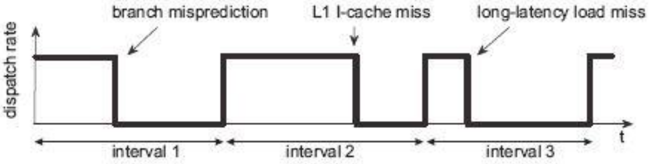
WP1: High Performance Architectures towards PetaFLOPS based on Multi-Core Processors

1. Main Activities and Results

***Task 1.1**: The Node Architecture of the PetaFLOPS supercomputer.*

According to the activities defined in task 1.1series of investigations on the evaluation of the performance of heterogeneous microprocessor architectures using user and system perspective metrics were performed. The tests have been conducted using a new simulation technique – Interval simulation, and a computer architecture simulator which is based on this technique. The evaluation is done by using multi-programmed multithreaded workloads constructed by using the benchmarks included in the two most popular free benchmark suites available – PARSEC, developed and maintained by Princeton University, and SPLASH2 – developed and maintained by The University of Stanford. The experiments compare the heterogeneous architecture models against one of the most wide-spread and popular design choices today – the Chip Multi-processors (CMPs). The heterogeneous design combines the performance of multi-core with the simplicity of many-core into a new diverse architecture. Heterogeneousy can be explored in different aspects. It can be expressed in the context of various computing elements with distinct Instruction Set Architectures (ISAs) cooperating to the completion of a common task. An example of such a case is the use of GPUs together with CPUs. In this case the computing elements themselves are homogeneous but the system, as a whole, is heterogeneous.

Interval simulation is a recently proposed method for simulation of multi-core and many-core processor architectures. It is based on the mechanistic model for out-of-order processors, proposed by Eyermanand al. Using interval simulation the process of performance analysis of the architecture in question can be significantly sped up. The interval model is based on the fact that the continuous flow of currently executing instructions is only interrupted by miss events. Each one of these events is characterized with a specific duration.

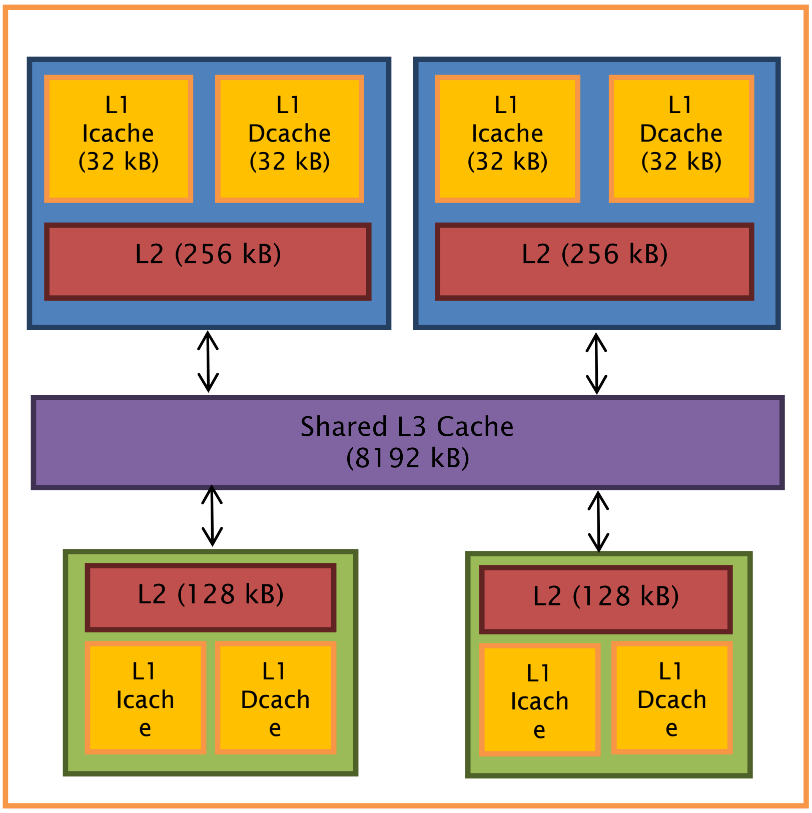


Types of miss events are: branch miss-prediction, L1 cache miss, long-latency load miss etc. Branch prediction, memory hierarchy, cache coherency and interconnection network define miss events. The characteristics of the former define the duration of the latter.

