THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 20, Number 2/2019, pp. 205–212

NEURAL BRANCH PREDICTION: FROM THE FIRST IDEAS, TO IMPLEMENTATIONS IN ADVANCED MICROPROCESSORS AND MEDICAL APPLICATIONS

Lucian N. VINŢAN[1,2]

[1] "Lucian Blaga" University of Sibiu, Computer Science and Electrical Engineering Department,
Emil Cioran Str., No. 4, 550025, Sibiu, Romania
[2] Technical Sciences Academy of Romania, Bd. Dacia 26, Bucharest, Romania
E-mail: lucian.vintan@ulbsibiu.ro

Abstract. This paper presents the idea of dynamic neural branch prediction, from its early theoretical foundations, firstly developed by the author of this paper, up to its implementations in present-day commercial advanced microprocessors. Also, the paper analyses a scientific work that proposes the use of neural branch predictors in predicting the brain's neuronal activity, with useful medical applications. Machine Learning domain significantly fertilized Computer Architecture research during the last years. Dynamic Neural Branch Prediction was one of the earliest successful examples in this sense, being now an interesting fertile story, already used by industry in developing commercial advanced superscalar and simultaneous multithreading microprocessors.

***Key words*:** computer architecture, advanced microprocessors, Two Level Adaptive Branch Prediction, machine learning, Neural Branch Prediction, Brain Activity Prediction.

1. NEURAL BRANCH PREDICTION: CONTEXT AND GENESIS

Accurate dynamic branch prediction becomes more and more important in present-day superscalar and simultaneous multithreading (SMT) microprocessors, due to the increased instruction issue rate and also due to the corresponding super-pipelined instruction processing structures (up to 24 pipelined stages – *Intel I7-Nehalem*). Average prediction accuracies of over 97% are required, because each misprediction involves a significant performance penalty incurred by the corresponding recovery process [1,2]. During the run-time prediction process, dynamic branch instructions should be detected in advance and then both the branch's direction (Taken or Not Taken) and the branch target address must be predicted, hopefully during the instruction fetch phase, thus without interrupting the flow of the dynamic pipelined instructions. High branch prediction accuracies were obtained using a set of prediction methods known as Two-Level Adaptive Branch Prediction that were mainly developed by Professor Yale Patt's research group [3].

A generic Two Level Adaptive Branch Predictor scheme usually produces a prediction based on three quasi-orthogonal pieces of information: the Program Counter's lower bits (PC), the Global History Register (GHR, also called global correlation information; it contains the outcomes, in terms of Taken or Not Taken, of the previously processed branches) and the Local History Register (LHR, also called per branch local correlation information; it contains the outcomes of the previous instances of the current branch). Sometimes the path information of the individual program is added to the predictor as well, alongside the PC, GHR and LHR. It consists of recordings within the GHR whether each branch of the program's path is taken or not [4].

Basically, the GHR is indexing a so-called Prediction History Table (PHT) that contains per branch local history. Further, the addressed LHR is indexing the Prediction Table (PT), containing the Branch Predictors. This lastly addressed Branch Predictor would then generate the prediction. Such branch predictors were implemented as simple Finite State Machines (FSM), usually in the form of saturated up / down counters. After the current branch is resolved, the outcome of the branch adequately updates the GHR, the LHR and the FSM Predictor structures. The complexity of such a scheme is exponential; therefore they involve PHT and PT data structures of huge capacities. In order to reduce this significant complexity, some

hashing schemes between PC, GHR and LHR are frequently used. Such Two Level Adaptive Branch Predictors were and still are implemented in commercial advanced superscalar and SMT microprocessors.

In [5] the authors firstly demonstrated that all Two-Level Adaptive Branch Predictors implement special cases of the more general Prediction by Partial Matching (PPM) algorithm, which is widely used in data compression and image processing. Thus it was shown that Two Level Adaptive Branch Predictors represent a less complex approximation of an ideal Markov predictor. They use the PPM algorithm to compute a theoretical upper bound on the accuracy of branch prediction as well. This work is a very important one, because it helps in deeply understanding the theoretical bases of the well-known Two Level Adaptive Branch Prediction Schemes. Before this paper was published, the branch predictors were designed and presented based only on empirical, ad-hoc intuitive methods, without a solid mathematical support and understanding. Unfortunately, this profound paper was not sufficiently cited and appreciated in the Computer Architecture community, compared with other valuable papers in the field. However, some Computer Architecture researchers like Andre Seznec and Pierre Michaud were influenced by the PPM model in their excellent works on branch prediction. So, for example, in one of their papers [6] a branch predictor is presented that "*uses partially tagged components as a PPM-like predictor*." The author of this paper was also influenced by the PPM predictor in developing predictors dedicated to next location prediction problem in a ubiquitous computing environment [26].

