Pipelined Decision Tree Classification Accelerator Implementation in FPGA (DT-CAIF)

Fareena Saqib, Aindrik Dutta, Jim Plusquellic, Philip Ortiz, and Marios S. Pattichis

Abstract—Decision tree classification (DTC) is a widely used technique in data mining algorithms known for its high accuracy in forecasting. As technology has progressed and available storage capacity in modern computers increased, the amount of data available to be processed has also increased substantially, resulting in much slower induction and classification times. Many parallel implementations of DTC algorithms have already addressed the issues of reliability and accuracy in the induction process. In the classification process, larger amounts of data require proportionately more execution time, thus hindering the performance of legacy systems. We have devised a pipelined architecture for the implementation of axis parallel binary DTC that dramatically improves the execution time of the algorithm while consuming minimal resources in terms of area. Scalability is achieved when connected to a high-speed communication unit capable of performing data transfers at a rate similar to that of the DTC engine. We propose a hardware accelerated solution composed of parallel processing nodes capable of independently processing data from a streaming source. Each engine processes the data in a pipelined fashion to use resources more efficiently and increase the achievable throughput. The results show that this system is 3.5 times faster than the existing hardware implementation of classification.

Index Terms—Data mining, decision tree classification (DTC), hardware implementation, FPGA

1 Introduction

The process of converting unidentified or unprocessed data into actionable information that is important and valuable to the user is known as data mining [1]. Recent advances in technology and ever increasing demands for analyzing larger datasets have created abundant opportunities for algorithmic and architectural development and innovations. Hence data mining algorithms have become increasingly significant and complex. Similarly there is a great demand for faster execution of these algorithms, leading to efforts to improve execution time and resource utilization.

Decision Tree Classification (DTC) is a widely used classification technique in data mining algorithms. It has applications in daily life; for example, the detection of spam e-mail messages. It is also used in highly sophisticated fields of medicine and astronomy. Several diverse predictive models in classification algorithms including artificial neural networks [2], decision trees [3] and support vector machines [4] have also been previously described in the literature. A number of solutions have also been suggested for hardware implementation by various authors [5]-[7]. Decision tree classification techniques categorizes each data records/tuples, having set of attributes/properties into subgroups or classes. Assigning of a category or class to each input dataset consists of a two-step process in DTC.

• *The authors are with the Electrical and Computer Engineering (ECE) Department, University of New Mexico, Albuquerque, NM 87131. E-mail: fareenasaqib1@gmail.com; {aindrik, jimp, pgortiz, pattichis}@unm.edu.*

Manuscript received 17 Oct. 2012; revised 22 Sep. 2013; accepted 07 Oct. 2013. Date of publication 17 Oct. 2013; date of current version 12 Dec. 2014. Recommended for acceptance by R. Gupta.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2013.204

The initial step is induction which involves construction of the decision tree model, where internal nodes and leaves constitute a decision tree model. Each internal node has a characteristic splitting decision and splitting attribute, while the leaves have particular category classification. Construction of a decision tree model from a training dataset/tuple constitutes of two phases. A splitting attribute and a split index are chosen by the model during the first phase. While during the second phase sorting of the tuples among the child nodes is performed based on the decision made in the first phase. This repetitive process is continued till the depth of the tree reaches a desired level. At this point, the decision tree can be used to predict the class of an input tuple which has not been classified yet.

The second step is the classification that includes application of the decision tree model to the test dataset to predict its respective class. The primary goal of such a classification algorithm is to utilize the given training dataset to construct a model which subsequently can be used to sort unclassified datasets into one of the defined classes [8]. Breiman et al. [9] presented decision trees approximately two decades ago, and described the decision trees as rooted tree structures, with leaves representing classifications and nodes representing tests of features that lead to those classifications. The accuracy of decision trees has been shown to be better or comparable to other models including artificial neural networks, statistical, and genetic models. The prediction in the classification process commences at the root, and a path to a leaf is followed by using the decision rules governed at each internal node. The characteristic class label to the leaf is then assigned to the incoming tuple.

DTC continues to function at high accuracy even in analysis of large data sets. Current technology advancements in data extraction and storage permit large amount of historic data to be preserved and utilized for data analysis and creation of more realistic classification rules. The property of DTC to function at high accuracy even when handling in large data sets makes it an appealing tool.

Decision trees have since been implemented in software programs. Although the software implementation of DTC is highly accurate the execution times and the resource utilization still require improvement to meet the computational demands in the ever growing industry. Whereas hardware implementation of Decision trees has not been investigated or reported in detail. Only a few researchers [10]-[12] proposed hardware realization of various decision trees using different architectures for specific problems.

