Multi-Objective Hardware-Software Co-Optimization for the SNIPER Multi-Core Simulator

Radu CHIS[1]
[1]Computer Science Department
Technical University
Cluj-Napoca, Romania
radu.chis@gmail.com

Lucian VINTAN[1,2]
[2]Computer Science & Electrical Engineering Department
"Lucian Blaga" University
Sibiu, Romania
lucian.vintan@ulbsibiu.ro

***Abstract*: Modern complex microarchitectures with multicore systems like CPUs, APUs (accelerated processing units) and GPUs require hundreds or thousands of hardware parameters to be fine-tuned to get the best results regarding different objectives like: performance, hardware complexity (integration area), power consumption, temperature, etc. These are only a few of the objectives needed to be taken into consideration when designing a new multicore system. Exploring this huge design space requires special tools like automatic design space exploration frameworks to optimize the hardware parameters. Although the microarchitecture might be very complex, the performance of the applications is also highly dependent on the degree of software optimization. This adds a new challenge to the DSE process. In this paper, using the multi-objective design space exploration tool FADSE, we tried to optimize the hardware and software parameters of the multicore SNIPER simulator running SPLASH-2 benchmarks suite. We optimized the hardware parameters (nr cores, cache sizes, cache associativity, etc.) and software parameters (GCC optimizations, threads, and scheduler) values that have been varied during the DSE process and shown the impact of these parameters on the optimization's multi-objectives (performance, area and power consumption). Furthermore, for the best found Pareto configurations the temperatures will be computed so that in the end we will have a 4-dimensional objective space.**

Keywords—Design Space Exploration, Multi-objective Optimization Algorithms, Sniper Multi-Core Simulator, SPLASH-2 benchmarks

I. INTRODUCTION

Modern systems require more complex and novel processor architectures. Current desktop computers CPUs already have 4-8 complex cores. Accelerated processing units (APUs) combine a CPU with an integrated GPU to deliver performance comparable to dedicated GPUs but at a fraction of the power consumption, while modern phones and tablets have CPUs with 4-8 cores, too. These new architectures bring multiple new research challenges. For example, the hardware parameters of the micro-architecture have to be optimized in parallel with the target applications (hardware software co-optimization). The huge search space created by the processor/compiler/application configurations is not feasible to be entirely evaluated.

With the increase of the number of cores on a chip, not only the performance increases, but also the power consumption and temperature. Due to the fact that the integrated transistors' dimensions decreased from year to year, the integration area of the CPUs didn't really increase. The three objectives - Performance (Instructions per Cycle, IPC), Power Consumption and Area - come into conflict with each other and all of them have to be either maximized or minimized. Optimizing hardware alone might not be sufficient and co-optimizing the workload and the hardware in a hardware-software co-optimization process can lead to much better results. Optimizing the system's performance in a given power, area and energy budget is widely adopted for the design of smartphones, tablets and laptops/desktops.

Evolutionary algorithms have been used to overcome the conflicting objectives. Different heuristics and meta-heuristics have been designed in Automatic Design Space Exploration Tools (ADSE) to find the Pareto front approximation in a feasible amount of time.

One of these ADSE tools is FADSE [3] (Framework for Automatic Design Space Exploration) that has been developed by former Ph. D. student Horia Calborean under the supervision of Prof. Lucian Vintan at the "Lucian Blaga" University of Sibiu. Computation-intensive searches using state of the art evolutionary multi-objective algorithms, guided by the human experience are automatically performed by FADSE as presented in our previous works [1], [2].

In this article we present the automatic design space exploration of the Sniper [4] multicore simulator with FADSE for both hardware (number of cores, cache sizes, cache associativity) and software parameters (GCC optimizations, number of threads and scheduler) using the NSGA-II meta-heuristic from the jMetal library. We will show the impact of these parameters on the objectives we set: CPI (clocks per instruction), Area and Energy. At the end of the DSE process, we also compute the temperatures of the best found configurations, thus having computed a 4-dimensional objective space.

This article is structured as follows: Section 2 provides an overview of the related work. The tools (FADSE and Sniper) are presented in Section 3, while Design Space Exploration and Optimization concepts along with the objectives used and the

parameters for the DSE algorithms are presented in Section 4. The results of our evaluation are shown in Section 5 and finally Section 6 concludes the paper and suggests directions for further work.