Dynamic branch prediction with neural methods, as an alternative to the Two Level Adaptive Branch Prediction (Markovian) model, was firstly proposed by Lucian Vinţan, the author of this paper, in 1999 [7]. The paper shows that a single Learning Vector Quantization-based branch predictor might be effective, comparable with thousands of classical FSM branch predictors organized in a Prediction Table. Furthermore, in another early paper [29], Vinţan proposed the first Multi-Layer Perceptron Branch predictor. These initial neural branch prediction ideas were later significantly developed, especially by Dr. Daniel Jiménez [8,9,10] and subsequently by many other Computer Architecture researchers, from both academia and industry. All these subsequent papers refine the initial ideas and make them feasible to be implemented in hardware. After almost 20 years from our first neural branch predictor proposal, an analysis of this concept might be useful, covering the development from initial ideas into the refined effective implementations in modern processors and other useful applications.

2. THE FIRST PROPOSED NEURAL BRANCH PREDICTORS

One of the main research objectives of our first paper which proposed a neural branch predictor [7] was to use neural networks to identify new correlations that can be exploited by these predictors. The author has shown that the developed Learning Vector Quantization (LVQ) neural branch predictor can exploit deeper correlations at linear complexities, rather than exponential complexities typical to the Two-Level Adaptive Branch Prediction schemes. An additional and very important fact is that the learning process of these supervised neural LVQ predictors can be mathematically proven to converge based on the theory of the gradient descent learning algorithm, as we tried to show in [11]. In contrast, a mathematical proof of the convergence of the FSM learning algorithms, implemented in classical branch predictors, is not possible. Moreover, the simplistic learning algorithms of the Two Level Adaptive Branch Predictors were developed in a purely empirical manner and they are entirely based on common sense and statistical empirical intuitions or measurements, not on a rigorous mathematical model.

As an alternative to classical Markovian Branch Predictors, this new approach replaces all the FSM predictors (one FSM predictor per branch) with a single simple LVQ neural network, used by all dynamic branches. By proposing this approach, the author is the first to show that the branch prediction problem can be viewed as a problem of binary pattern classification. More precisely, the problem is to dynamically classify a set of binary vectors, each vector representing local/global histories, branch address, path information etc. in two classes: the Taken class and the Not Taken class. It was shown that the LVQ neural algorithm was suitable for such an innovative approach. The quantitative results show that this unique (global) LVQ predictor achieved prediction accuracies comparable with the state of the art classical Markovian branch predictors. The author wrote in a visionary manner: *"At this time, our intuition is that a simplified neural network predictor could be designed within the timing restraints of a superscalar processor. [...] the cost would be far less than one of Two Level Adaptive predictors and it may even be*

possible to implement multiple cut-down neural network predictors, associated which each branch." [7], anticipating somehow the future simple single cell perceptron branch predictors. These perceptron-based branch predictors replaced the FSM predictors at far lower hardware complexities (see below). This first paper was quite fertile, being cited until the year 2018 by about 70 scientific published works, according to Google Academic, including top level Q1 Clarivate Analytics journals like *IEEE Transactions on Computers, ACM Transactions on Computer Systems, IEEE Micro*, etc., according to Journal Citation Report (JCR) 2017 or top level conferences like the *International Symposium on Computer Architecture* (ISCA). It was one of the very early papers that proposed the application of Machine Learning methods in Computer Architecture research. In [10], a paper published at a top scientific conference in the field of Computer Architecture, Dr. Daniel Jiménez, who developed the first single cell perceptron branch predictor feasible to be implemented in hardware, wrote: *"Dynamic branch prediction with neural methods was first proposed by Vintan [...]".* In the paper [12], published in a very prestigious scientific journal, the authors have written: *"The idea of the neural branch prediction was originally introduced by Vintan [...]"*. In [13] it is written: *"Vintan pioneers the idea of using perceptrons for branch prediction".* In [14] the authors, two of which were working for the Intel Corporation, wrote: *"Perceptrons have been proposed earlier for branch prediction [Vintan...]"*. Dr. A. Seznec and Dr. P. Michaud also cited this paper in their interesting and valuable work presented in [6]. Dr. Pierre Michaud wrote in his recent paper, published in a top level scientific journal [15]: *"In 1999-2000, two research teams, independently, started exploring the use of artificial neural networks for branch prediction [Vintan, Jiménez]".* All of these valuable scientific papers have cited our first paper focused on this issue [7].