Sniper is a multi-core architecture simulator based on the interval method which models the timing for the individual cores. The simulator maintains a `window' of instructions for each simulated core. This window of instructions in reality corresponds to the reorder buffer of a superscalar out-of-order processor. Miss events, as described above, are determined with the assistance of the window and attention is paid to overlapping misses of different duration. Instructions are fed into the window tail. The progress of each core is determined by considering the instruction at the head of the window.

Both used suites offer a variety of benchmarks which are mostly multithreaded and cover various areas of application. For example, the PARSEC benchmark suite features c-anneal – a benchmark which simulates annealing, fluid-animate – fluid motion simulation, ray-trace – tracing a ray of light through a scene and more. The SPLASH2 benchmark suite features programs such as ocean – particle simulation, LU – matrix factorization, Blacks-Scholes – stock exchange analysis based on the Blacks-Scholes algorithm etc. The workloads used in our experiments are comprised of all possible combinations of benchmarks from both suites. This has been done in order to gather enough data for a statistic estimation of the performance of the examined configurations since scheduling for heterogeneous systems is not implemented in this simulator.

Experimental results have been collected using the Sniper Multi-Core simulator running on a 2-core Intel i3 processor with two more virtual cores, 8 Gb of RAM, 3 Mb LLC with Linux Ubuntu 12.04 installed. All possible combinations of workloads including benchmarks from both suites have been constructed to avoid bias that might be presented by the lack of a scheduling policy.
A heterogeneous configuration has been built based on the balanced processor design suggested by Eyerman et al.



The experimental results are presented below. Trend-lines represent the moving average for each data series. The series ***avg*** represents the average of all trend-line and acts as a base for comparison.

Table

Heterogeneous base (het-base)	
L1 I-cache L1 D-cache L2 Cache L3 Cache	16,16,32,32 16,16,32,32 128,128,256,256 8192
Heterogeneous 64k L1 (het-64k)	
L1 I-cache L1 D-cache L2 Cache L3 Cache	32,32,32,32 32,32,32,32 256,256,256,256 8192
Heterogeneous 512k L2 (het-512k)	
L1 I-cache L1 D-cache L2 Cache L3 Cache	32,32,32,32 32,32,32,32 512,512,256,256 8192
Homogeneous (gen)	
L1 I-cache L1 D-cache L2 Cache L3 Cache	32 32 256 8192

One can easily see that the greater part of the moving average for the homogeneous configuration is found above the absolute average for all series, i.e. parallel execution overhead using this configuration and the above mentioned workloads is to a degree greater than the overhead incurred when executing these workloads using the other configurations. Contrary to this trend is the trend observed in the ***het-base*** series. It is mostly concentrated below the absolute average. It is difficult to compare the other two configurations since their observed performance is similar to a great extent. We conclude that for most workloads of the base heterogeneous configuration offers the best or close to the best performance which makes it the configuration of choice.

The second experimental setup is focused on “issue width“variance. Optimal results in this experiment are observed in the tests with the base heterogeneous configuration and the homogeneous configuration with the base heterogeneous configuration exhibiting the lowest ANTT and thus best performance.

