THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 19, Number 4/2018, pp. 597–604

INFORMATION SCIENCE

A MULTICORE ARCHITECTURE WITH SELECTIVE LOAD VALUE PREDICTION

Arpad GELLERT, Lucian VINTAN

"Lucian Blaga" University of Sibiu, Computer Science and Electrical Engineering Department,
Emil Cioran Str., No. 4, 550025 Sibiu, Romania

Corresponding author: Arpad GELLERT, E-mail: arpad.gellert@ulbsibiu.ro

Abstract. This paper presents how a multicore architecture can be improved with selective load value prediction. We integrated into the Sniper multicore simulator a small and fast per-core load value prediction table which keeps the long latency load instructions and their last result values, in order to predict the next values on further executions, unlocking thus in a speculative manner the subsequent dependent instructions. The evaluations have been done on the Splash-2 parallel benchmarks and shown an average speedup of about 4% and 1.25% energy consumption reduction.

Key words: multicore architecture, selective load value prediction, speculative execution, sniper simulator, benchmarking.

1. INTRODUCTION

In the last years the research in computer architecture was focused on multicore and manycore systems which mainly exploit the parallelism from concurrent programs. Some of the techniques applied to increase the performance in superscalar processors can have benefits in multicore systems, too. One such technique is load value prediction which speculates the results of loads to unlock subsequent dependent instructions.

In this work we have enhanced the Sniper state of the art multicore simulator with load value prediction capabilities. For doing this, we have integrated into Sniper private (per-core) Load Value Prediction Tables (LVPT). The value prediction is selectively applied only on load instructions with miss in the first level of data cache (DL1). In this way, by focusing only on the high latency loads, a small and fast LVPT is enough to take all the benefits of load value prediction in terms of performance and energy consumption. Thus, we have used the LVPT structure in order to apply the Selective Load Value Prediction (SLVP) technique. We have applied a manual design space exploration of the proposed parameterized speculative architecture on the Splash-2 parallel benchmarks.

Our objectives are to analyze the prediction accuracy of the SLVP, the speedup and the energy consumption of multicore microarchitectures enhanced with SLVP, as well as to investigate the correlation between the obtained speedups and the number of critical loads (with miss in the DL1 cache). We believe that the importance of our work could be significant, taken into account that, as far as we know, nobody investigated the benefits of implementing value prediction in multicore systems.

The rest of the paper is organized as follows. Section 2 presents the state-of-the-art of value prediction techniques. Section 3 describes the SLVP unit and how it is integrated into a multicore architecture. Section 4 presents the simulation methodology. Section 5 presents the evaluations. Section 6 concludes the paper.

2. RELATED WORK

Value prediction was first introduced by Lipasti et al. in [9] and it was further intensively investigated in monocore processors. Monocore architectures enhanced with direct mapped SLVP tables have been presented in our previous works [5, 6, 7]. In [8], we have parameterized the SLVP table in a monocore

microarchitecture, allowing to access it with instruction or data address, to use multiple values per entry, set-associative table organization, selective access on miss in the L1 data cache or in the L2 unified cache. The evaluations performed with M-SIM on the SPEC 2000 benchmarks have shown that a set-associative SLVP table is more effective than a direct mapped one and using multiple values per SLVP entry is better.

In [11], Perais and Seznec proposed an efficient confidence estimation mechanism for value prediction implemented in monocore architectures. They used 3-bit confidence counters, predicting only on saturated corresponding counter and resetting the counter on misprediction. In fact, with this highly selective confidence mechanism the prediction accuracy is between 95-99% but the corresponding coverage is lower. In [13], the authors have proposed an effective block-based value prediction scheme which associates the predicted values with fetch blocks instead of distinct instructions. They have also presented the Differential VTAGE predictor (D-VTAGE), an improved VTAGE predictor, which uses stride-based prediction instead of last value prediction. The experimental results have shown a remarkable average speedup of 11.2%. In [12], the authors proposed the Early Out-of-order Late Execution (EOLE) superscalar microarchitecture. EOLE delays the value prediction validation in the pipeline until the commit stage. By checking the correctness only at commit time, the authors avoid selective replay and enforce complete pipeline squash on misprediction, simplifying thus the design. In both works [11] and [12] the value prediction is applied on all instructions in mono-threaded architectures. In contrast with these valuable methods, we apply the value prediction in a multicore architecture, selectively, by focusing only on critical load instructions.