Further, Vinţan introduces and evaluates a new global neural predictor, called the multi-layer perceptron branch predictor [29]. Such a neural network replaced all the conventional FSM predictors. The implemented learning algorithm was backpropagation. It was shown that the multi-layer perceptron branch predictor (MLPBP) outperformed a conventional Two Level Adaptive Branch Predictor (more precisely a so-called GAp predictor) with 4 additional percent points in average prediction accuracy, which is remarkable. Very important as well is the fact that the author evaluates the benefits of offline (static) training for the designed MLPBP scheme and he proves that this pre-training process further increases the prediction accuracy with about 2%. This work was further developed and extended by the author with the help of an international group of researchers, with results published in [16]. In spite of its long latency of prediction, a MLPBP scheme with backpropagation learning algorithm could successfully predict even non-linearly separable branches. It is well-known that a simple single cell perceptron is not able to do this [11].

3. THE SINGLE CELL PERCEPTRON DYNAMIC BRANCH PREDICTION

The first Single Cell Perceptron Branch Predictor feasible to be implemented in a superscalar / SMT pipelined processor was introduced in [8,9]. Instead of having a FSM predictor like in the conventional approaches, each branch now had a simple perceptron predictor. The perceptron branch prediction rule is simple: if its output value is positive, the branch will be predicted as Taken; otherwise, it will be predicted as Not Taken. This prediction process is done during the perceptron's feed-forward phase. Taking into account that the perceptron's binary input represent the branch's local/global history (X) and the corresponding weights (W) represent signed integers (bytes), the prediction is computed as the $\text{sign}(W*X=\sum_{k=0}^{n} W_k X_k)$, $X_k \in \{0 \text{ or } -1,\ 1\}$, where $*$ operator represents the dot product between vectors W and X. Thus, $W*X$ could be implemented in a parallel manner, using Tree of Adders. Such an parallel implementation needs maximum $(n-1)$ adders (hardware complexity), involving $(\log_2 n)$ adder levels (timing complexity). Thus, the perceptron branch predictor hardware complexity is $O(n)$. Actually, the hardware perceptron branch predictor is more effective, being implemented as a 4-stage pipelined Wallace-Tree of 3 to 2 Carry-Save Adders (WT-3-to-2-CSA) which reduces adding n bytes (weights), having normally $O(\log_2 n)$ timing complexity, as we already pointed out, to the problem of adding just two bytes [9,35], having thus $O(1)$ timing complexity. Non-pipelined WT-3-to-2-CSA's timing complexity (depth, noted with h) is $O(\log_{3/2} n) < O(n-1)$, the last one corresponding to a "naïve" conventional adder, containing $(n-1)$ cascaded two bytes adders. More precisely, the following mathematical relationships are fulfilled: $h(n) = 1 + h(\lceil 2n/3 \rceil)$ with $h(3)=2$; $n(h) = \lfloor 3n(h-1)/2 \rfloor$; $2*1.5^{h-1} < n(h) < 2*1.5^h$, where $n(h) =$ the

maximum number of inputs for an h-level Carry-Save Adders tree, and $h(n)$ = the depth for $n > 2$ summands (inputs); $\lceil x \rceil$ represents x's rounding (ceiling function), and $\lfloor x \rfloor$ represents x's cropping (floor function). The computation might be even quicker because only the result's sign bit is needed to compute the prediction. If the result's sign bit is logical 1, the prediction is "Taken"; otherwise, it is "Not Taken". Related to the perceptron branch predictor's learning process, Dr. Jiménez proposed a simplified algorithm derived from the well-known gradient descent algorithm, feasible to be implemented in hardware:

For each bit do in parallel
  if desired perceptron's output (t) = *perceptron's input* x_k *then*
    $w_k = w_k + 1$
  else
    $w_k = w_k - 1$
  end if