This paper focuses on the speedup of the classification step using hardware acceleration. We propose a pipelined architecture for the hardware implementation of axis-parallel binary decision tree classification that meets the current demands of increased throughput with minimal resource utilization. The proposed design supports a streaming architecture by using double-buffered input and output memories to simultaneously receive and process data. Our experiments prove that our proposed hardware acceleration of classification algorithms increases throughput by reducing the number of clock cycles required to process the data and generate results. The architecture also requires minimal resources and is therefore area efficient. For scalability this proposed architecture, when configured with a high speed communication unit, enables processing and data transfer simultaneously. As long as the performance of the decision tree classification engine meets or exceeds that of the communication unit, processing time is not affected by the transfer of data.

We developed the decision tree classification algorithm in detail and explored techniques for adapting it to a hardware implementation successfully.

2 Background

A number of hardware implementations of decision tree examples are reported in the literature [10], [11]. The approach of using single

0018-9340 © 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

level classification technique instead of staged or multi-level technique limits the throughput because of having a restraint in the design that new instance cannot be applied to the input before completion of the classification of the previous data instance, resulting in low throughput. On the other hand the staged/leveled technique allows a new instruction/data fetch every clock cycle and thus optimizes the throughput.

A more advanced approach, proposed by [10] is based on the equivalence between decision trees and threshold networks hence resulting in fast throughput since the signals have to propagate through two levels only, irrespective of the depth of the original decision tree. Most of the architectures for hardware implementation of decision trees mentioned in the literature require a considerable number of hardware resources [12].

Past research work has been reported on hardware implementations of data mining algorithms. Baker and Prasanna [13] used FPGAs to implement and accelerate the Apriori [14] algorithm, a popular association rule mining technique. They developed scalable systolic array architecture to efficiently carry out the set operations, and used a "systolic injection" method for efficiently reporting unpredicted results to a controller. In [15], the same authors used a bitmapped CAM architecture implementation on an FPGA platform to achieve significant speedups over software implementations of the Apriori algorithm. Several software implementations of DTC have been proposed [16], [17], which used complex data structures for efficient implementation of the splitting and redistribution process. These implementations focused on parallelizing DTC using coarse-grain parallelization paradigms.

Li and Bermak [18] suggested a decision tree classifier based on an axis-parallel decision tree. Bachir et al. [19] presented both a hardware-dedicated decision tree technique for the generation of exponential variates and a derived architecture implemented in FPGA.

Podgorelec and Kokol [20] proposed a self-adapting evolutionary algorithm for the induction of decision trees and described the principle of decision making based on multiple evolutionary induced decision trees-decision forest. Chrysos et al. [21] presented data mining on the web for classifying and mining huge amounts of e-data by an implementation of data mining algorithm on a modern FPGA to accelerate certain very CPU intensive data-mining/data classification schemes. Subsequently they exploited parallelism at the decision variable level and evaluated its implementation on a modern high- performance reconfigurable platform [22].

The objective of this paper was to find an architecture that could ensure high throughput with significant reduction in hardware complexity. Generally, with an increase in the data sizes, the running time stretches to several hours. In the architecture designed for this research, each data record is assigned to a class using the predefined classification rules. The developed solution yielded high accuracy while handling large datasets. The hardware implementation for this study helped enhance the performance over software implementations.

3 Decision Tree Classification Architecture

In this paper we propose an efficient pipeline based implementation of a decision tree classification algorithm. The hardware accelerator for decision tree classification performs parallel operations using concurrent engines, where each engine implements pipeline technique and thus fetches data records in every cycle, enhancing the performance of classification process.

In our solution we proposed and adopted a two phased decision tree classification process. Firstly in the induction Phase a training dataset is used in order to determine the rules, based on which the classification is to be done, at each node. We have opted to provide these induced decision rules from the Microblaze softcore microprocessor to the decision tree classification engine. In the next phase, the classification is performed at the hardware level. The induction process will be investigated in future work. Our proposed architecture employs a pipelined data path, where the data is distributed in a pipeline to execute concurrently, which is of significant importance for large datasets to reduce the clock cycles.

