCES

[1] B. Ma, H. Zhang, Y. Guo, Z. Liu, and Y. Zeng, "A Summary of Traffic Identification Method Depended on Machine Learning," in *2018 International Conference on Sensor Networks and Signal Processing (SNSP)*, 2018, pp. 469–474.

[2] P. Wang, X. Chen, F. Ye, and Z. Sun, "A Survey of Techniques for Mobile Service Encrypted Traffic Classification Using Deep Learning," *IEEE Access*, vol. 7, pp. 54024–54033, 2019.

[3] Z. Aouini, A. Kortebi, Y. Ghamri-Doudane, and I. L. L. Cherif, "Early classification of residential networks traffic using C5.0 machine learning algorithm," in *2018 Wireless Days (WD)*, 2018, pp. 46–53.

[4] A. Dominguez, P. P. Carballo, and A. Nunez, "Programmable SoC platform for deep packet inspection using enhanced Boyer-Moore algorithm," in *2017 12th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2017, pp. 1–8.

[5] D. Tong, Y. R. Qu, and V. K. Prasanna, "Accelerating Decision Tree Based Traffic Classification on FPGA and Multicore Platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 11, pp. 3046–3059, Nov. 2017.

[6] J. S. Rojas, Á. R. Gallón, and J. C. Corrales, "Personalized Service Degradation Policies on OTT Applications Based on the Consumption Behavior of Users," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10962 LNCS, 2018, pp. 543–557.

[7] L. Scrucca, "On Some Extensions to GA Package: Hybrid Optimisation, Parallelisation and Islands Evolution," *The R Journal*, vol. 9, no. 1, p. 187, 2017.

[8] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, Apr. 1960.

[9] Xilinx Inc., "UltraFast High Level Productivity Design Methodology Guide," 2019.

[10] Xilinx Inc., "Vivado Design Suite User Guide. High-Level Synthesis," 2019.

[11] H. Oudah, B. Ghita, and T. Bakhshi, "A Novel Features Set for Internet Traffic Classification using Burstiness," in *Proceedings of the 5th International Conference on Information Systems Security and Privacy*, 2019, pp. 397–404.

[12] R. Kułaga and M. Gorgoń, "FPGA Implementation of Decision Trees and Tree Ensembles for Character Recognition in Vivado Hls," *Image Processing & Communications*, vol. 19, no. 2–3, pp. 71–82, Sep. 2014.

[13] J. R. Struharik, "Implementing decision trees in hardware," *SISY 2011 - 9th International Symposium on Intelligent Systems and Informatics, Proceedings*, pp. 41–46, 2011.