II. RELATED WORD

There are a number of tools for the DSE of processor microarchitectures. They use different heuristics to find the optimal solutions for a given hardware architecture. Just to name some of them: FADSE (multi-objective automatic design space exploration framework that is very versatile and uses state-of-the-art multi-objective optimization algorithms), Archexplorer [6] (a Web-based permanent and open design-space exploration framework that lets researchers compare their designs against others), Multicube Explorer [7] (an interactive open-source framework to enable the designer to automatically explore a design space of configurations for a parameterized architecture), Magellan [8] (a Search and Machine Learning-based Framework for Fast Multi-core Design Space Exploration and Optimization), NASA [9] (a Generic Infrastructure for System-level MP-SoC Design Space Exploration), SystemCoDesigner [10] (a novel SystemC-based ESL tool to automatically optimize a hardware/software SoC - System on Chip - implementation with respect to several objectives), HArtes [11] (holistic approach to reconfigurable real-time embedded systems).

Although hardware-software co-design is the design paradigm of choice for embedded system design (SystemCoDesigner and HArtes implement a hw-sw co-design), it is not yet generally employed for performance processor design. In [5] the authors propose a manual DSE for Sniper multicore and demonstrate that increases in performance and energy-efficiency of 1.66x and 1.25x can be obtained by co-optimizing hardware and software.

III. FADSE AND SNIPER

As mentioned before, FADSE is a very versatile ADSE tool, which we easily connected to the Sniper simulator through a connector. This way the DSE process is guided by FADSE, while the simulations for the computations of the different objectives are run in Sniper.

A. FADSE

FADSE is an automatic design space exploration framework mainly targeted to computer systems. It was introduced by Calborean and Vintan in 2010 [3]. The framework uses state-of-the art evolutionary multi-objective optimization algorithms which include: NSGA-II [12], SPEA2 [13], SMPSO [14] and many other. FADSE is written in JAVA so it is platform independent and also is very versatile, because almost any computer system simulator can be connected to it through a dedicated connector. There are already a lot of connectors implemented for mono and multi-core simulators like: Sniper, GAP & GAPtimize, M-SIM3, M-SIM2, M5 and Multi2Sim V2.

FADSE also contains different quality indicators for comparing different algorithms or different runs of the same algorithm, which do or do not require the true Pareto front. Most important of these are the ones that do not require the true Pareto front like Coverage [16], Hypervolume [16] and Two Set Hypervolume Difference [17].

Further details are given in [3] and the source code is publicly available at: http://code.google.com/p/fadse/.

Our developed FADSE tool also can incorporate restriction rules (for example: L2CacheSize should be smaller than L3CacheSize), relations (for example: if no L3 cache size is used, invalidate all parameters for L3 cache size) and, as far as we know, it is the first one using fuzzy logic rules for the representation of computer architecture domain-knowledge. This domain-knowledge was integrated in the mutation genetic operator, with benefits in reducing the search time and improving the solutions' quality, as presented in [1] and [2]. For the further reducing of the search time, distributed evaluations of the individuals are allowed; a database saves the simulation results for future reuse, while checkpointing and error recovery mechanisms oversee the simulations, so that the DSE process is not started again from scratch.

B. Sniper/McPAT

Sniper is an accurate, high-abstraction level parallel multi-core x86 simulator that uses interval simulation. "Interval simulation is a recently proposed simulation paradigm for simulating multi/many-core and multi-processor systems at a higher level of abstraction than current practice of detailed cycle-accurate simulation" [4].

McPAT [19] is a fully-integrated power, area and timing modeling framework. It provides an integrated solution for multi-threaded and multi-core processors and models many types of power dissipation processes. The timing and area models in McPAT are derived from CACTI [20], while the dynamic power model is similar to Wattch [18], additions being short circuit and leakage models.

Sniper/McPAT has some limitations as the authors describe in [5]. So, Sniper is a user-level simulator, hence, it does not model system-level code. Also, Sniper does not model the internals of a superscalar out-of-order processor core. A high-level overview of the combined architecture is shown in Fig. 1.

Sniper's simulation speed is very high: around 2 MIPS when simulating 16-core architectures on an 8-core host machine as shown in [5].