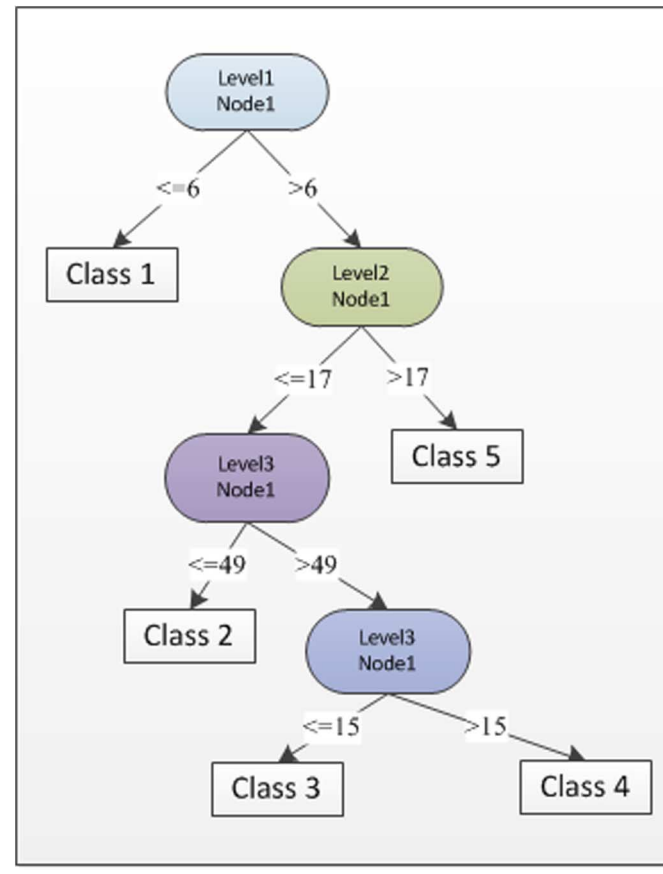

Fig. 1. Decision rules in form of decision tree.

The decision tree classification engine architecture concentrates on axis-parallel binary trees, where each node in the tree can have no more than two child nodes and only one of the attributes comprising the dataset is compared against a constant at each node. These constants are determined in the induction phase for each node. Fig. 1 shows an example of binary decision tree, for a given dataset, where the leaf nodes represent the classes that divide the data into different categories, and each internal node represents the test conditions, from which it traverses and reaches one of the classifications.

The decision tree classification subsystem implements each level of tree using a stage as represented in Fig. 2. Each stage consists of a decision logic, coefficient memory and internal registers. The input address to the coefficient memory is a function of the path through the decision tree that was taken to arrive at that particular node. Each coefficient memory stores coefficient values, attribute index of the incoming data from which to compare the coefficient, operation to be performed and a pointer to either the memory location of the next stage or the class assigned. The output of the coefficient memory contains all the information needed to perform the operation associated with the node in the tree being addressed.

The decision tree classification engine has three major parts: a) the double-buffered input block RAM b) the decision tree classification subsystem, and c) the double-buffered output block RAM. The decision logic reads the incoming data and takes the rules from its associated coefficient memory, processes them and forwards the data to the next stage with the processed results. The intermediate results decide whether a category is assigned to the data or further

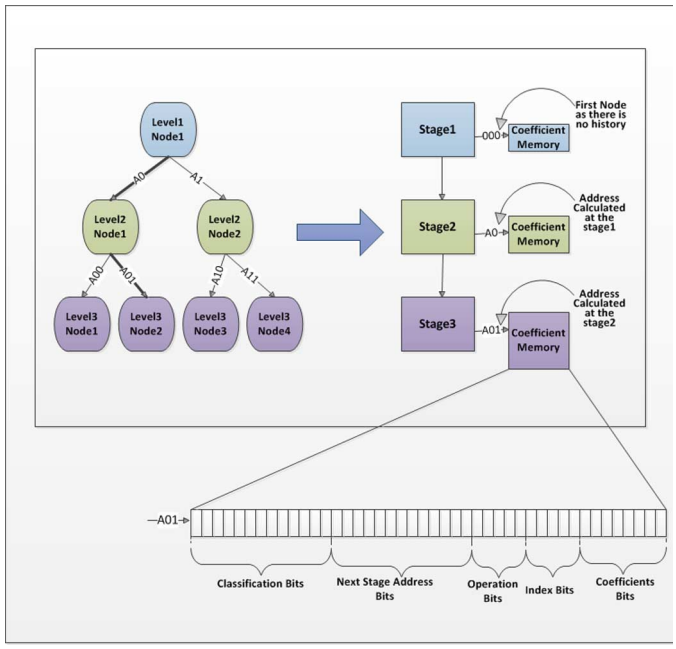

Fig. 2. Decision tree stages.

processing is required in the next stage. In case when the classification is complete for a data, the data is forwarded to the next stages without further processing, otherwise the processing and comparison is repeated until it is assigned to a class and then stored in the output memory. All these operations are performed in a pipelined manner where in every clock cycle the data is forwarded into next stage and newer data is fetched.

Fig. 3 represents a decision tree with depth of n, having n stages from which the data passes through, and then the classification is stored in the output block memory. The unclassified data is provided by the double-buffered input block RAM to the first stage of the engine, from where it is processed and propagated down the pipeline. The classifications for each tuple, are stored in the double-buffered output block RAM. The Xilinx Logicore IP Block Memory Generator has been used in order to implement the input and output memories. Where, block memory generator uses embedded block memory primitives in Xilinx FPGAs to implement memories of different depths and widths. Our proposed design implemented on Digilent Nexys2 Spartan 3 E board uses two fully independent ports each with its own read and write interfaces and access to a shared memory space. These ports can operate at different clock frequencies thus making it possible for the classification subsystem to operate at double the frequency of the on-board system clock.