In the case of the perceptron branch predictor, the vector X represents the input vector, as in a classical Two-Level Adaptive Branch Predictor (X usually represents local/global histories, branch's address, path information, etc.) It is considered that if, for a given X value, the desired perceptron's output value is $t = -1$ the branch was not taken and if $t = 1$ the branch was taken. Since t and input bits x_k are always either -1 or 1, this learning algorithm increments the k^{th} weight (w_k) when the branch outcome agrees with the sign of x_k, and decrements the weight when it disagrees with the corresponding input value. In contrast with the Two-Level Adaptive Branch Predictors case, the input X is not indexing a Prediction Table. The vector X in this case represents just an input value for the single cell perceptron branch predictor. In this way, the perceptron's complexity is just linear, and not exponential, as in the case of a Two-Level Adaptive Branch Prediction scheme. This fact represents a great advantage for the perceptron predictor. Of course, perceptron predictors are very effective in predicting linearly separable branches but they cannot successfully predict branches that are not separable by a hyper-plane in the orthogonal space of their representation (PC, GHR, LHR). In contrast, Two Level Adaptive Branch Predictors could be effective in predicting such branches because each context pattern X, representing the most recent bits of branch's history, indexes its own FSM predictor.

In [9], some very clever simplifications are proposed, with the goal of adapting the dynamic perceptron branch prediction algorithm to hardware implementation restrictions and of reducing the prediction's latency. This paper, published in a prestigious Q1 Clarivate Analytics journal (according to JCR 2017), cited the first paper that introduces the neural branch prediction idea, developed by Lucian Vinţan [7]. It was shown that additional performance gains can be found for branch history lengths of up to 66 bits, which is very remarkable. Obviously, such long histories are impossible to exploit by Two Level Adaptive Branch Predictors, due to the corresponding exponential complexity. However, the main disadvantage of this proposed perceptron predictor consists in its relatively high latency, in spite of the use of ingenious high-speed arithmetic tricks, which essentially reduce the perceptron predictor's implementation to a binary adder. In order to further reduce the prediction latency, in [10] a perceptron predictor is presented which chooses its weights for generating a prediction according to the path of the current branch, rather than according to the Program Counter of the branch and a global history register. In order to reduce the prediction's latency each branch is predicted correlated with the previous branches' predictions rather than with the previous branches outcomes. Each time a branch is fetched, its 0^{th} weight is added to a running total that has been kept for the last branches, with each summand added during the processing of the previous branches [26]. This so-called path-based neural prediction method has two main important advantages:

- The prediction latency is almost completely hidden because the prediction computation can begin in advance of the effective prediction, with each pipelined step proceeding as soon as a new element of the path is executed.
- Prediction accuracy is improved because the predictor incorporates path information as an additional new useful input [4].

In [33] the authors are investigating the feasibility of deep learning algorithms and structures, in particular Deep Belief Networks, to branch prediction research. Unfortunately, the proposed approach does not outperform present-day state of the art branch predictors in a convincing manner. Anyway, exploring

deep learning methods, including Deep Convolutional Neural Networks, might be an effective alternative to perceptron based branch predictors.

4. IMPLEMENTATIONS IN COMMERCIAL ADVANCED MICROPROCESSORS

Some prestigious companies, such as Intel, Oracle, AMD, Samsung etc., were interested in implementing such neural branch predictors in their modern commercial microprocessors. For example, neural branch prediction methods were implemented in an out-of-order version of Intel's IA-64 simulator (Itanium Processor Family) [17]. However, we do not officially know if Intel already implements a neural branch predictor in its commercial processors, although the author strongly believes this. Usually, details about CPUs design and implementations kept secret by the semiconductor industry.