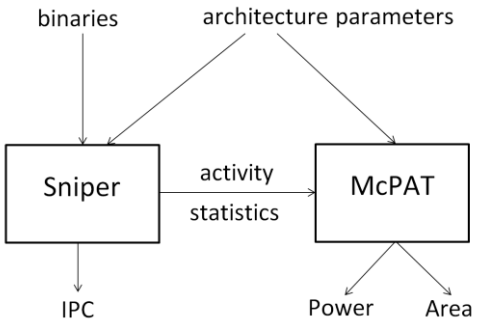

Fig. 1. Overview of Sniper/McPAT

The simulation environment is composed of Sniper 5.3. The baseline processor configuration is created after an Intel Nehalem [23] machine using the gainestown.cfg file in Sniper. Some basic information about this processor is shown in Table I.

The benchmarks used in this paper are the ones in the SPLASH-2 [21] benchmark suite, compiled with GCC 4.8.2 in 4 optimization modes using the -o1, -o2, -o3 and -os optimization flags [22]. The default input size for the DSE process was "test"; only some of the results were run on "small", a longer and more accurate input size than "test", because of the time constraints.

We installed and ran this environment on Ubuntu 14.04, disposing an Intel Quad Core I7 4770K CPU, clocked at 4.4 GHz with 16 GB of DRAM and SSD hard drive.

IV. Design Space Exploration and Optimization

Our simulations took into consideration 3 objectives: performance – measured in average clocks per instruction (CPI), hardware complexity (integration area) and energy consumption. All these objectives have to be minimized. CPI is computed by Sniper, while hardware complexity and energy consumption are computed by McPAT.

The design space exploration process can be done in different ways:

1) Isolated optimization:

a) Fixed SW: Fixed software parameters, vary the hardware parameters and analyze the impact of the hardware parameters on the objectives.

b) Fixed HW: Fixed hardware parameters, vary the software parameters and analyze the impact of the software parameters on the objectives.

2) Co-optimization of the hardware and software parameters

TABLE I. GAINESTOWN INFORMATION

Produced	From 2008 to present
Max. CPU clock rate	1866 MHz to 3333 MHz
Min. feature size	45 nm
Instruction set	x86
Microarchitecture	Nehalem
CPUID code	106Ax
Product code	80602
Cores	4
L2 cache	4×256 kB
L3 cache	8 MB
Package(s)	LGA 1366
Brand name(s)	Xeon 55xx

A. Hardware Optimization

In order to find the best configurations we varied a number of hardware parameters. These parameters and the minimum and maximum values are shown in Table II.

Number of cores, cache size, block size and associativity are the well-known parameters and are per individual core. L3 Cache Shared Cores represent the number of cores that a L3 cache shares. The DRAM interleaving controllers must be a multiple of the Shared Cores. All the above parameters are powers of 2. This computes to a design space of 373.248, which is not that big but the time to evaluate each of these configurations on the test input size is about 2 minutes, which computes to about 500 days of computation on a single core. To have more accurate results, simulation on small or large input sizes are needed, which take at least 10x the time.

In order to incorporate the domain-knowledge of the architecture designer, FADSE uses a set of rules to take out the infeasible individuals, like one that has the L1 cache size greater than L3 cache size. We know that such a CPU is not acceptable. This way the search space will be composed only of the feasible individuals. The rules are the following:

- L3 cache size is greater or equal to L2 cache size
- L2 cache size is greater or equal to L1 cache size
- Number of cores is greater or equal to L3 cache shared cores
- DRAM interleaving controllers greater or equal to L3 cache shared cores

B. Software Optimization

The software parameters we varied are presented in Table III.

The GCC optimization flags correspond to the default flags [22]: o1 (Moderate optimization; optimizes reasonably well but

TABLE II. HARDWARE PARAMETERS

Parameter	**Values**	
	Min	***Max***
Number of cores	1	16
L1 DCache Assoc	1	8
L1 Dcache Block Size	64	128
L1 Dcache Size	64	256
L2 Cache Assoc	1	8
L2 Cache Block Size	64	128
L2 Cache Size	64	2048
L3 Cache Assoc	1	8
L3 Cache Size	64	16384
L3 Cache Shared Cores	1	16
DRAM interleaving controllers	2	32

TABLE III. SOFTWARE PARAMETERS

Parameter	Values
GCC optimization	-o1, -o2, -o3, -o4
Number of threads	1,2,4,8,16
Scheduler	pinned, roaming

does not degrade compilation time significantly), o2 (Full optimization; generates highly optimized code and has the slowest compilation time), o3 (Full optimization as in o2; also uses more aggressive automatic inlining of subprograms within a unit and attempts to loops vectorization) and os (Optimize space usage -code and data - of resulting program).