Fig. 4 shows the RTL level block diagram of one such hardware module/stage of the classification subsystem. In each module there is a memory element, namely coefficient memory associated with it. These memory elements are also generated using the Xilinx Logicore

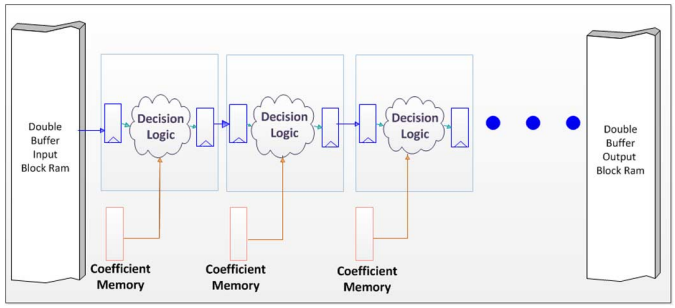

Fig. 3. Decision tree classification subsystem.

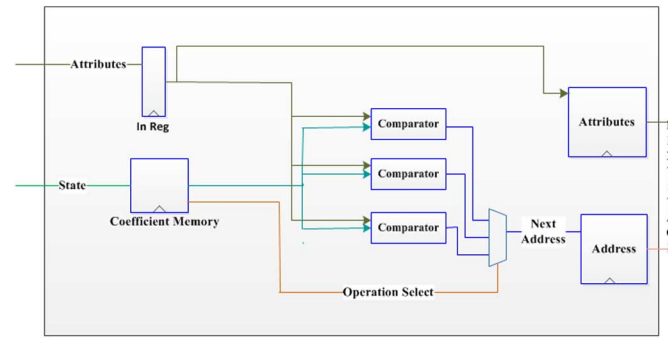

Fig. 4. RTL level block diagram of hardware module.

Distributed Memory Generator IP. During the memory configuration stage the RAM_access bit is set high. This allows the Microblaze to access the coefficient memory, in order to write the rules for each node associated to that level.

The control unit ties the address lines of the coefficient memory to the address value received from the previous hardware module in the pipeline. The size of the coefficient memory depends on which level of the tree it is associated. Hence the size varies from one 64 bit wide line to 2 n 64 bit wide lines where n is the number of levels in the decision tree.

The attributes are transferred to the module from the double-buffered input block RAM or the previous stage in the pipeline. Depending on the Attribute Index the attribute to be compared is selected and transferred to the comparators. The constant that it is to be compared against is fetched from the coefficient memory based on the address received from the previous stage. The new address for the coefficient memory of the next stage signifying the path to be taken (left child or right child) while traversing the decision tree is sent to the next stage in the pipeline based on the operation select lines and the comparator outputs.

The decision tree classification has been implemented as a Hardware-Software Co-Design. The Xilinx soft-core microprocessor Microblaze has been used to supply and fetch data to and from the reconfigurable decision tree classification engine. The data coming in is read by the Microblaze which sits on the Peripheral Local Bus (PLB). Microblaze in turn transfers the data to the double-buffered input block RAM of the decision tree classification engine. The engine is a custom peripheral designed as a slave module of the PLB. Once the double-buffered input RAM is written to with a given batch of data the Microblaze activates the classification engine by asserting a signal. The classified data is written into the double-buffered output block RAM.

In order to increase the efficiency of the engine it has been made parallel. Fig. 5 shows the overall pipelined and parallel architecture where the decision tree subsystem is instantiated eight times thus facilitating computation of eight classification result every clock cycle. After the initial latency, equivalent to the number of levels in the tree, 8 tuples of the dataset are categorized every clock cycle. Our tested design of the proposed architecture allows a depth of up to 13 levels, therefore the maximum latency for this design is 13. The address management for writing to the double-buffered input block RAM and reading from the double-buffered output block RAM has been done in such a way that eight consecutive tuples can be read and classified in every clock cycle. The double-buffered input and output RAMs are designed to allow for simultaneous buffering and processing. The operations of each RAM are switched after the given batch of data records are processed by the classification subsystem.

In theoretical analysis we analyzed following characteristics and limitations of the hardware architectures designed previously.

i. Only single data record is fetched in every cycle, thus requiring more clock cycles.

