A Design Space Exploration Methodology for Parameter Optimization in Multicore Processors

Prasanna Kansakar* and Arslan Munir†
Parallel and Distributed Computing and Embedded Systems Laboratory
Department of Computer Science and Engineering
University of Nevada, Reno
*Email: *pkansakar@nevada.unr.edu, †arslan@unr.edu*

***Abstract*— Due to the increasing proliferation of computing systems in diverse application domains, the need for application-specific design of multicore/manycore processing platforms is paramount. In order to tailor processors for application-specific requirements, a multitude of processor design parameters need to be tuned accordingly. Tuning of processor design parameters involves rigorous and extensive design space exploration over large search spaces. In this paper, we propose an efficient design space exploration methodology for multicore parameter optimization. Our proposed methodology includes an intelligent initial parameter setting algorithm, the results of which are leveraged by two search algorithms–exhaustive search and greedy search. We evaluate the methodology in a cycle-accurate simulator (ESESC) using standard set of PARSEC and SPLASH2 benchmarks for applications with low-power and high-performance requirements. The results reveal that the quality of solutions (design configurations) obtained from our methodology are within 1.35%–3.69% of the solutions obtained from fully exhaustive search while only exploring 2.74%–3% of the design space. Our methodology achieves on average a 35.32× speedup in design space exploration time as compared to fully exhaustive search in finding the best processor design configuration.**

***Keywords*-multicore/manycore processors; processor design parameters; design space exploration; parameter optimization**

I. Introduction and Motivation

Computing systems have permeated into several diverse application domains each having unique computing requirements that are not efficiently met by a generic computing system. Consequently, there is a need to design multicore and/or manycore processors tailored to application-specific requirements. Multicore/manycore processor design involves exploring a large design space to find a configuration that satisfies multiple (possibly conflicting) design metrics [1]. Simulating all the possible configurations in the design space is often not feasible, especially for large design spaces, because of resource and time constraints. Design time is a critical factor as it influences the *time-to-market*. A longer time-to-market can result in significant revenue losses for vendors [2].

To meet short time-to-market and domain-specific application requirements, efficient design space exploration methodologies are imperative for the application-specific design of multicore/manycore processors. Furthermore, when considering optimization of multiple, possibly conflicting, design metrics, it is often impossible to find a global optimal solution. Pareto analysis [3] has to be employed in the case of multi-parameter optimization to select a trade-off solution, which favours one of the conflicting metrics over the others. There exists only limited work [4]–[6] on parameter optimization methodologies, especially for multicore/manycore processors. In particular, the challenge is to devise parameter optimization methodologies that quickly explore the design space to provide an optimal or near-optimal solution.

In this paper, we propose a methodology for multicore/manycore processor parameter optimization in which we partially explore the processor design space and use the Pareto Efficiency concept to find a parameter configuration that gives the best trade-off solution for the specified application requirements. We utilize a combination of exhaustive, greedy and one-shot (initial tunable parameter settings) searches to efficiently perform design space exploration. Our methodology also accommodates multiple conflicting design metrics in the optimization process. We extensively tested our methodology on a cycle-accurate simulator using a large set of multi-threaded benchmarks. Our main contributions in this paper are:

- We propose an efficient methodology (in terms of design time, resources, and solution quality) for multicore parameter tuning that combines three different exploration methods: exhaustive, greedy, and one-shot.
- We present a one-shot optimization algorithm that can determine initial settings for each tunable parameter to within 51.26% of the best setting for that parameter.
- We describe an intelligent set partitioning algorithm, which groups parameters based on their significance towards the targeted application-specific design requirements. The algorithm partitions the parameters in three significance groups one for each of the exploration methods: exhaustive, greedy and one-shot.
- We propose exhaustive and greedy search algorithms, which determine best parameter settings to within 1.35%–3.69% of the best settings obtained from fully exhaustive search of the design space.

II. Related Work

There has been work done in literature relevant to design parameter tuning [4]–[6] and several innovative optimization algorithms based on exhaustive search, greedy search, genetic algorithms, evolutionary algorithms etc., have been proposed.