The number of threads is varied from 1 to the maximum number of cores, while the scheduler can take two different values: pinned (this is the default - threads are assigned to a single core, round-robin; no migration) or roaming (initial affinity: all cores; threads freely migrate to idle cores).

The variation of only the software parameters will bring us 40 different configurations, while searching through the whole design space, hardware and software parameters, will account for a number of 373.248 * 40 ~ 15 million possible configurations. Exhaustive search through such a search space will take almost half a year if every configuration will take only 1 second to evaluate on a single core machine, which is not our case.

V. OPTIMIZATION RESULTS

Setting the hardware parameters to fixed values would fix also the integration area objective and the optimization would run only in a 2-D space. This is why we didn't try such an approach in the current paper. The variation of the software parameters will be presented in the following paragraphs and at the end the hardware and software co-optimization.

A. *Influence of the optimization flags on IPC and Power*

Fig. 2 shows that the best performance can be obtained using GCC optimization flags of –o3 and –os. The highest improvements can be seen for lu.ncont, while the highest decreases of performance can be found for radix –o3 and water.nsq with –o4. At average, the best energy efficiency is obtained for -o3 and and –os optimizations. Further details can be observed in Fig. 3.

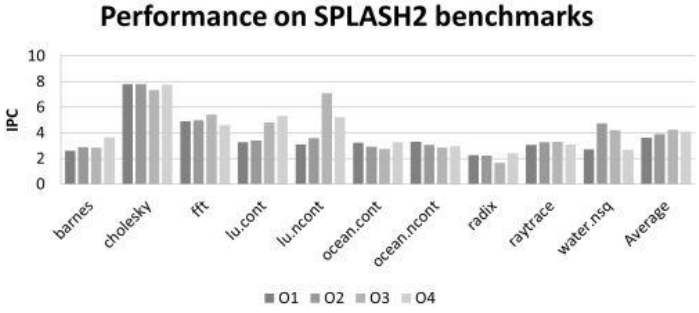

Fig. 2 Software optimization's impact on performance

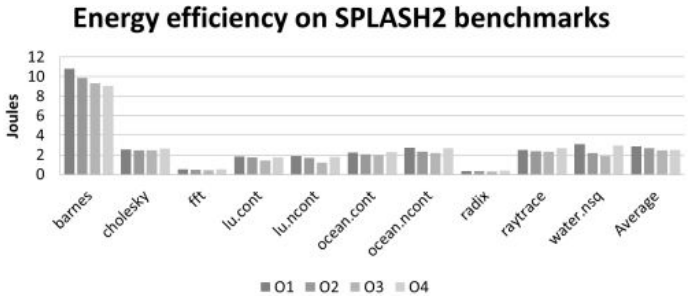

Fig. 3. Software optimization's impact on the Energy

B. *Influence of the number of threads on IPC and Power*

Another very important software parameter is the number of threads that an application can use. A variation of the number of threads from 0 to the number of cores on performance and energy efficiency are giving the expected results: the more threads there are, the better it is for the performance, while the energy remains almost the same. The power dissipated increases with the number of threads, but the time needed to run the benchmarks decreases with the same amount. The results are shown in Fig. 4 and Fig. 5 for the SPLASH-2 benchmarks.

Because there is almost no difference between the 8 cores with 4 threads and 4 cores with 4 threads in respect to the energy efficiency and performance, just the complexity of the processor is greater, we have decided to set always the number of threads equal to the number of cores in the hardware-software co-optimization run presented at the end of this paragraph. This way we get always the highest performance from every configuration, almost the same energy efficiency, but at a reduced complexity (area).