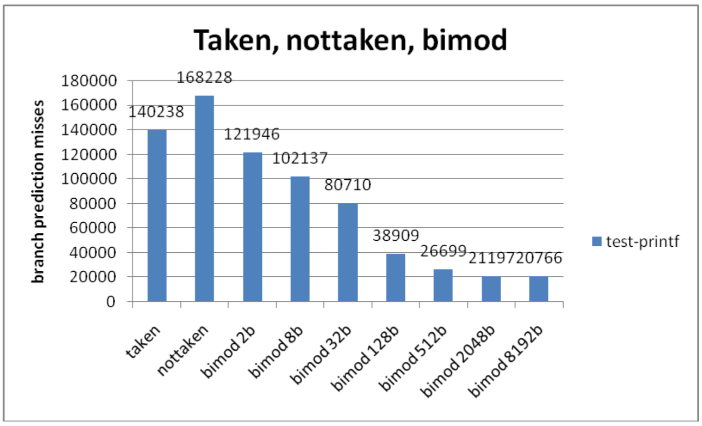
The third experimental setup is focused on ROB size variance. In this scenario the heterogeneous configuration with ROB size 128 and the homogeneous configuration display very similar results close to the absolute average. This is expected since both configurations employ equally sized ROBs. The heterogeneous configuration with the largest ROB – 256, is the slowest with respect to performance. This might be explained with the ROB being a memory structure – too large a size inevitably results in performance degradation. The base heterogeneous configuration again manifests best performance for a prevailing amount of workloads. This experiment clearly shows the well-known truth that larger memory structures do not guarantee better performance as microprocessor design requires subtle balance between all components.

In all three experiments that have been conducted, the base heterogeneous configuration which has been built using the principles for a balanced processor design manifested optimal or close to the optimal performance. This paper strived to demonstrate that heterogeneous microprocessor architectures are a feasible design and are able to provide adequate performance for different types of workloads. Therefore, heterogeneous design approach can be considered a reasonable alternative to homogeneous. It is worth to mention that whenever greater performance per Watt is necessary heterogeneous design is a clear winner since simpler cores yield smaller die size, at the very least, and overall lower energy consumption.

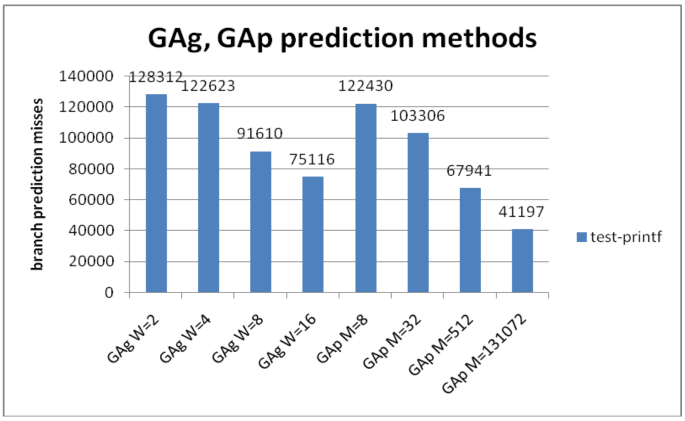
Another important field of research in WP1 was concentrated on the basic static and dynamic methods for branch prediction. The purpose of the research is to determine which of known branch prediction methods give the best results.

Dynamics methods which are used in the modern processors attempt to predict if the branch will be taken or not based on a history of previous branches. Dynamics branch predictor includes: Two bit counter, Two level branch predictor, GAg Predictor, GAp / Gas Predictor, PAg Predictor, G-share, Branch prediction with perceptron.