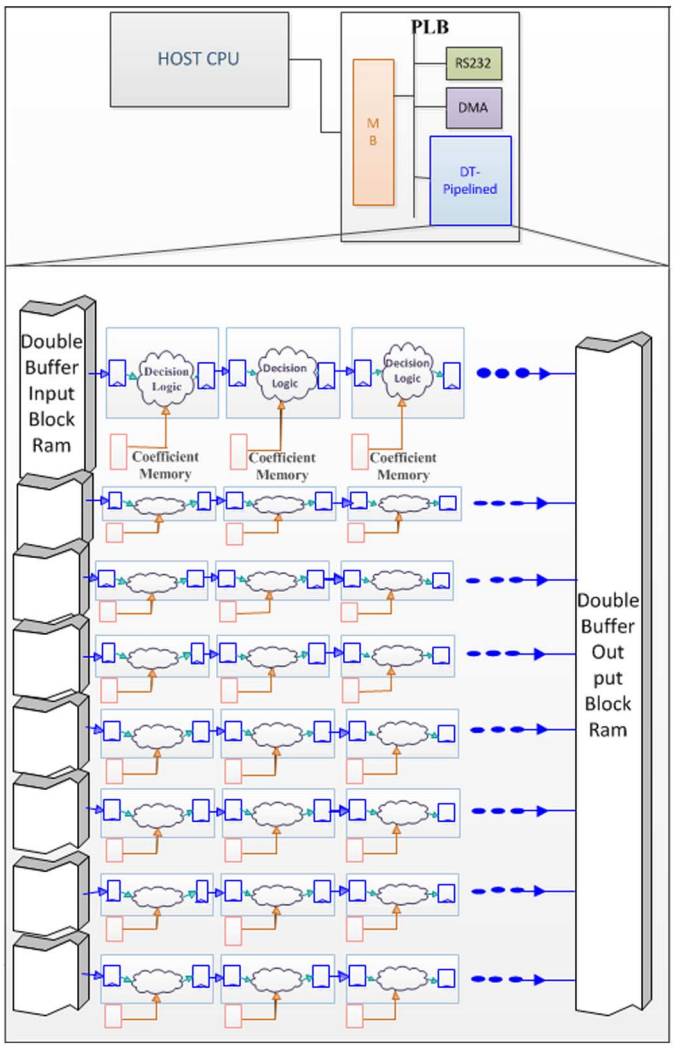

Fig. 5. Parallel and pipelined decision tree engine.

ii. Data record is fetched in sequential order from single input memory.
iii. The engine performs the classification and stores in output memory and only then fetches the new data record. Thus wasting the processing cycles.

Following are the enhancements in our proposed architecture where we utilize the hardware pipelines and parallelism to overcome the above mentioned limitations:

i. Engine is made of pipelined stages, each stage implements rules of one level of the tree.
ii. Pipeline to make use of processing cycles when data is written in memory, thus to increase the performance.
iii. Engine works on clock frequency double to that of the interface clock.
iv. Multiple data records are read as well as written simultaneously in every cycle, exhibiting parallelism, thus reducing the overall clock cycles.
v. Distributed memories are used for the coefficient lookup tables inside the peripheral for making the engine memory efficient, and to reduce the clock cycles to access the data.
vi. The block RAMs are placed in the peripheral such that the bus is not used in the memory accesses, thus reducing the clock cycles required for setting-up bus protocol.
vii. Also, the on-chip block memories are used for the pre-processed datasets, the classification rules and storing the classification results.

Consequently, we are able to optimize the access to memories in one clock cycle, in the given architecture. This results in overall reduction of clock cycles and hence a greater impact on the performance.

The development board used for this work is the Diligent Nexys-2 Spartan-3 E FPGA Board featuring a single Xilinx XC3S1200 E-FG320 FPGA. This particular component does not support PCI Express and without access to a high-performance interface, the proof-of- concept design discussed in this paper is implemented using RS232 to move data back and forth from the host to the FPGA. As the bandwidth of an RS232 link is inappropriate for an application requiring high performance, the I/O transfer time in the performance results as their inclusion would have completely hidden the performance increases realized by our parallel architecture.

The theoretical performance of the Gen-2 PCIe hard core in the Virtex-6 FPGA is 500 MB/s/lane, giving an x8 design a raw bandwidth of 4 GB/s in each direction. Assuming to achieve 80% efficiency due to bursting and DMA, this would be equivalent to transferring 3.2 GB/s, or 800 Mwords/s in each direction. Our design, for 4 attributes processes eight 32-bit samples in parallel at 100 MHz, the raw bandwidth of our logic is also 800 Mwords/s. Therefore, if we replace the RS232 interface with a PCIe interface, the I/O bandwidth would, at a first- order estimate, match that of our parallel implementation. For this reason, it is reasonable at this stage to include only the performance results for the parallel implementation and ignore the transfer time represented by our legacy RS232 interface, as a modern interface such as PCI Express would be able to keep up with our design's classification rate. The FPGA Implementation and experimental results are discussed in the next section.

4 Experimental Results

We have implemented the proposed architecture on Digilent Nexys2 Spartan 3 E FPGA board to perform classification in hardware accelerator. Variety of datasets, varying from benchmark to synthetic datasets have been used. The Number of tuples also varies to verify and validate the performance dependencies of the engine. Data pre-processing includes data cleansing, that is to normalize the data and conversion into hex-decimal number, to feed in the engine.