C. *Influence of the Scheduler on IPC and Power*

Thread scheduling support was added in Sniper 4.0. It is fully implemented in the simulator and no application modifications or re-linking required. The scheduling component supports affinity mask for each thread (allowed cores) and also pre-emptive round robin scheduling. The used types of schedulers are: pinned and roaming. The pinned

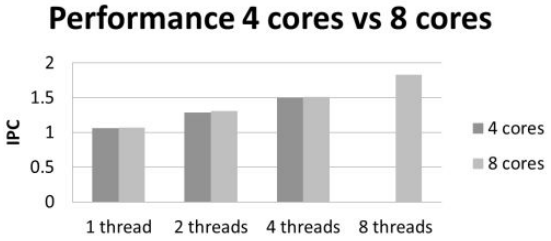

Fig. 4. Influence of number of threads on Performance

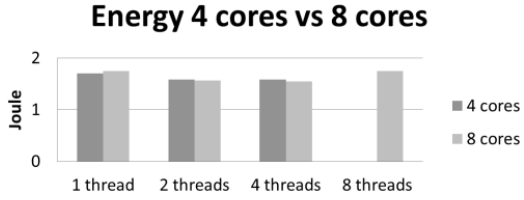

Fig. 5. Influence of number of threads on Energy

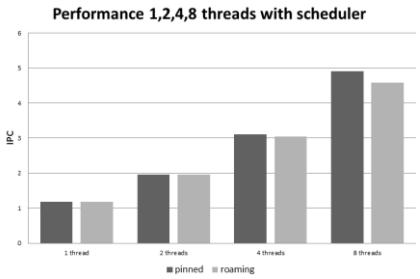

Fig. 6. Scheduler impact on performance

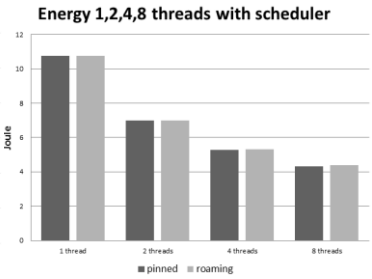

Fig. 7. Scheduler impact on energy

scheduler is the default one. Here threads are assigned to a single core, round-robin, without any migration. The roaming scheduler allows free migration of threads to idle cores and the initial affinity consists of all cores. Also custom thread scheduling is supported, but it is not used in this paper. As it can be seen, differences between the two schedulers start to appear at 8 threads and above. In our case the scheduler is not going to influence significantly the outcome of our simulations. It can be observed, that on average at 8 threads, the IPC decreases slightly with the roaming scheduler.

D. Hardware-Software Co-Optimization

With all the information gathered we started an actual DSE optimization process to find out the best individuals regarding the 3 objectives: CPI, Area and Energy. We ran 2 sets of simulations: HW – just hardware parameters modified and HWSW – where we varied both hardware and software parameters, to see the improvements that the hardware-software co-optimization has. Because we have seen that the number of threads does not have a very big impact on the performance and energy, just the area is bigger, we decided to do the runs with the number of threads set equal to the number of cores. Both runs started from quasi the same population: hardware parameters were the same, but because the HWSW run could also change the software parameters, these were not set to the default values: scheduler = pinned and optimization flag set to –o3. The NSGA-II was the heuristic algorithm of choice, which ran for 75 generations a population of 30 individuals.

The results do show what we had expected. The HWSW co-optimization run yielded better results than the result where only hardware parameters were modified.

The coverage metric is a quality metric and can be used to compare the two runs by the fraction of individuals from one algorithm that are nondominated by individuals from the other