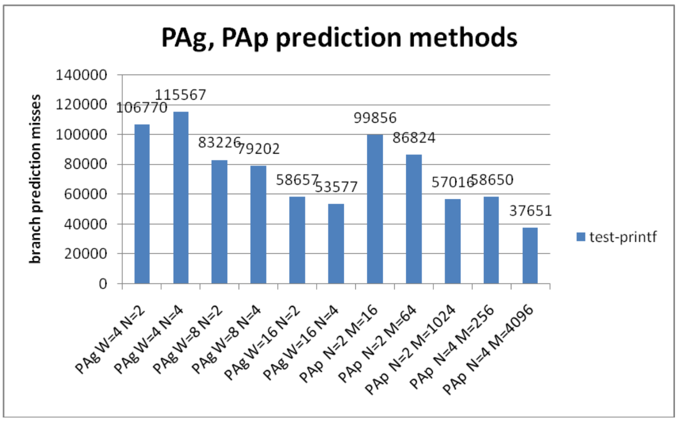
To study the performance of different methods for predicting we have used Simple Scalar and Advanced Branch Prediction Simulator. *SimpleScalar* simulation technique was created by Todd Austin in 1996 and received publicity when it is modified to simulate multithreaded processors and multiprocessors. The basic architecture of the simulator is based on superscalar processor MIPS R10000. Its main advantage is that with minimal effort can change software and hardware architecture to study. *Advanced Branch prediction simulator* is a trace simulator that allows you to explore issues related to the provision of transitions. Predictor is also the detector difficult to predict transitions, interactive and easy to use.



Static branch prediction methods shown when it is assumed that the transition is always taken the number of branch prediction miss is smaller than when it is assumed that you will not get what is due to the fact that the branches are unconditional and conditional and unconditional branches always run. Bimodal predictor is the first of methods for dynamic forecasting; it gives better results than static methods as we may see. In this method we study the changes in the number of branch prediction misses depend of resizing of history table. As a result of the research showed that with increasing size to 2Kb for branch prediction is reduce the number of misses. After that, the reduction of number of falling with increasing size of the table is negligible. And as we may see bimodal predictor gives much better results than the static methods.



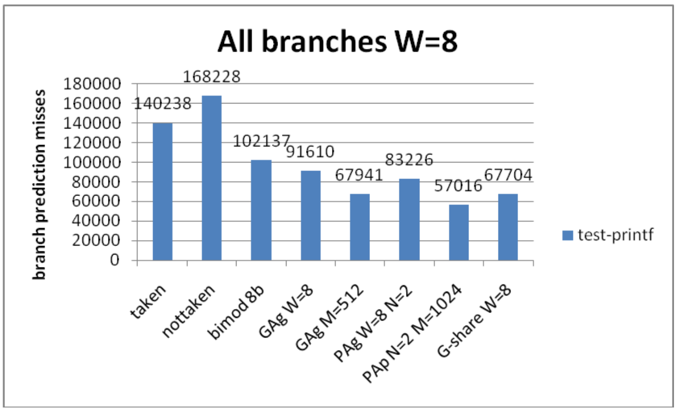
The Figure shows first two branch prediction methods GAg and GAp. As we can see GAp give us better results than GAg method, because in GAg method we have global register and global branch history table ie in this table we save history for all branches, while in GAp we have own history table for each address.

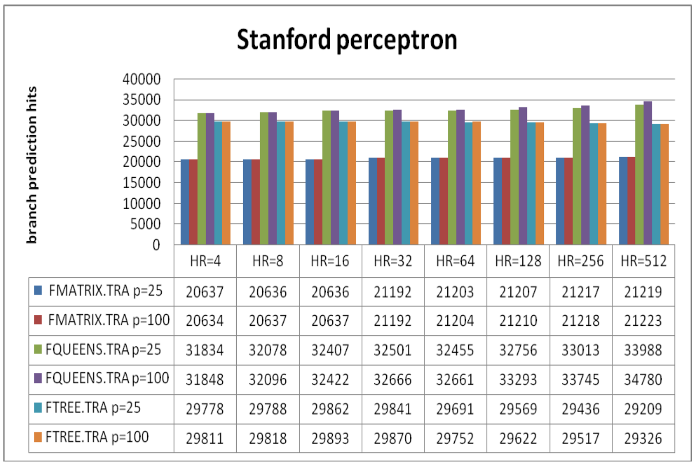


In PAg prediction method register containing information about last k branches for each address and branch history table is common. In Pap method we have individual register and branch history table for every address. As we see in the Figure the second method with own register and branch history table is better of the two, as in the case have own table for each address, i.e. we don’t need to look for in a common table as in the first method. In this method access must be done much faster.