In [10], the authors have shown that dynamic instructions' value prediction can violate the sequential consistency in microarchitectures that support multithreading and / or multiprocessing. The problem can occur when dynamic value prediction is applied in a code sequence which manipulates pointer-based shared variables. Thus, the authors have shown that predicting an instruction's value and later validating this prediction according with the instruction's obtained result might be not sufficient in a multicore system, due to some potential shared variables consistency errors, even if these values were correctly predicted. In order to solve such shared variables consistency problems we applied in our developed simulation framework the value-based detection solution proposed in [10].

3. SELECTIVE LOAD VALUE PREDICTION IN MULTICORE ARCHITECTURES

The proposed SLVP technique requires per-core LVPTs within the multicore microarchitecture. The role of the LVPT is to exploit load value locality through value prediction. It keeps the last data values of the critical load instructions with the hope that they will have the same outputs on eventual next dynamic executions, based on the value locality statistical principle. On next occurrences, if the attached predictability confidence is sufficiently high, the stored values can be speculatively used by the subsequent dependent instructions, increasing thus the performance. Based on the structure of a Nehalem core presented in [4], we provide in Fig. 1 how a dual core architecture can integrate SLVP. Obviously, this organization can be extended to any number of cores. We have evaluated configurations consisting in 1, 2, 4, 8 and 16 cores. As Fig. 1 illustrates, the LVPT is private together with the IL1 cache, DL1 cache, L2 unified cache, branch predictor (BP), reordering, paging, and execution units; only the L3 cache is shared by the cores. The generic structure of the LVPT is presented in Fig. 2.

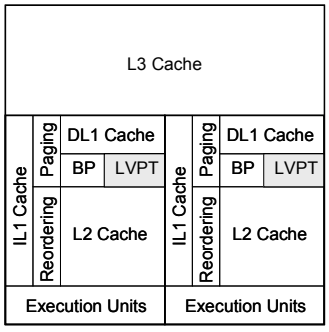

Fig. 1 – Dual Core Microarchitecture with SLVP.

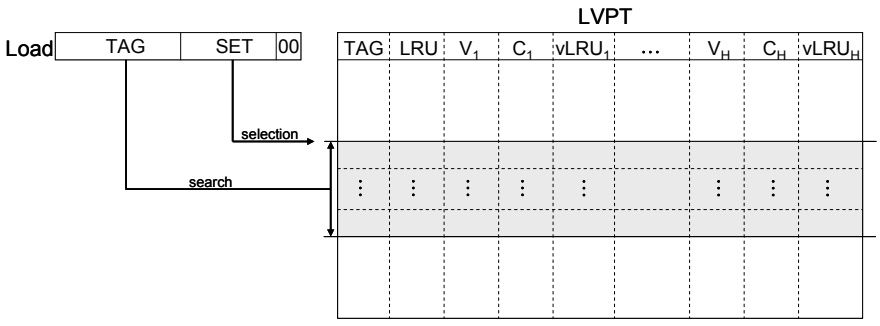

Fig. 2 – The LVPT structure.

Each LVPT entry has a TAG field consisting in the most significant bits of the load instruction's address, a LRU field necessary for the decisions regarding the replacements within the set-associative table, a history of the last distinct values, each such value V having associated a confidence automaton C and a vLRU field. The role of the vLRU fields is to decide which one of the H stored values must be replaced when a new one occurs. The vLRU is set to the maximum for the correct value and is decremented for the other values. The role of the confidence automaton is to dynamically classify each load value kept in the LVPT into unpredictable or predictable. We denote the confidence automata U/P, where U is the number of unpredictable states and P is the number of predictable states. We have implemented the confidence automaton as a saturating counter which is incremented on correct predictions and decremented on mispredictions. The initial state is the unpredictable one.