Anyway, as far as we know, the first company that officially recognized the implementation of a neural branch predictor in its advanced microprocessors was Samsung. The quad-core 2.6 GHz, sub 3-Watt/core microprocessor is named Samsung Exynos M1, codenamed Mongoose and running ARMv8-A code 64 bit/32 bit compliant, having Brad Burgess as chief architect [18]. Actually the implemented branch predictor is a hybrid one, allowing 2 branches/cycle processing, containing a neural network (simple perceptron) branch predictor, a 4 k-entry main Branch Target Buffer, a 64-entry Call/Return stack (for predicting Return instruction's target addresses), an Indirect Jumps/Calls Predictor (for predicting indirect Jumps/Calls' target addresses), a Loop Predictor, etc. [30]. Unfortunately, technical details about the specific design of the neural branch predictor are not available. For example, Samsung Galaxy S7 (2016) and S7 Edge smartphones are using this processor. The corresponding branch predictor is a perceptron predictor and, according to our knowledge, it was designed by D. Jiménez. According to some publicly available information, it seems that other similar Neural Branch Predictors have been implemented, specifically perceptron predictors, in some advanced commercial microprocessors like Sun/Oracle Sparc T4-4 (2011; it is used in 8 core 3 GHz Sun/Oracle Sparc T4 data center servers), AMD Bulldozer (2011; it probably has local/global neural branch predictors; it is used in Opteron 6200 servers), AMD Piledriver (2012), AMD Bobcat/Jaguar (2014; it might use 26 bits of global history; the CPU is used in notebooks, tablets and servers like Opteron X2100), AMD Zen (2016/2017, 4 cores per CPU, used in Epyc 7000 series servers) and further AMD Ryzen (2017, for mobile and embedded computers, high-end desktops, etc.) [19,20,21].

The recent IBM z14 microprocessor produced in 2017 and designed for IBM Z mainframes (up to 170 cores running at 5.2 GHz with a performance > 146,000 MIPS) contains a perceptron branch predictor, too [32]. This perceptron direction branch predictor is dedicated to difficult predictable branches, thus branches that aren't highly predictable with the classical Markovian predictors, implemented through Branch Target Buffers (BTB) and Prediction / Pattern History Tables (PHT, containing branch's local history) structures. The predictors are predicting based on the local history, global history and path information [1,2,3,4]. The perceptron's weights table represents an 32 entries 2-way set associative structure that corresponds to the so-called path history bits. This binary information is obtained through a hash of the past taken branches Program Counters' values (thus it represents a global path). Each global path history bits corresponds to a certain used weight. A bit of 1 involves a positive weight during the summation process; a bit of 0 (−1) a negative one. As in [8] the prediction (Taken / Not Taken) represents the weights sum's sign. The weights sum's value represents the correlation's value with the path history. The learning process is similar with that presented in [9,10,11], implementing a simplified gradient descent algorithm. Actually, the whole direction prediction scheme is a hybrid, Markovian (BTB, PHT) – neural one. The meta-predictor is based on the confidence information attached to each predictor (usefulness degree). The confidence information is incremented for a correct prediction and decremented otherwise. The predictor having the highest current confidence will provide the prediction. The designed branch predictor simultaneously predicts up to 9 branches / cycle. The Processor Unit chip contains 10 two-way simultaneous multithreading and 6 instructions / cycle cores working at a clock frequency of 5.2 GHz.

The AMD Bulldozer branch prediction mechanism is a hybrid one as well, implemented using a local predictor and a global predictor. Very likely, the branch predictor is based on simple hashed perceptrons, like in the AMD Ryzen processor [22]. The Sparc T4/T5 (S3 core from Oracle) implements an advanced hybrid branch predictor shared by all active threads, consisting in a small low-latency branch target cache for the

branch's target address prediction, which is backed by a very accurate simple perceptron-based direction predictor [23]. Evaluations have shown that the perceptron branch predictor generates the best cost-performance trade-off. All these commercial implementations clearly show that the neural branch prediction concept was a fertile one in the field of Processor Architecture, being a useful complementary alternative to the well-known and more complex Two Level Adaptive Branch Predictors. In [31] the authors proposed an original effective functional link artificial neural network branch predictor. Quantitative comparisons with a perceptron-based branch predictor show that the proposed design is comparable with the state of the art perceptron predictors from both prediction's accuracy and latency points of view.

In this context, the idea of hybrid branch prediction is natural, meaning that some different Markovian and Neural branch predictors might work well together, exploiting their advantages in a synergistic manner. In this case, an important research challenge is to design and implement an effective (adaptive) meta-predictor [25,26,34]. Many commercial advanced superscalar microprocessors implement two, three or even more distinct branch predictors.