In the next Figure we may see a comparison of different methods for branch prediction as to them was added Gshare method is relatively new and gives notice as one of the best results together PAp.

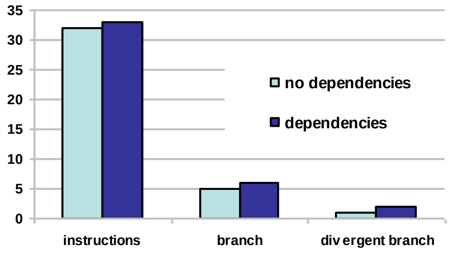
In the last Figure one of the modern methods for predicting using perceptron is considered. In Stanford tests with increasing size of history register we see increasing of the number of hits, while when we increase the number of perceptrons and we use large register results are very good.

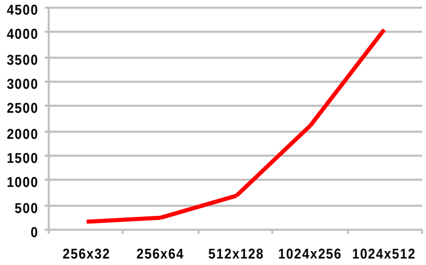


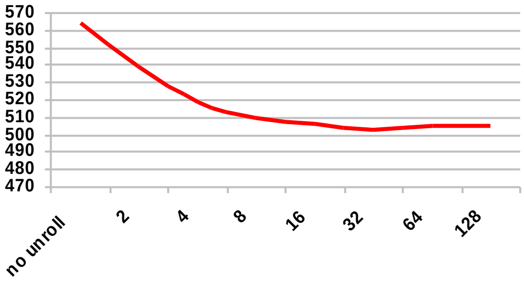


	HR=4	HR=8	HR=16	HR=32	HR=64	HR=128	HR=256	HR=512
■ FMATRIX.TRA p=25	20637	20636	20636	21192	21203	21207	21217	21219
■ FMATRIX.TRA p=100	20634	20637	20637	21192	21204	21210	21218	21223
■ FQUEENS.TRA p=25	31834	32078	32407	32501	32455	32756	33013	33988
■ FQUEENS.TRA p=100	31848	32096	32422	32666	32661	33293	33745	34780
■ FTREE.TRA p=25	29778	29788	29862	29841	29691	29569	29436	29209
■ FTREE.TRA p=100	29811	29818	29893	29870	29752	29622	29517	29326

The third field of research is the evaluation of data dependencies in multithreaded applications. Modern heterogeneous architectures include advanced computing devices with great multi-threading capabilities. Unfortunately not all applications utilize all these capabilities due to the specific or complexity of algorithms. One of the obstacles which come from the natural execution of algorithms is "data dependencies". The main objective of the research is to provide results from the area of data dependencies in multi-threaded parallel applications and to show influence of these dependencies on the particular applications. In the next pictures we show some of the results, achieved in experiments with three fields of applications: Parallel equation of sine wave, CFD simulation and N-body simulation. Two architectural resources were used: Intel Core i7 and two graphical accelerators NVIDIA model GTX295 with 480 CUDA cores, and Intel Core i3 and graphical accelerators AMD Radeon HD4300 with 80 Stream cores.