When a load instruction with miss in the DL1 cache occurs, the SET field (consisting in the least significant bits of the load instruction's address, excluding the last two bits which are always 0) is used to select the corresponding set from the LVPT. After that, the TAG is searched within the TAG field from the selected set. If it is not found, the load instruction is normally executed and, after this, it is inserted into the selected set of the LVPT by replacing the least recently used load instruction from that set (the entry with the lowest LRU). In that case, all the confidences are reset, the first vLRU is set to the maximum value and all the other vLRUs are reset. If the TAG is found, the highest confidence C is identified. If that confidence is in unpredictable state, the load is normally executed, without value prediction. If the confidence is in predictable state, its associated value V is predicted as being the result of the load. This predicted value is forwarded to the dependent instructions from the reservation stations that will be speculatively executed. After the normal execution of the load, the value of the real result is used to update the LVPT entry. If it is a new value, it will replace the least recently used value of that entry (the value having the lowest vLRU) and all the vLRUs and confidences are correspondingly updated. Otherwise, only the vLRUs and the confidences are updated. In the case of misprediction, a recovery process is also necessary and the dependent instructions executed with wrong values are squashed and re-executed with the correct values (selective re-issue).

4. SIMULATION METHODOLOGY

In this study we have used the Sniper 6.1 multicore simulator [1] and the large datasets of the Splash-2 suite of parallel benchmarks [14] which are characterized as follows: *Barnes* simulates in three dimensions a system of bodies; *Cholesky* operates matrix factorization; *Fmm* simulates in two dimensions a system of bodies; *Lu* factors a dense matrix; *Ocean* studies large-scale ocean movements; *Raytrace* renders a three-dimensional scene using ray tracing; *Water* evaluates forces and potentials in a system of water molecules.

The simulations have been run on a computer with Intel Core 2 CPU at 2.4 GHz and a DRAM of 2 GB, under Fedora 22 (kernel 4.0.8-300). The simulated microarchitecture is Intel Nehalem and can include between 1 and 16 cores configured to run at 2.66 GHz. The baseline configuration is presented in Table 1.

Table 1

Parameters of the simulated architecture interacting with the SLVP

DL1 / IL1 cache	Size	16 KB
	Block size	64 B
	Associativity	4
	Latency	3 cycles
L2 cache	Size	256 KB
	Block size	64 B
	Associativity	8
	Latency	9 cycles
L3 cache	Size	8192 KB
	Block size	64 B
	Associativity	16
	Latency	35 cycles
Memory	Latency	175 cycles
SLVP	Latency	1 cycle
	Recovery	7 cycles

The latencies have been determined using Membench in [2] and configured in *gainestown.cfg*. The cache sizes and associativity degrees have been configured in *nehalem.cfg*. The L3 cache is shared among cores.

Since the Instructions Per Cycle (IPC) metric of a certain core can be computed as the number of instructions executed by that core divided to the number of CPU cycles, we can define the IPC of a multicore architecture, on a certain benchmark, as the total number of instructions executed by all the cores divided to the longest cycle time among cores:

$$\mathrm{IPC} = \frac{\sum_{i=1}^{N} I_i}{\max_{1 \le k \le N} \{C_k\}}, \tag{1}$$

where I_i is the number of instructions executed by the core i, N represents the number of cores and C_k is the execution time, in cycles, for the core k. The speedup of a multicore architecture extended with value prediction (VP), with respect to the baseline multicore architecture (B), is:

$$S_{VP}(N) = \frac{\mathrm{IPC}_{VP}}{\mathrm{IPC}_{B}} \tag{2}$$

Since the total number of dynamic instructions executed by these two compared architectures is approximately the same (the slight differences can be neglected), the speedup can be computed as:

$$S_{VP}(N) = \frac{\max_{1 \le r \le N} \{C_{B_r}\}}{\max_{1 \le q \le N} \{C_{VP_q}\}}. \tag{3}$$

Finally, the relative speedup of a multicore architecture with value prediction can be computed as:

$$\mathrm{RS_{VP}}(N) = \frac{\max_{1 \le r \le N} \{C_{B_r}\} - \max_{1 \le q \le N} \{C_{VP_q}\}}{\max_{1 \le r \le N} \{C_{B_r}\}} \cdot 100 \quad [\%]. \tag{4}$$

The energy consumption, expressed in J, can be determined using the following formula:

$$E = \frac{P \cdot C}{f_{CPU}}, \tag{5}$$

where P represents the total power consumption measured in W (provided, in our case, by the McPAT simulator), f_{CPU} represents the frequency of the simulated microprocessor in Hz and C represents the total number of execution cycles. The relative energy reduction is given by the following formula:

$$E_{reduction} = \frac{E_B - E_{VP}}{E_B} \cdot 100 \quad [\%], \tag{6}$$

where, E_B and E_{VP} are the energy consumptions of the baseline and our improved architectures, respectively. Obviously, a positive value of $E_{reduction}$ means an improved energy consumption.