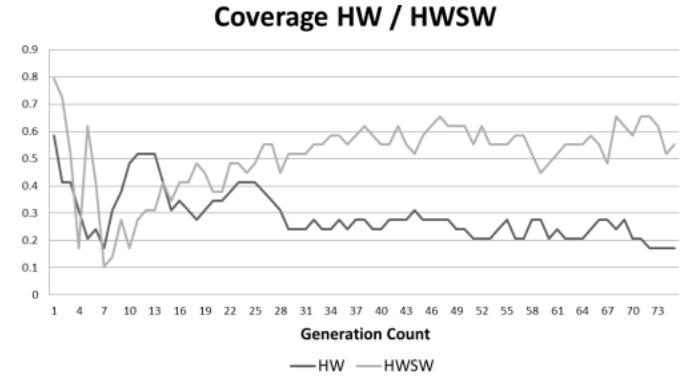

Fig. 8. Coverage metric comparison

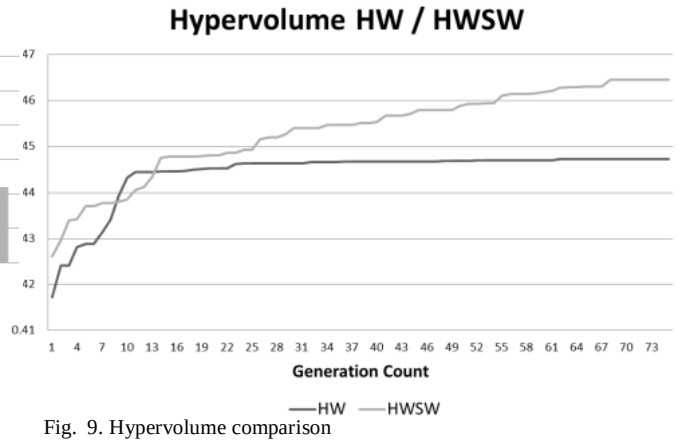

Fig. 9. Hypervolume comparison

run. Fig. 8 presents the coverage over the 75 generations. It can be said that the HWSW run is better than the HW run, because more individuals from the HWSW run are nondominated by individuals from the HW run. A quite interesting point is the fact that there are still around 20% of the individuals from the HW run that are not dominated by individuals from the HWSW run. One would expect the co-optimization to find all the best individuals, but because of the huge design space, 15 million configurations and the small population size, 30 individuals (individuals) it will take a while, more than our 75 generations to get to that point.

The Hypervolume computes the volume enclosed by the current Pareto front and the 3-D hypervolume reference point in our minimization problem. For a maximization problem, the hypervolume is computed as the volume enclosed by the current Pareto front approximation and the axes. Considering our minimization problem, the hypervolume reference point's coordinates are set to the maximum values of the objectives. Looking at the this indicator we can say that the HW run has a better convergence speed; this means that after around 25 generations this run has found the best solutions and their quality is not going to significantly increase in the future. Looking at Fig. 9., the HWSW run, we can say that this run is continuing to give us better and better results and the trend is seen throughout all the generations. The hypervolume is a quality/convergence metric and it can be said that the HWSW run is still converging and the run should have not been stopped at the 75th generation. On the other hand, after the 30th generation of the HW run, the quality of the solutions increases negligible and we could have stopped that run at the 30th generation.

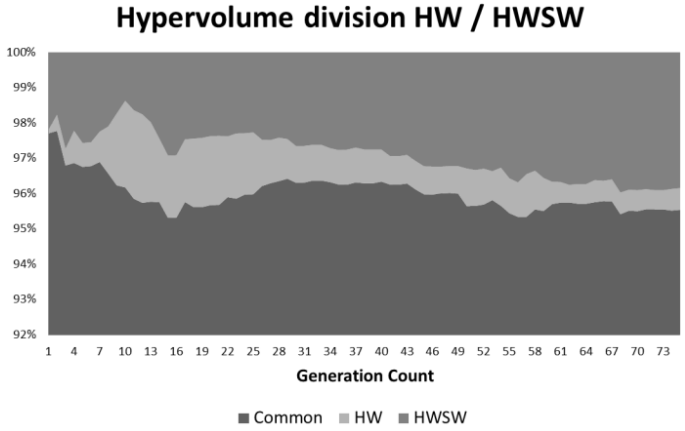

Fig. 10. Two set hypervolume difference comparison

TABLE IV. SOME OF THE BEST INVIDUALS BELONGING TO THE PARETO FRONTS

Individual with		Objectives			Some Parameters				
		*Energy (Watt*cycle)*	*CPI (clocks/instr)*	*Area (mm^2)*	*Cores*	*Dram controllers*	*L3 cache shared cores*	*Software optimization*	*Scheduler type*
HW	Best CPI	5.02E-08	**0.477960701**	1405.18	16	32	8	o3	pinned
	Best Area	4.62E-09	0.940438872	**44.294**	1	32	1	o3	pinned
	Best Energy	**4.60E-09**	0.951374207	44.3023	1	32	1	o3	pinned
HW SW	Best CPI	4.06E-08	**0.45045045**	932.831	16	32	8	os	pinned
	Best Area	3.77E-09	1.182654402	**27.5819**	1	32	1	o1	pinned
	Best Energy	**3.73E-09**	1.2	27.5819	1	32	1	o3	pinned

From the two previous metrics, we could conclude that the HWSW run is much better than the HW run: it continues to provide us better results and also more than 50% of its individuals are nondominated by individuals from the second run. But in fact, the difference between the two runs is not that big. Looking at the Two Set Hypervolume Difference in Fig. 8 we see that more than 95% of the hypervolume is common. So in fact, both runs differ at most in 5%: the HW run dominates by itself around 1%, while HWSW dominates just by itself 4%. This gap is possible to get bigger until the co-optimization run reaches its best found solutions.