Givargis et al. [7] developed a system PLATUNE that carried out exhaustive searches in two stages: first over clusters of strongly interconnected parameters to obtain pareto-optimal configurations local to each cluster and second over all the clusters to obtain a global pareto-optimal solution. The approach could explore design spaces as large as 10^{14} configurations, but it took an order of 1-3 days to complete. Palesi et al. [8] proposed an improvement over [7] by introducing a new *threshold value* that distinguished between clusters based on the size of their partial search-space since exhaustive search in a large partial search space is infeasible. Exhaustive search method was used for clusters with partial search-spaces smaller than the threshold value and a genetic exploration algorithm was used for larger spaces. Through this improvement, they were able to achieve 80% reduction in simulation time while still remaining within 1% of the results obtained from exhaustive search. Genetic algorithms were also used in the system MULTICUBE, by Silvano et al. [9]. The MULTICUBE system defined an automatic design space exploration algorithm that could quickly determine an approximate pareto-front for given design requirements.

Munir et al. [10] proposed another alternative to overcome the overhead of exhaustive search in their work on dynamic optimization of wireless sensor networks. Their approach was divided into two phases. In the first phase, a one-shot algorithm intelligently selected initial parameter settings and further ordered the parameters based on their significance towards the application requirements. In the second phase, the design space was explored using a greedy search algorithm. Their approach yielded a design configuration that was within 8% of the optimal configuration while only exploring 1% of the design space.

In this paper, we improve on the work carried out by Munir et al. [10]. We leverage a similar approach to design space exploration but add two new phases: a set partition phase and an exhaustive search phase. The addition of the exhaustive search phase aims at increasing the degree of closeness to the optimal solution by exploring a larger portion of the design space, as argued by Silvano et al. [9]. The limit on the number of configurations considered in the exhaustive search is determined by the set-partitioning phase that uses a threshold value [8].

III. Parameter Optimization Methodology

Our design space exploration methodology consists of four distinct phases: initial parameter setting selection and parameter significance, set partition, exhaustive search, and greedy search. This section formulates the design space, benchmarks, and the objective function. The section then discusses the phases of our proposed methodology.

A. Design Space

Consider n number of tunable parameters which describe the design of a multicore/manycore processor. Let P be the list of these tunable parameters defined as $P = \{P_1, P_2, P_3, \ldots, P_n\}$. Each tunable parameter P_i (where $i \in \{1, 2, \ldots, n\}$) in the list P denotes the set of possible settings for i^{th} parameter. Let $L = \{L_1, L_2, L_3, \ldots, L_n\}$ be the set containing the size of the set of possible settings for each parameter in list P, such that, $L_i = |P_i| \ \ \forall\ i = 1, 2, \ldots, n$ where $|P_i|$ is the cardinality of the set P_i. Each parameter setting set P_i in the list P is defined as $P_i = \{P_{i1}, P_{i2}, P_{i3}, \ldots, P_{iL_i}\} \ \ \forall\ i \in \{1, 2, \ldots, n\}$. Note that the values in the set P_i are arranged in ascending order. The state space for the design space exploration is the collection of all the possible configurations that can be obtained using the n parameters. This state space S is given as $S = P_1 \times P_2 \times P_3 \times \ldots \times P_n$ where $\times$ represents Cartesian product.

We denote s as a design configuration in the state space S. A subscript to s refers to a specific design configuration. For example, a state s_f that consists of the first setting of each tunable parameter can be written as $s_f = (P_{11}, P_{21}, P_{31}, \ldots, P_{n1})$. For an incomplete/partial design configuration of a tunable parameters we use the term δs_a.

B. Benchmarks

Each of the configurations, selected from the state space S by our algorithm, is tested on m number of test benchmarks. The design metrics for each simulated configuration are collected separately for each benchmark.

C. Objective Function

In our methodology, design configurations are compared with each other based on their objective function values. The objective function of a design configuration is the weighted sum of the design metrics obtained after simulating that design configuration. Let o be the number of design metrics and V defined as $V_s^k = \{V_{s(1)}^k, V_{s(2)}^k, V_{s(3)}^k, \ldots, V_{s(o)}^k\} \quad \forall\ k = 1, 2, \ldots, m$, be the set of values of the design metrics obtained from the simulation of the configuration $s \in S$. Let w defined as $w = \{w_1, w_2, w_3, \ldots, w_o\}$ be the set of weights for the design metrics based on the application requirements, such that $0 \leq w_l \leq 1$ and $\sum w_l = 1 \ \ \forall\ l = 1, 2, \ldots, o$. These weights are set by the system designer.