We will also use Pearson's correlation in order to determine what is influencing the speedup in a SLVP-based multicore architecture. Thus, we will analyse the correlation between the relative speedup and the DL1 miss rate, the prediction rate and the prediction accuracy, respectively. The prediction rate is the percentage of predicted critical loads divided by the total number of critical loads. The prediction accuracy is the percentage of correctly predicted critical loads divided by the number of predicted critical loads.

5. EXPERIMENTAL RESULTS

We started our evaluations by analyzing the LVPT's parameters. For doing this, we used a dual core architecture ($N = 2$) with the configuration presented in Table 1.

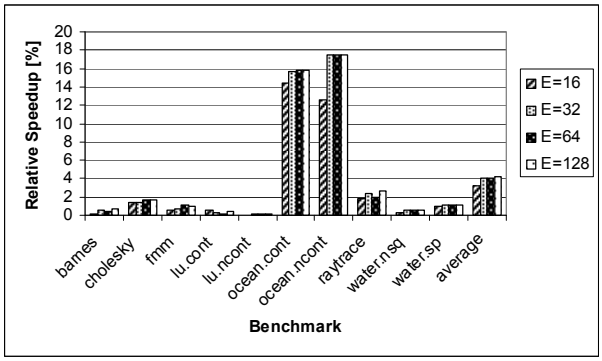

Fig. 3 – Varying the number of LVPT entries.

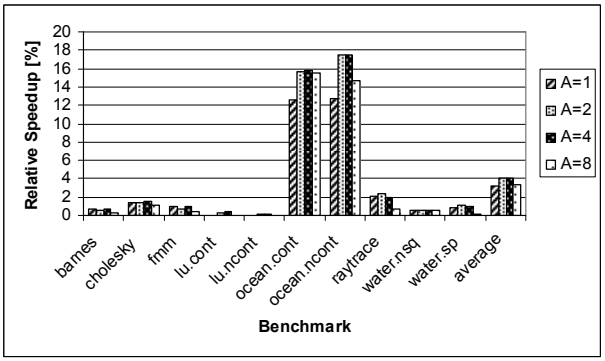

Fig. 4 – Varying the LVPT associativity.

For the LVPT we fixed the associativity to 2, the history (the number of stored values per entry, denoted H) to 2, we have also considered an 1/2 confidence automaton and we have varied the number of LVPT entries (*E*). This initial configuration was chosen after some apriori laborious simulations. Figure 3 illustrates the relative speedup obtained over the baseline architecture using different LVPT sizes. The relative speedup has been computed using formula (4). As Fig. 3 shows, the optimal number of LVPT entries is 32. Over this size the performance improvement is insignificant. On the most of the benchmarks the relative speedup was less than 2%, but on the *ocean.cont* and *ocean.ncont* benchmarks it was over 15% (see Table 2 and the explanation presented below it). Thus, the average relative speedup, obtained with the optimal LVPT size ($E = 32$) was 4.01%.

The next analyzed parameter is the LVPT associativity (*A*). Thus, we have fixed the size of the LVPT to the optimal 32 entries obtained above, the load values' history to 2, we have used 1/2 confidence automata and we have varied the LVPT associativity degree. Here are necessary some comments. Of course that this "hill-climbing" optimization method is not ideal, thus it can't find the global optimum. On the other hand, the multi-objective automatic optimization problem is a NP-hard problem in this case, as we have already shown in [3]. Such complex optimization problems are solved using heuristic algorithms, in an approximate manner. It would be the main aim of another dedicated work. Thus, in this paper we'll assume a manual optimization method. Figure 4 depicts the relative speedup over the baseline architecture using different LVPT associativity degrees. As the evaluations show, the optimal associativity is 2.