An open source tool WEKA [23], which is an open source tool under the GNU GPL license, was used for induction to establish the rules. For the induction, classification algorithm J48 was exploited for all the datasets used in the experiments conducted for the implementation. The rules were extracted from the binary decision tree generated through the induction. Further, the rules were formulated and provided to the micro-blaze for the classification process.

The Xilinx Platform Studio was used to program the micro-blaze; and to program hardware we used Xilinx ISE 12.4. Micro-blaze was provided with the rules of the classification; where with different datasets we have different classifications rules. With each test performed the data is fed into the memory. The speed of the clock is 50 MHz, whereas our proposed hardware accelerator operates on double clock frequency that is 100 MHz.

4.1 Accuracy of the Model

The accuracy of our parallel implementation of the pipelined architecture is shown in Table 1. Here Iris and Contact lenses from UCI machine learning repository [24] are the bench mark datasets, whereas synthetic datasets 1, 2 and 3, generated using Datagen [25]. A number of attributes varying from 4 to 6, are used with each configuration, having the number of tuples ranging from 100 to 1000. The results validate that our architecture supports varying number of attributes and tuples without deteriorating the accuracy of the model.

TABLE 1
The Accuracy of the Decision Tree Model

Dataset	Total number of instances	Correctly classified	%Correctly classified
Iris	150	147	98.00%
Contact Lenses	24	20	83.30%
Dataset1- 4 attributes	1000	1000	100.00%
Dataset2- 5 attributes	1000	1000	100.00%
Dataset3- 6 attributes	1000	1000	100.00%

4.2 Comparison with Software Implementations

For the comparison with the software, execution times of the decision tree classification engine is compared with WEKA data mining software, R-project and C implementation of classification process. In R-project the tree is implemented by recursive partitioning using Rpart routines and classification is performed using predict routine. The WEKA tool uses the ID3 for induction process, and performs classification on the test data. The same datasets were used for all the software and hardware implementations.

Detailed results of the study are shown in Table 2, presenting the time each implementation takes as well as the overall speedup/performance gains of hardware accelerator compared to software. The results show that the speed of C implementation is in microseconds and it takes less time than WEKA and R-project. WEKA, a java based tool, shows better performance than R-project. R-project is an interpreted language which is implemented in C, but in orders of magnitude slower than specialized C implementation of the classification.

We also tested our proposed system on datasets with 4, 5 and 6 attributes by varying the number of tuples from 100 to 8000 and it was established that there is no impact of the number of attributes on the performance of the engine. This occurred mainly due to the fact that

TABLE 2
Comparison with Software Implementations

No. of Tuples	Time for the hardware implementation			Time for the software implementation approx.			Speedup compared to C Implemantation
	No. of clock cycles	Latency	Total Time (ns)	Weka (ms)	R-project (ms)	C (us)	
100	7	4	220	0.5	1.33	12.3	55x
250	17	4	420	0.5	1.38	33.84	80x
500	32	4	720	1	1.52	76.92	106.83X
750	48	4	1040	1.5	1.66	112.3	107.9X
1000	63	4	1340	2	1.81	175.38	130X
2000	125	4	2580	2	2.42	286.45	111X
3000	188	4	3840	2	3.16	430.21	111X
4000	250	4	5080	5	3.87	570.23	112X
5000	313	4	6340	5	4.54	720.42	113X
6000	375	4	7580	6	5.4	860.3	113X
7000	438	4	8840	11	5.89	1006	113X
8000	500	4	10080	15	6.81	1154	114X

TABLE 3
Comparison with Hardware Implementations

Data set	DT	SMpL	Speedup	SMpL-P	Speedup
Glass	2.24	6.46	2.88x	6.96	3.12x
Balance-scale	1.5	4.97	3.31x	4.27	2.85x
Heart	1.26	4.99	3.96x	4.54	3.60x
Diabetes	1.48	6.72	4.54x	7.14	4.82x
ionosphere	1.88	6.46	3.43x	6.25	3.32x
Liver	1.74	6.37	3.66x	4.05	2.32x
Soner	1.92	6.55	3.41x	6.16	3.20x
Page block	1.29	6.93	5.37x	6.25	4.84x
Zoo	2.57	4.99	1.96x	5.88	2.28x

we have implemented an axis-parallel decision tree, the hardware takes the same number of cycles for classification regardless of the number of attributes of the dataset.

Our design is currently limited by the locally available memory and no high speed communication link to stream data, the maximum number of dataset tested is 8000 data records. If streaming data is available decision tree classification engine is designed to process at a fixed throughput that is linearly related to data set size. Theoretically the number of $\text{clock cycles} = (\text{data records}/8 + \text{latency}) + \text{clock cycle}$ for switching the buffered memory. For example for the dataset of 1 million records it will take 2.5 milliseconds.