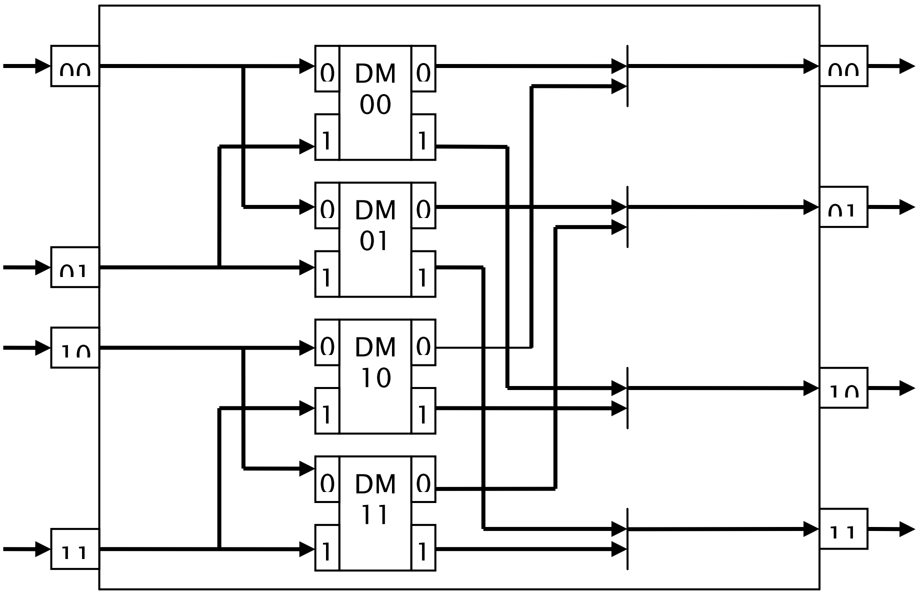


The team working on this task includes V. Lazarov, M. Marinova, D. Gurnevski, G. Dimova, M. Grafova, and P. Dimitrova. The results achieved in task 1.1 are presented in publications [M_12], [MLD_12a], [MGL_13], [LGM_13a], [M_13a], [DML_13a], and [MLD_13a].

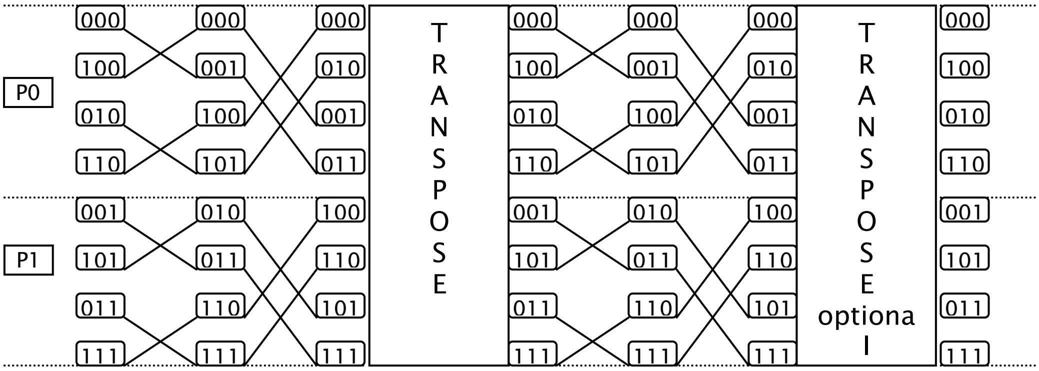
***Task 1.2*: Hierarchy architecture of PetaFlops supercomputer.**

In order to realize the activities of the task the main efforts were concentrated in the definition of the optimal structure for networking the different nodes of the supercomputer, oriented to different tree architectures. During the research we analyzed the influence of the different structures on the global throughput and bandwidth and the mutual possible blockages between the nodes. Special attention was dedicated on the problem of latency in long distance accesses to information. The analyses are performed analytically and through computer simulations. Variants of parallel architectures for FFT are mainly studied. For the Fast Fourier Transform algorithm (FFT) are known two types of natural architectures – the direct and the indirect hypercube. The development of "Indirect Hypercube" concept still unfinished. Fast Hartley transform (FHT)/Real-valued Fast Fourier transform (RFFT) algorithms are important. These types of algorithms have irregular computational structure. Aim of the research is to present a further development of the concept "Indirect Hypercube". Described are two types of SIMD array architectures (radix-2 and radix-4).

The structure of the network is shown on the figure.



The main direction was to analyze the possibilities of the Transpose algorithms. The study proves that transpose modules used by FFT transpose algorithms with small modifications can be used by RFFT/FHT transpose algorithms.