Next we have evaluated the impact of the value history's length over the relative speedup, using a LVPT with 32 entries, an associativity degree of 2 and also 1/2 confidence automata. As Fig. 5 shows, the performance is quite invariant to history length's changes. Therefore we have chosen to continue the experiments with just one stored value per LVPT entry ($H = 1$).

We have continued our evaluations by analyzing different confidence automata, fixing the LVPT to 32 entries, associativity degree of 2, and values' history of 1, as we previously obtained through the already presented simulations. Figure 6 illustrates that the optimal confidence automaton is a three states one, using one predictable and two unpredictable states (denoted 1/2).

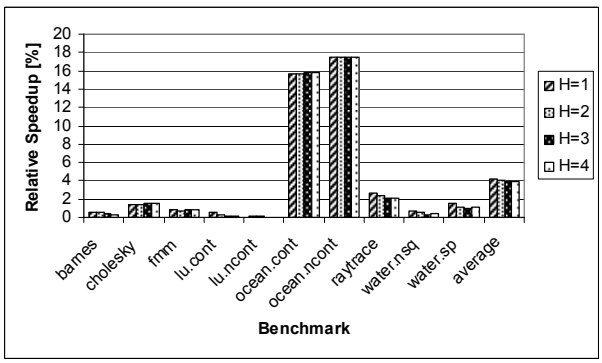

Fig. 5 – Varying the LVPT history.

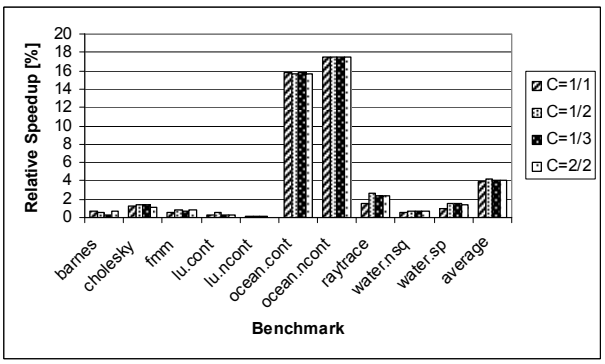

Fig. 6 – Varying the confidence automata.

Table 2

Analyzing the quasi-optimal LVPT in a dual core architecture with 16 KB DL1 cache

Benchmarks	Loads Miss DL1 [%]	Prediction Rate [%]	Prediction Accuracy [%]	Relative Speedup [%]
barnes	4.98	3.03	82.07	0.51
cholesky	3.59	0.45	75.30	1.38
Fmm	1.69	0.42	89.87	0.88
lu.cont	0.99	0.00	84.32	0.5
lu.ncont	12.57	0.02	99.79	0.09
ocean.cont	22.92	5.09	99.12	15.72
ocean.ncont	22.12	5.28	99.09	17.53
raytrace	6.01	1.70	87.00	2.59
water.nsq	1.39	0.31	88.25	0.64
water.sp	2.10	0.65	66.88	1.55
average	**7.84**	**1.70**	**87.17**	**4.14**

We have evaluated also the relative speedup by predicting all the critical loads without a classification into predictable and unpredictable. The performance in that case was significantly lower. Moreover, on most of the benchmarks we have observed significant performance degradation even related to the baseline architecture. Consequently, the role of the confidence automata is essential for better prediction accuracies (and thus fewer recoveries) which constitute the basis for the speedup of such speculative architectures.