5. BRAIN ACTIVITY PREDICTION USING NEURAL BRANCH PREDICTORS

Brain-Machine Implants for studying neuronal activity in vivo – simultaneously monitoring hundreds of neurons – are rapidly improving nowadays. Scientists are integrating dedicated low-power processors (for example, energy-efficient ARM CortexM cores) on neuroprostheses in order to achieve sophisticated computation, like performing signal processing on neuronal spiking data. Such processors must be energy efficient, adaptively exploiting their special low-power processing modes. For example, present-day brain-machine implants cannot exceed 50-300 mW power budgets [24]. The key idea for the optimization of a processor's energy consumption is to place it in a low-power processing mode in the absence of neuronal firing. This strategy involves the accurate prediction in the brain's neuronal activity (in this case, CPU transits in normal power mode, operating at normal clock frequency) or a break in this activity (CPU transits in low power mode). In [24] the author proposes the use of state of the art branch predictors in order to predict the brain's neuronal activity (low or high). This prediction is requested for two main reasons: to correspondingly manage the CPU in its idle / active processing modes (1) and to acquire and process some of the brain's signals during the transitions from low to high neuronal activity and vice versa, and further during the brain's stable activity (2). In its nominal operation, the embedded CPU analyzes about 500 ms of the cerebellum's activity. Acquiring and processing cerebellum signals are of great interest for studying and treating some diseases like memory problems, movement, equilibrium, and motor learning disorders, hallucination and psychosis, epilepsy (including its prediction) etc.

More precisely, a branch predictor can predict if a neuron fires or not (relaxation) at a given discrete time (Purkinje neuron firing / quiescence is similar with a branch taken / not taken). Each monitored neuron's activity is predicted by a certain branch predictor. The embedded ARM CortexM processor used in this case was modified so that a part of its branch predictors is only used for this special brain neuronal activity prediction. Therefore, in idle low-power mode a hardware Finite State Machine co-opts a part of the CPU's branch predictor for performing neuronal activity prediction. The neuronal activity prediction is necessary because the (about 10) milliseconds of neuronal activity leading up to synchronized firing are of great interest. Thus it is necessary to predict neuronal activity ahead of time. The author performs brain surgeries on mice to extract 26 minutes of neuronal spiking activity from their cerebella and, based on this useful history information, evaluates the ability of some branch predictors, implemented in the embedded processor belonging to the brain-machine implant, to predict a high or a low neuronal activity in the cerebellum. More specifically, the Purkinje neurons in the cerebellum are being monitored in this work through a microelectrode array, called Utah array, used together with corresponding analog-to-digital converters (ADCs) which digitizes neurons'activities to collect intra-cellular neuron's recordings. These state of the art devices contain several tens/hundreds connections. The question is: what is a low or a high neuronal activity in the cerebellum? Purkinje firing can be separated into two classes: unsynchronized (low activity) and synchronized firing (high activity) within a given set of Purkinje neurons. As a consequence, the predictor enables energy-efficiency by using low-power modes when Purkinje synchronization is absent, and enables nominal operation when synchronized firing activity is predicted. A special logic implemented in the embedded CPU assesses if enough neurons are predicted to fire in order to constitute synchronization

and, therefore, to trigger the processor in its normal processing mode. In this way, it is increased the brain implant's battery lifetime, too.

The author shows that some neural branch predictors based on perceptrons achieve a prediction accuracy of up to 85% for this task, being far better than Two Level Adaptive Branch Predictors, especially due to their ability to exploit deep correlations. This fact is remarkable because synchronization requires multiple neurons to be simultaneously predicted correctly as transiting in their firing states. These perceptron branch predictors are predicting based on both a neuron's local history (intra-neuronal recordings) and inter-neuronal correlations (global history or correlation in terms of branch prediction). This last correlation is shown to be very important because micro-bands of correlated neurons synchronize. Taking into account that perceptron branch predictors scale linearly with the global history, and not exponentially like classical branch predictors, they can effectively exploit the deep correlation of the current neuron's behavior with its corresponding neighbors. Also, the author developed a cerebellar monitoring implant and uses the dedicated processor's branch predictor to guide the energy management of the CPU. Initially, the processor was set in its idle low power mode (however, leaving the corresponding branch predictor on!) but, when the perceptron branch predictor detects a high neural activity in the brain, the processor transits in its normal processing mode. The author reports that this innovative approach saves up to 59% of the processor's energy [24]. It is useful to point out that this breakthrough contribution cites two of the papers developed by the author of this work [7,29], firstly introducing the neural branch prediction idea.