Table IV. shows the differences between the best found objectives in each of the runs. This clearly shows that the hardware-software co-optimization generates better results. The CPIs are almost equal but the area and power consumption of the solution found by the HWSW run are much better than the ones found by changing only the hardware parameters. Also the individuals with the best area and best energy are significantly better than the ones found by the hardware optimization. There is no difference in area between the best area and best energy for HWSW, but it can be seen what influence the optimization flags have on a specific configuration. Increasing the benchmark sizes to "small" or "large" could give us greater differences in respect to the Energy and CPI. Table IV. also shows that the best CPI have the configurations with the most cores, 16, but also that optimization flag –os could improve performance on the SPLASH-2 benchmark suite.

E. Temperature computations

For the best 3 found individuals by the HWSW, Best CPI, Best Area and Best Energy, also the temperature has been computed as presented in [24]. The temperatures have been reported by HotSpot after the simulation of the benchmarks on the "small" input size. The maximum temperature of the Best CPI is 65.21 degrees Centigrade, for the Best Energy the maximum temperature was 51.21 degrees, while the best Area had a maximum temperature of 50.52 degrees. Average temperatures are not of importance because the maximum temperature can damage the chip, although the medium temperature could be well under the maximum threshold.

VI. CONCLUSIONS AND FURTHER WORK

Hardware-software co-optimization should be the design paradigm of choice also for the high performance computing systems, not only for the embedded systems. It is increasing in popularity because at an early design cycle there are simulators for that architecture and also the applications that will run on that simulator, and so an optimization of both hardware and software parameters could be conducted.

We have seen that an optimization of only hardware parameters does not yield the best results and some of the software parameters have a great impact on the outputs. The number of threads is a very important software parameter that should be taken into consideration and in this paper we have seen that a number of threads equal to the number of cores gives the best results.

The GCC optimization flags also influence the performance and energy of a configuration, in some of the benchmarks having a greater impact like "lu.ncont", while in others not so much like "cholesky". O3 and os optimization flags give us the best performance, but also the highest power consumption.

The schedulers influence on the IPC increases with the number of threads. It can be the case that at hundreds of threads one implementation of the scheduler is going to be much better than the other one. For our simulations, both schedulers were having almost the same influence and as a further work we could test with more cores and more threads.

All metrics have shown that the hardware-software co-optimization is better that only changing hardware parameters during a design space exploration process. The Sniper multicore simulator is a great x86 simulator, has been validated against the Intel Nehalem architecture and provides a lot of possibilities for future optimizations.

Further developments that will be looked at in future articles are:

- Taking into account that our research group already integrated HotSpot simulator in Sniper [24] we intend to Pareto optimize Sniper through FADSE in a native 4D space by adding the temperature objective.
- Developing more effective domain micro-ontologies related to Sniper, but also other specific systems, like

the embedded, using fuzzy rules, semantic nets, and conceptual graphs.

- Integrate Response Surface Models in FADSE for the acceleration of the DSE process.
- Connect other tools to FADSE, which are not necessarily computer systems' simulators.

ACKNOWLEDGMENT

The authors would like to thank Andrei Daian, student at „Lucian Blaga" University of Sibiu, for providing a Sniper connector between FADSE and Sniper simulator. This connector does the translation from the individuals from the heuristic algorithm that is run, to an actual command line for a configuration that will be run in the Sniper multi-core simulator. The computation of the results is also made through the connector and objectives are sent back to the heuristic algorithm.

We would also like to thank Claudiu Buduleci, student at "Lucian Blaga" University of Sibiu, for connecting HotSpot with Sniper for the temperature computations.