4.3 Comparison with Previous Hardware Implementations

For the comparison with the previous hardware implementations, the clock cycles required by the FPGA implementation of decision tree classification engine are compared with the SMpL and SmpL-p implementation proposed by Struharik et al. [12]. The SMpL-p architecture employs one hardware module per level of the decision tree. In the experiments performed by Struharik et al. for the classification, 16 of the 23 datasets, used are binary trees. We have performed the experiments on the subset of the datasets used in the SmpL and SmpL-P, and compared the performance in terms of the clock cycles in Table 3 and it shows that decision tree classification engine has on average 3.5x speedup over these implementations.

4.4 Resource Utilization

Based on the RTL level hardware requirements SmpL and SmpL-P requires M.n multipliers, whereas our implementation requires 0 multipliers. Also the number of adders for our implementation is $2.M * 8$ adders, whereas the SMpL requires approximately M. [2 n] adders, where n is the number of attributes and M is the level of the trees. Hence the hardware requirements are also minimal for our proposed hardware engine.

The devised architecture area utilization, in terms of lookup tables and flip flops and the block RAM utilization is also optimized. Table 4 shows the utilization summary at different hierarchies of the design.

TABLE 4
The Resource Utilization of the Decision Tree Model

Implementation	LUTs	FF	Block RAM
DT Engine 1 Stage	62	96	0
DT Engine 4 Stage	240	332	0
DT Engine 8 parallel instances of 4 Stages	2952	3100	18
Whole design	6442	5336	22

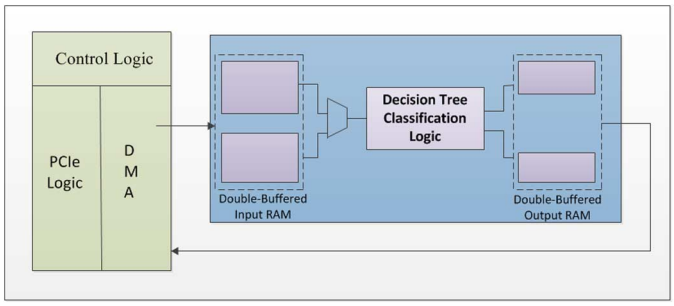

Fig. 6. Streaming architecture.

The utilization of number of slices of the decision tree classification engine with 8 parallel classification subsystems instances is 29%, whereas each instance of 4 stage pipelined decision tree module uses 205 slices bringing it to 2% total utilization. The whole design number of slices utilization is 5386 which is 62% utilization. Thus the proposed architecture in comparison with SMpL-p has reduced hardware complexity of the modules and reduced execution time.

4.5 Data Streaming with High Performance Communication Link

Our ideal architecture would consist of a streaming interface between PCIe and the decision tree classification engine. Using this interface, the host computer could set up DMA transfers to a fixed destination address on the DT peripheral and continually stream data to the limits of the communication link. At the DT peripheral, onboard logic would manage the streaming data such that a double-buffered input memory could be used to maintain constant bandwidth between the host and peripheral. In this way, the decision tree classification engine would hide the addressing complexity from the host. As long as the processing capability of the engine met or exceeded that of the communication link, saturation would not occur.

This architecture allows classification of big data in a streaming manner. Fig. 6 shows the streaming architecture, the double-buffered input and output RAMs are designed to support simultaneous buffering and processing of data. A memory controller switches the first memory from buffering mode to processing mode once the memory is filled, and connects the other memory to the communication unit for buffering. In such a manner the communication overhead is hidden.

6 Conclusion

The proposed architecture has the advantage of being highly scalable and exhibits high levels of parallelism. The performance of pipelined architecture is linearly dependant on the number of data records/tuples and independent of the number of attributes in a particular dataset. Higher levels of parallelism can be achieved by increasing the number of parallel pipelines, also trees of greater depth up to 13 can be modeled by increasing the number of pipeline stages.

The design has the minimum resource utilization thus the power consumed is also an advantage of the binary decision tree classification accelerator engine. Contrary to the previous implementations, we focused on the pipelining of different stages; efficient use of the on- chip memories; and registers to optimize the area used and minimize the clock cycles, thus helping in accelerating the process of classification.

The next phase of this work will focus on obtaining a more modern evaluation platform, implementing a PCIe-based streaming interface to our logic, and characterizing the results. Also making the design concurrent by using multiple FPGA boards connected together using Aurora protocol.