Further we present a detailed analysis of the current quasi-optimal LVPT configuration ($E = 32$, $A = 2$, $H = 1$, $C = 1/2$) in a dual core architecture with 16 KB DL1 cache. Table 2 shows the DL1 miss rate, the prediction rate, the prediction accuracy and also the relative speedup for each benchmark. As Table 2 shows, we have calculated a strong positive correlation between the relative speedup and the percentage of loads with miss in the DL1 cache and the load value prediction rate, respectively, and there is a lower correlation with the prediction accuracy. More precisely, the Pearson correlation coefficient between the relative speedup and the percentage of loads with miss in DL1 cache is 0.9. The same correlation coefficient between the relative speedup and the load value prediction rate is 0.9, and it became only 0.54 by correlating the relative speedup with the load value prediction accuracy. Thus, our SLVP unit is the most efficient in architectures and benchmarks that involve high percentages of loads with miss in the DL1 cache (thus high DL1 miss rates) and high load value prediction rates. That is why we have obtained good relative speedups on some benchmarks (like *ocean.cont* and *ocean.ncont*), but low speedups on other benchmarks (like *lu.ncont*). In other words, high prediction rates involve a high number of dynamic loads on shared variables. This high number means that there is a significant communication between the threads belonging to a certain task (Splash-2 benchmark in our case). This explanation is validated by the fact that particularly the two *Ocean* benchmarks contain a lot of communication compared with the other benchmarks that are more computation-intensive, without so many intrinsic communication processes. Table 2 also shows some very low prediction rates which, in our opinion, might be the consequence of low load value localities.

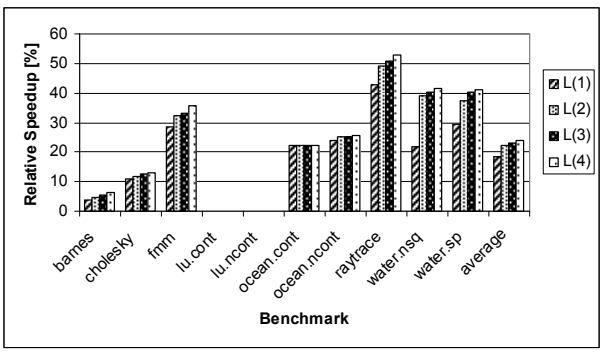

Fig. 7 – The value localities of the critical loads from the Splash-2 benchmarks.

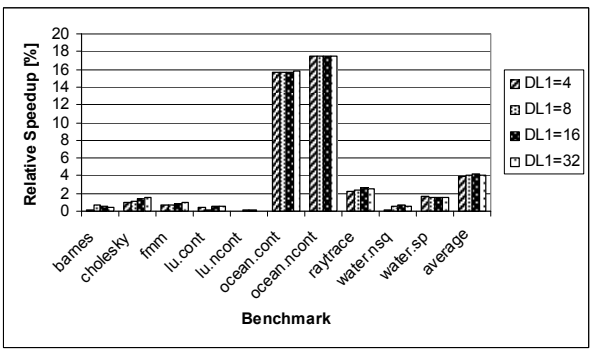

Fig. 8 – Varying the DL1 cache size.

Lipasti et al. [9] have first introduced the value locality concept as "the likelihood of the recurrence of a previously-seen value within a storage location". Thus, the value locality represents the upper value

predictability limit. Significant value localities involve significant prediction accuracies, and vice-versa. Measurements using the SPEC'95 benchmarks have shown that value locality on load instructions is about 50% using a history of one and 80% using a history of 16 previous instances. We have measured the value locality of the critical load instructions from the Splash-2 parallel benchmarks. In Figure 7, L(H) denotes the value locality of the critical loads considering their last H distinct data values. As Figure 7 illustrates, the average value locality of the critical loads from the Splash-2 benchmarks is only 18.31% (less than the 50% from SPEC'95) considering one (the last) data value. The average value locality is very low with higher history lengths, too: 22.21% with 2, 23.04% with 3 and 23.86% with 4 data values, respectively. The value locality is less than 0.2% on the *Lu* benchmarks. This low value locality is an intrinsic characteristic of the critical loads from the Splash-2 parallel benchmarks and explains the low prediction rates presented in Table 2. The constant locality of the *Ocean* benchmarks, whose prediction rates were the highest, explains why their relative speedups are constant along different history lengths. Thus, the very low prediction rates presented in Table 2 are consequences of the low value localities of critical loads.