REFERENCES

[1] Chis R., Vintan M., Vintan L., "Multi-objective DSE Algorithms' Evaluations on Processor Optimization, Proceedings of 10th International Conference on Intelligent Computer Communication and Processing", pp. 27-34, Cluj-Napoca, September 2013

[2] Jahr R., Calborean H., Vintan L., Ungerer T., "Finding Near-Perfect Parameters for Hardware and Code Optimizations with Automatic Multi-Objective Design Space Explorations", Concurrency and Computation: Practice and Experience, doi: 10.1002/cpe.2975, 2012

[3] Calborean H., Vintan L., "An Automatic Design Space Exploration Framework for Multicore Architecture Optimizations", Proceedings of The 9-th IEEE RoEduNet International Conference, pp. 202-207, Sibiu, June 2010

[4] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations," International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Nov. 2011

[5] W. Heirman; S. Sarkar; T. E. Carlson; I. Hur; L. Eeckhout, "Power-Aware Multi-Core Simulation for Early Design Stage Hardware/Software Co-Optimization", International Conference on Parallel Architectures and Compilation Techniques (PACT), November 2012

[6] V. Desmet, S. Girbal, A. Ramirez, A. Vega, and O. Temam, "ArchExplorer for Automatic Design Space Exploration," IEEE Micro, vol. 30, no. 5, pp. 5–15, 2010.

[7] V. Zaccaria, G. Palermo, F. Castro, C. Silvano, and G. Mariani, "Multicube Explorer: An Open Source Framework for Design Space Exploration of Chip Multi-Processors," in Architecture of Computing Systems (ARCS), 2010 23rd International Conference on, 2010, pp. 1–7.

[8] S. Kang and R. Kumar, "Magellan: A Search and Machine Learning-based Framework for Fast Multi-core Design Space Exploration and Optimization," in Design, Automation and Test in Europe, 2008. DATE '08, 2008, pp. 1432–1437.

[9] Z. J. Jia, A. D. Pimentel, M. Thompson, T. Bautista, and A. Nunez, "NASA: A generic infrastructure for system-level MP-SoC design space exploration," in Embedded Systems for Real-Time Multimedia (ESTIMedia), 2010 8th IEEE Workshop on, 2010, pp. 41–50.

[10] J. Keinert et al., "SystemCoDesigner - an automatic ESL synthesis approach by design space exploration and behavioral synthesis for streaming applications", ACM Transactions on Design Automation of Electronic Systems (TODAES), January 2009.

[11] M. Rashid, F. Ferrandi, K. Bertels, "hArtes design flow for heterogeneous platforms", Proceeding of: Quality of Electronic Design, (ISQED) March 2009.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," Evol. Comput. IEEE Trans. On, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[13] C. C. Coello, G. B. Lamont, and D. A. van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, 2007.

[14] A. J. Nebro, J. J. Durillo, J. García-Nieto, C. A. C. Coello, F. Luna, and E. Alba, "SMPSO: A New PSO-based Metaheuristic for Multi-objective Optimization," in 2009 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM 2009), 2009, pp. 66–73.

[15] M. Sierra and C. Coello Coello, "Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and Dominance," in Evolutionary Multi-Criterion Optimization, vol. 3410, C. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Eds. Springer Berlin Heidelberg, 2005, pp. 505–519.

[16] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," Evol. Comput. IEEE Trans. On, vol. 3, no. 4, pp. 257–271, Nov. 1999.

[17] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications", vol. 63. Shaker Ithaca, 1999.

[18] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations", in ISCA, pages 83–94, 2000.

[19] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures", in MICRO, Dec. 2009

[20] N. Jouppi. "A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies", in ISCA, pages 51–62, pages 469–480, 2009.

[21] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations", in ISCA, pages 24–36, 1995.

[22] GCC optimization level: gcc.gnu.org/onlinedocs/gcc-4.6.1/gnat_ugn_unw/Optimization-Levels.html

[23] Nehalem architecture: http://en.wikipedia.org/wiki/Xeon

[24] A. Florea, C. R. Buduleci, R. Chis, Á. Gellert, L. Vintan, "Enhancing the Sniper Simulator with Thermal Measurement", Proceedings of The 18-th International Conference on System Theory, Control and Computing, Sinaia (Romania), October 17 - 19, 2014 (submitted).