References

[1] R. Narayanan, D. Honbo, J. Zambreno, G. Memik, and A. Choudhary, "An FPGA Implementation of Decision Tree Classification," *Proc. IEEE Int'l Conf. Design, Automation and Test in Europe*, Apr. 2007, pp. 189-194.
[2] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford Univ. Press, 1995.
[3] L. Rokach and O. Maimon, "Top-Down Induction of Decision Trees—A Survey," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 35, no. 4, pp. 476-487, Nov. 2005.
[4] V. Vapnik, *Statistical Learning Theory*. Wiley, 1998.
[5] D.C. Hendry, A.A. Duncan, and N. Lightowler, "IP Core Implementation of a Self-Organizing Neural Network," *IEEE Trans. Neural Networks*, vol. 14, no. 5, pp. 1085-1096, Sept. 2003.
[6] S. Himavathi, D. Anitha, and A. Muthuramalingam, "Feedforward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization," *IEEE Trans. Neural Networks*, vol. 18, no. 3, pp. 880-888, May 2007.
[7] D. Anguita, S. Pischiutta, S. Ridella, and D. Sterpi, "Feed-Forward Support Vector Machine Without Multipliers", *IEEE Trans. Neural Networks*, vol. 17, no. 5, pp. 1328-1331, Sept. 2006.
[8] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.
[9] L. Breiman, J.H. Freidman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*. Wadsworth and Brooks, 1984.
[10] A. Bermak and D. Martinez, "A Compact 3D VLSI Classifier Using Bagging Threshold Network Ensembles," *IEEE Trans. Neural Networks*, vol. 14, no. 5, pp. 1097-1109, Sept. 2003.
[11] S. Lopez-Estrada and R. Cumplido, "Decision Tree Based FPGA Architecture for Texture Sea State Classification," *Proc. IEEE Int'l Conf. Reconfigurable Computing and FPGA's, ReConFig.*, pp. 1-7, Sept. 2006.
[12] J.R. Struharik, "Implementing Decision Trees in Hardware," *Proc. IEEE 9th Int'l Symp. Intelligent Systems and Informatics (SISY)*, pp. 41-46, Sept. 8-10, 2011.
[13] Z. Baker and V. Prasanna, "Efficient Hardware Data Mining with the Apriori Algorithm on FPGAs," *Proc. IEEE Symp. Field Programmable Custom Computing Machines (FCCM)*, 2005, pp. 3-12.
[14] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo, "Fast Discovery of Association Rules," *Proc. Advances in Knowledge Discovery and Data Mining*, pp. 307-328, 1996.
[15] Z. Baker and V. Prasanna, "An Architecture for Efficient Hardware Data Mining Using Reconfigurable Computing Systems," *Proc. IEEE Symp. Field Programmable Custom Computing Machines (FCCM)*, 2006, pp. 67-75.
[16] J. Shafer, R. Agrawal, and M. Mehta, "SPRINT: A Scalable Parallel Classifier for Data Mining," *Proc. Int'l Conf. Very Large Databases (VLDB)*, 1996, pp. 544-555.
[17] M. Joshi, G. Karypis, and V. Kumar, "ScalParC: A New Scalable and Efficient Parallel Classification Algorithm for Mining Large Datasets," *Proc. 11th Int'l Parallel Processing Symposium (IPPS)*, 1998, pp. 573-579.
[18] Q. Li and A. Berma, "A Low-Power Hardware-Friendly Binary Decision Tree Classifier for Gas Identification," *J. Low Power Electron. Appl.*, vol. 1, pp. 45-58, 2011.
[19] T.O. Bachir, M. Sawan, and J.J. Brault, "A New Hardware Architecture for Sampling the Exponential Distribution," *Proc. IEEE Canadian Conf. Electrical and Computer Eng.*, pp 1393-1396, 2008.
[20] V. Podgorelec and P. Kokol, "Evolutionary Induced Decision Trees for Dangerous Software Modules Prediction," *Information Processing Letters*, vol. 82, pp. 31-38, Feb. 2002.
[21] G. Chrysos, P. Dagritzikos, I. Papaefstathiou, and A. Dollas, "Novel and Highly Efficient Reconfigurable Implementation of Data Mining Classification Tree," *Proc. 21st Int'l Conf. Field Programmable Logic and Applications*, pp. 411-416, 2011.
[22] G. Chrysos, P. Dagritzikos, I. Papaefstathiou, and A. Dollas, "HC-CART: A Parallel System Implementation of Data Mining Classification and Regression Tree (CART) Algorithm on a Multi-FPGA System," *ACM Trans. Architecture and Code Optimization*, vol. 9, no. 4, p. 25, Jan. 2013, article 47.
[23] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10-18, 2009.
[24] A. Frank and A. Asuncion, *UCI Machine Learning Repository*, School of Information and Computer Science, Univ. of California, http://archive.ics.uci.edu/ml, 2010.
[25] G. Melli, *The Datgen Dataset Generator, version 3.1*, http://www.datasetgenerator.com, 1999.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.