6. CONCLUSIONS

This article presented the idea of dynamic neural branch prediction, from its early theoretical foundations up to its implementation in modern superscalar / SMT microprocessors and beyond. Based on some recent commercial implementations, it seems that the neural branch predictor would be an important component in the hybrid branch predictor of every present-day advanced superscalar or SMT microprocessor. We also analyzed a very interesting scientific work that proposes the use of neural branch predictors in predicting the brain's neuronal activity, as being of low or high level, with very useful medical applications. This prediction process is implemented through brain-machine implants and it is very useful for studying and effectively treating some neuronal diseases.

The neural branch prediction idea is fertile because, more generally, using state of the art Machine Learning methods in Computer Architecture design could significantly improve the CPU's multi-criterial performances [25]. Adapting such powerful techniques to hardware constraints is a true art, involving not only serious technical knowledge but also researcher's intuition and talent.

The field of Machine Learning has already fertilized research in Computer Architecture. Dynamic Neural Branch Prediction – pioneered by the author of this paper – was one of the earliest successful examples in this sense, already used by the industry in developing commercial advanced superscalar and simultaneous multithreading microprocessors. The author believes that some further applications of neural prediction implemented in hardware might be seen in dynamic instruction value prediction [26,27,28]. For example, dynamically classifying instructions could be very useful for predicting values of instructions (for example, classifying dynamic instructions as predictable or not predictable). Also, the idea of hardware neural prediction might be useful in implementing a meta-predictor composed of some hybrid predictors that are simultaneously working together. More generally, exploiting the synergism of different domains could generate effective interdisciplinary approaches.

REFERENCES

1. L. VINȚAN, *Instruction level parallel architectures* (in Romanian), Edit. Academiei Române, Bucharest, 2000.
2. J. HENNESSY, D. PATTERSON, *Computer architecture: A quantitative approach*, Morgan Kaufmann (Elsevier), 6-th Edition, 2018.
3. T. YEH, Y.N. PATT, *A comparison of dynamic branch predictors that use two levels of branch history*, ISCA-20 Conference, San Diego, May 1993.
4. L. VINȚAN, C. EGAN, *Extending correlation in branch prediction schemes*, 25th Euromicro International Conference, Milano, Italy, September 8-10, 1999.