On the figure we have 16 pt. ordered RFFT/FHT on 2 processors (2D transpose algorithm). The proposed method combines the advantages of the transposition method for FFT algorithms (minimization of communication and simplicity of the approach) with the advantages of the RFFT/FHT algorithms (lowering twice memory and operational requirements).

The team working on this task includes V. Lazarov and Ph. Philipov, and some of the obtained results are published in [F_12], [LF_12], [FL_13a].

<u>Task 1.3</u>: Architecture of I/O subsystem in the supercomputer node.

In the frame of this task the efforts are concentrated in defining the kind of input-output ports, I/O processors and the distribution of the functions between the processing and communication parts of the module. Adopted are the simulation techniques and means for the investigations, and also the specialized hardware for speeding the experiments. The studies are under realization and the results will be announced soon.

2. Publications with acknowledgments to the project ДЦВП 02/1

a) published:

1. [MGL_13] M. Marinova, M. Grafova, V. Lazarov. Simulation tools for evaluating of behavior of different techniques for branch prediction in modern processor architectures. Proceedings of ICACSE 2013: International Conference on Applied Computer Science and Engineering, Barcelona, WASET, Issue 74, p.894-p898, February 2013.
2. [M_12] M. Marinova. Evaluation of Data Dependencies in Multithreading Applications. Proceedings of the Second International Conference “Education, Science, Innovations” (ESI’12), European Politechnical University, ISSN 1314-5711, pp. 193-196, Pernik, June 9-10.2012.
3. [F_12] Ph. Philipov. A Parallel Real-Valued FFT Transpose Algorithm. Proceedings of the Second International Conference “Education, Science, Innovations” (ESI’12), European Politechnical University, ISSN 1314-5711, pp. 181-191, Pernik, June 9-10.2012.
4. [LF_12] V. Lazarov, Ph. Philipov. Investigation of the *Indirect Hypercube* as a Natural Architecture for FFT/RFFT/FHT Parallel Algorithms of Transpose Type. Proceedings of the Second International Conference “Education, Science, Innovations” (ESI’12), European Politechnical University, ISSN 1314-5711, pp. 235-254, Pernik, June 9-10.2012.

b) accepted:

1. [MLD_12a] M. Marinova, V. Lazarov, G. Dimova. Performance Evaluation of Heterogeneous Microprocessor Architectures. Journal „Information Technologies and Control“.
2. [FL_13a] F. Filipov, V. Lazarov. Indirect Hypercube as a Natural Architecture for Sine and Cosine Realization. International Conference “Education, Science, Innovation” (ESI’13), European Politechnical University, Pernik, 2013.
3. [M_13a] M. Marinova. Evaluation of Thread Architectures using Interval Simulation. International Conference “Education, Science, Innovation” (ESI’13), European Politechnical University, Pernik, 2013.
4. [DML_13a] Petya Dimitrova, Maria Marinova, Vladimir Lazarov. Trace-Driven Similator for research of cache memories parameters. International Conference, „45 Years TU-

Varna“, September, 2013.

5. [MLD_13a] Maria Marinova, Vladimir Lazarov, Petia Dimitrova. Simulation tools for evaluating the performance of cache systems for educational purposes. International Conference, „45 Years TU-Varna“, September, 2013.
6. [LGM_13a] V. Lazarov, D. Gurnevski, M. Marinova. Evaluation of Instruction Dependencies in Multithreaded Application executed on heterogeneous architectures. Списание „Компютърни системи и технологии“ на ТУ-София.

3. Others

[1] Organizational activities: participation in Operational Board Meetings (V. Lazarov) and in the operational group for organizing the work in WP1, WP3 and WP4 (V. Lazarov)

[2] Participation in Project Seminar in Triavna – V. Lazarov, M. Marinova, D. Gurnevski, M. Grafova.

[3] Introduction of some project studies and results into course content of “High Performance Architectures” for BAS, NBU (New Bulgarian University), and EPU (European Politechnical University) -V. Lazarov, M. Marinova.

[4] Additional activities for dissemination and popularization of the obtained results.