Table 3

Power and energy consumption evaluation

Benchmarks	Baseline power [W]	SLVP-based power [W]	Baseline energy consumption [J]	SLVP-based energy consumption [J]	Relative energy reduction [%]
barnes	25.4	25.8	20.22	20.43	-1.06
cholesky	40.54	41.29	2.62	2.63	-0.45
fmm	42.54	43.07	17.29	17.35	-0.35
lu.cont	47.09	47.63	7.73	7.78	-0.64
lu.ncont	43.23	43.56	7.81	7.86	-0.67
ocean.cont	24.8	27.1	63.18	58.18	7.90
ocean.ncont	25.9	28.76	65.42	59.91	8.42
raytrace	24.91	25.49	10.34	10.31	0.33
water.nsq	33.95	34.38	25.78	25.94	-0.62
water.sp	35.1	35.78	7.55	7.58	-0.36
average	34.35	35.29	22.79	21.80	1.25

Next we have analyzed the influence of the DL1 cache size over the relative speedup. Thus, we have varied this parameter in both the baseline and the SLVP-based dual core architectures. We have considered the previously established LVPT parameters: 32 entries, associativity degree of 2, value history of 1 and 1/2 confidence automata. As Figure 8 depicts, the average relative speedup is the same, about 4%, on all the evaluated DL1 cache sizes. Therefore, we will apply the next evaluations with the initial DL1 cache size of 16 KB. Next we have evaluated the power consumption, the energy consumption (based on (5)) for the baseline and the SLVP-based (E = 32, A = 2, H = 1, C = 1/2) architectures, both with DL1 = 16 KB and N = 2 cores. We have also determined the relative energy reduction using formula (6), as a measure of energy efficiency. The power consumption is slightly higher with SLVP due to the new LVPT unit, but the energy consumption is lower (by 1.25% at average) due to the lower processing time. Next we have evaluated the relative speedup by varying the number of cores simultaneously in the baseline and in the SLVP-based architectures.

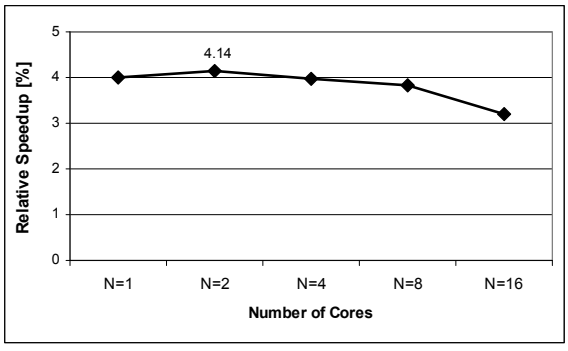

Fig. 9 – Varying the number of cores.

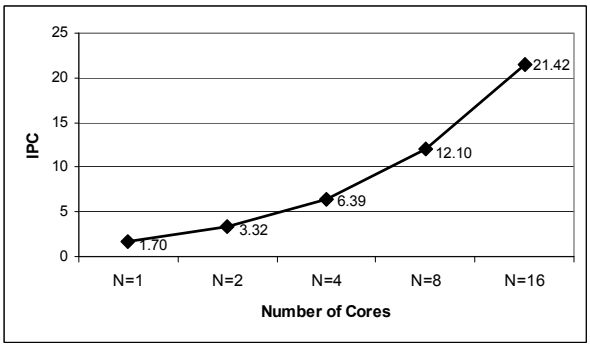

Fig. 10 – The IPC of SLVP-based architectures.

Figure 9 presents the relative speedup averaged on all the benchmarks. The highest performance gains were obtained for less cores. As the number of cores grows, the relative speedup is lower. We have also evaluated the IPC of the SLVP-based architectures by varying the number of cores. Figure 10 presents the IPC averaged on all the benchmarks. As the number of cores increases the IPC grows only slightly non-linearly.

6. CONCLUSIONS

In this work we have extended the Sniper multicore simulator with selective load value prediction capabilities. Since the technique is selectively applied only on critical load instructions, a small and fast LVPT is enough to exploit the benefits in terms of performance and energy consumption. As far as we know, we are the first researchers who integrated the SLVP technique into a speculative multicore architecture and evaluated the speedup and the energy consumption reduction on native concurrent applications.