5. I-C. CHENG, J. COFFEY, T. MUDGE, *Analysis of branch prediction via data compression*, 7-th International Conference on Architectural Support for Programming Languages and Operating Systems, October 1996.
6. A. SEZNEC, P. MICHAUD, *A case for (partially) tagged geometric history length branch prediction*, Journal of Instructions Level Parallelism, February 2006.
7. L. VINȚAN, *Towards a high performance neural branch predictor*, International Joint Conference on Neural Networks, Washington DC, USA, July 10-16, 1999.
8. D.A. JIMÉNEZ, C. LIN, *Dynamic branch prediction with perceptrons*, Seventh International Symposium on High Performance Computer Architecture (HPCA-7), Monterrey, NL, Mexico, 2001.
9. D. JIMÉNEZ, C. LIN, *Neural methods for dynamic branch prediction*, ACM Transactions on Computer Systems, **20**, *4*, pp. 369–397, November 2002.
10. D. JIMÉNEZ, *Fast path-based neural branch prediction*, The 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-36), San Diego, USA, December 2003.
11. L. VINȚAN, *Dynamic neural branch prediction fundamentals*, Buletinul AGIR, **XXI**, *1*, pp. 64–71, 2016.
12. D. TARJAN, K. SKADRON, *Merging path and gshare indexing in perceptron branch prediction*, ACM Transactions on Architecture and Code Optimization, **2**, *3*, pp. 280–300, September 2005.
13. J. SINGER, G. BROWN, I. WATSON, *Branch prediction with Bayesian networks*, Proc. Workshop on Statistical and Machine Learning Approaches Applied to Architectures and Compilation, pp. 96−112, 2007.
14. H. AKKARY, S. SRINIVASAN, R. KOLTUR, Y. PATIL, W. REFAAI, *Perceptron-based branch confidence estimation*, 10-th International Symposium on High Performance Computer Architecture, Madrid, February 14-18, 2004.
15. P. MICHAUD, *An alternative TAGE-like conditional branch predictor*, ACM Transactions on Architecture and Code Optimization, **15**, *3*, 24 p., 2018.
16. C. EGAN, G. STEVEN, P. QUICK, R. ANGUERA, L. VINȚAN, *Two-level branch prediction using neural networks*, Journal of Systems Architecture, **49**, *12-15*, pp. 557–570, Elsevier, December 2003.
17. E. BREKELBAUM, J. RUPLEY, C. WILKERSON, B. BLACK, *Hierarchical scheduling windows*, Proceedings of the 34th International Symposium on Microarchitecture, Istanbul, Turkey, December 2002.
18. C. WILLIAMS, *Neural network spotted deep inside Samsung's Galaxy S7 silicon brain*, The Register, 22 August 2016.
19. ORACLE, *Oracle Sparc T4-4*, http://www.oracle.com/us/products/servers-storage/servers/sparc-enterprise/t-series/sparc-t4-4-faq-496527.pdf (accessed on August 7, 2018)
20. J. WALTON, *The AMD Trinity Review (A10-4600M): A new hope*, AnandTech, May 15, 2012.
21. J. DAVE, *AMD Ryzen reviews, news, performance, pricing, and availability*, PCGamesN, December 6, 2017.
22. A. FOG, *The Microarchitecture of Intel, AMD, and VIA CPUs*, 2014, available online at http://www.agner.org/optimize/microarchitecture.pdf (accessed on August 7, 2018).
23. M. SHAH, R. GOLLA, G. GROHOSKI, P. JORDAN, J. BARREH, J. BROOKS, M. GREENBERG, G. LEVINSKY, M. LUTTRELL, C. OLSON, Z. SAMOAIL, *Sparc t4: A dynamically threaded server-on-a-chip*, *IEEE Micro*, **32**, *2*, pp. 8–19, 2012.
24. A. BHATTACHARJEE, *Using branch predictors to predict brain activity in brain-machine implants*, Proceedings of MICRO-50, Cambridge, MA, USA, October 14-18, 2017, online available at: https://www.cs.rutgers.edu/~abhib/abhib-micro17.pdf (accessed on August 7, 2018).
25. L. VINȚAN, R. CHIȘ, M. ALI ISMAIL, COȚOFANĂ C., *Improving computing systems automatic multi-objective optimization through meta-optimization*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **35**, *7*, pp. 1125–1129, July 2016.
26. L. VINȚAN, *Prediction techniques in advanced computing architectures*, Matrix Rom Publishing House, Bucharest, 2007.
27. Á. GELLÉRT, A. FLOREA, U. FIORE, P. ZANETTI, L. VINȚAN, *Performance and energy optimisation in CPUs through fuzzy knowledge representation*, Information Sciences, **476**, pp. 375–391, February 2018.
28. Á. GELLÉRT, L. VINȚAN, *A multicore architecture with selective load value prediction*, Proceedings of the Romanian Academy, Series A: Mathematics, Physics, Technical Sciences, Information Science, **19**, *4*, pp. 597–604, 2018.
29. L. VINȚAN, *Towards a powerful dynamic branch predictor*, Romanian Journal of Information Science and Technology, **3**, *3*, pp. 287–301, 2000.
30. B. BURGESS, *Samsung Exynos M1 processor*, Hot Chips: A Symposium on High Performance Chips, Samsung Austin R&D Center, Cupertino, California, August 2016.
31. A.K. SAMAL, P.K. MALLICK, J. PRAMANIK, S.K. PANI, R. JELLI, *Functional Link Artificial Neural Network (FLANN) based design of a conditional branch predictor*, In: "Cognitive Informatics and Soft Computing", (eds. P. Mallick et al.), vol. 768, Springer, Singapore, 2019.
32. C. JACOBI, A. SAPORITO, M. RECKTENWALD, A. TSAI, U. MAYER, M. HELMS, A.B. COLLURA, P.K. MAK, R.J. SONNELITTER, M.A. BLAKE, T.C. BRONSON, *Design of the IBM z14 microprocessor*, IBM Journal of Research and Development, **62**, *2-3*, 2018.
33. Y. MAO, J. SHEN, X. GUI, *A study on deep belief net for branch prediction*, IEEE Access, **6**, pp. 10779–10786, 2018.
34. L. VINȚAN, *Towards synergic meta-algorithmic approaches in complex computing systems*, Romanian Journal of Information Science and Technology, **20**, *3*, pp. 241–255, 2017.
35. C.S. WALLACE, *A suggestion for a fast multiplier*, IEEE Trans. on Electronic Computers, February 1964.

Received September 10, 2018