By applying a manual design space exploration of the proposed speculative multicore architecture on the Splash-2 parallel benchmarks, we concluded that the optimal SLVP-based multicore configuration is using a DL1 cache of 16 KB and a 2-way associative LVPT (A = 2) with 32 entries (E = 32), only the last value (H = 1) and confidence automata having one unpredictable and two predictable states (C = 1/2). The average relative speedup was about 4%, with a maximum of 17.53%. We have varied the number of cores and the highest performance gains were obtained for fewer cores within the simulated architecture. We have also observed a strong positive correlation between the relative speedup and the percentage of loads with miss in the DL1 cache and the load value prediction rate, respectively. Therefore, the SLVP technique is the most efficient in microarchitectures and benchmarks that involve high percentages of loads with miss in the DL1 cache and high load value prediction rates. The power consumption was slightly higher with SLVP due to the new LVPT unit, but the energy consumption was lower by 1.25% due to the lower processing time.

REFERENCES

1. T.E. CARLSON, W. HEIRMAN, S. EYERMAN, I. HUR, L. EECKHOUT, *An Evaluation of High-Level Mechanistic Core Models*, ACM Transactions on Architecture and Code Optimization (TACO), **11**, *3*, October 2014.
2. T.E. CARLSON, W. HEIRMAN, *The Sniper User Manual*, November 2013.
3. R. CHIS, L. VINTAN, *Multi-Objective Hardware-Software Co-Optimization for the SNIPER Multi-Core Simulator*, 10th International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, September 2014, pp. 3-9.
4. A. FLOREA, C. BUDULECI, R. CHIS, A. GELLERT, L. VINTAN, *Enhancing the Sniper Simulator with Thermal Measurement*, The 18th International Conference on System Theory, Control and Computing, Sinaia, October 2014.
5. A. GELLERT, A. FLOREA, L. VINTAN, *Exploiting Selective Instruction Reuse and Value Prediction in a Superscalar Architecture*, Journal of Systems Architecture, Elsevier, **55**, *3*, pp. 188-195, 2009.
6. A. GELLERT, G. PALERMO, V. ZACCARIA, A. FLOREA, L. VINTAN, C. SILVANO, *Energy-Performance Design Space Exploration in SMT Architectures Exploiting Selective Load Value Predictions*, International Conference on Design, Automation and Test in Europe (DATE 2010), March 2010, Dresden, Germany, pp. 271-274.
7. A. GELLERT, H. CALBOREAN, L. VINTAN, A. FLOREA, *Multi-Objective Optimizations for a Superscalar Architecture with Selective Value Prediction*, IET Computers & Digital Techniques, **6**, *4*, pp. 205-213, July 2012.
8. A. GELLERT, A. FLOREA, U. FIORE, P. ZANETTI, L. VINTAN, *Performance and Energy Optimisation in CPUs through Fuzzy Knowledge Representation*, Information Sciences, Accepted in March 2018.
9. M.H. LIPASTI, C.B. WILKERSON, J.P. SHEN, *Value Locality and Load Value Prediction*, The 7th International Conference on Architectural Support for Programming Languages and Operating Systems, October 1996, pp. 138-147.
10. M.M.K. MARTIN, D.J. SORIN, H.W. CAIN, M.D. HILL, M.H. LIPASTI, *Correctly Implementing Value Prediction in Microprocessors that Support Multithreading or Multiprocessing*, 34th Annual ACM/IEEE International Symposium on Microarchitecture, Austin, Texas, December 2001.
11. A. PERAIS, A. SEZNEC, *Practical Data Value Speculation for Future High-End Processors*, 20th International Symposium on High Performance Computer ArchitectureOrlando, FL, USA, February 2014, pp. 428-439.
12. A. PERAIS, A. SEZNEC, *EOLE: Paving the Way for an Effective Implementation of Value Prediction*, 41st Annual International Symposium on Computer Architecture, Minneapolis, MN, USA, June 2014, pp. 481-492.
13. A. PERAIS, A. SEZNEC, *BeBoP: A Cost Effective Predictor Infrastructure for Superscalar Value Prediction*, 21st International Symposium on High Performance Computer Architecture, San Francisco, CA, USA, February 2015, pp. 13-25.
14. S.C. WOO, M. OHARA, E. TORRIE, J.P. SINGH, A. GUPTA, *The SPLASH-2 Programs: Characterization and Methodological Considerations*, 22nd International Symposium on Computer Architecture, Italy, June 1995, pp. 24-36.

Received January 8, 2018