The objective function $\mathcal{F}$ of a design configuration s for a test benchmark k is then defined as $\mathcal{F}_s^k = \sum w_l V_{s(l)}^k \ \ \forall\ l = 1, 2, \ldots, o$.

The optimization problem, considered in this paper, is to minimize the value of the objective function $\mathcal{F}$. The design metrics are chosen such that the minimization of their values is the favourable design choice. For example, when considering the performance metric, the design goal is to maximize performance, however, to model this performance requirement into the objective function, we use execution time as a measure of performance. Minimizing execution time would fit with minimizing the objective function while still modelling the design goal of maximizing performance.

The optimization problem for each test benchmark k is defined as: $min.\ F_s^k$, s.t. $s \in S$.

D. Methodology Phases

Phase I: Initial Parameter Setting Selection and Parameter Significance: This phase involves the steps in selecting initial parameter setting of each tunable parameter. In this phase, the parameters are processed one at a time; objective functions are calculated for the parameter under consideration by varying between its first setting (corresponding to configuration s_f) and last setting (corresponding to configuration s_l) while keeping all the other parameters arbitrarily assigned to their first setting. The significance of each tunable parameter with respect to the objective function is taken as the difference in magnitude of its objective function values for s_f and s_l configurations. This difference is stored in a set of parameter significance D. The greater the value of the difference i.e., the magnitude D_i^k (for the i^{th} parameter of the k^{th} benchmark , the greater is the significance of the corresponding parameter with respect to the objective function. Judging by whether the difference D_i^k is positive or negative, the best setting for the current parameter is chosen as either the first setting or the last setting. The best settings for the parameters are stored in the set of best settings B_i^k.

Phase II: Set Partitioning: This phase separates the list of tunable parameters into different search sets based on their significance values with respect to the objective function. The set of parameter significance values, D^k, is sorted in descending order based on the absolute magnitude of the significance values. The index information of the sorted values is preserved in I^k (for the k^{th} benchmark). For example, if the fifth entry D_5^k has the greatest value, D_5 will become the first entry after sorting, and first entry of the set I^k will be 5, that is, $I_1^k = 5$. The index information in I^k is then used to sort the list of tunable parameters P and the tunable parameters set sizes L. This results in an arrangement of parameters such that the parameters with higher significance are placed towards the start of the list and the parameters with lower significance are placed towards the end of the list. The parameters are then sorted into three sets. The parameters with the highest significance in the sorted parameter list are separated into the exhaustive search set. The number of parameters considered depends on the exhaustive search threshold value T such that the size of the partial search space of the exhaustive search set is less than this value. The set of parameters which remain is then divided into two halves: the upper half of the set is separated as greedy search set and the lower half is separated as one-shot search set. We choose to partition the set into two halves because we have observed empirically that selecting the half sets provides efficient design space exploration without significantly compromising the solution (i.e., best design configuration) quality.

Phase III: Exhaustive Search Optimization: In this phase, partial design configurations are generated from parameters in the exhaustive search set, which are then combined with the parameters not considered in the exhaustive set to form a complete simulatable design configuration. The parameters that are not in the exhaustive search set are set to their best values obtained in the initial parameter setting phase defined by Initial Parameter Setting Selection and Parameter Significance phase. At the end of the exhaustive search phase, the design configuration that yields the smallest value for the objective function is returned.

Algorithm 1 Greedy Search Algorithm

Input: P - List of Tunable Parameters; D - Significance of Parameters towards Objective Function; B - Set of Best Settings for Oneshot and Exhaustive Search; $\mathcal{G}$ - Set of Parameters for Greedy Search
Output: B - Complete set of Best Settings

```
s_G = ∅, δs_G' = ∅, G_P = ∅
for k = 1 to m do
    F^k_{s_b} = ∞
    for i ← 1 to n do
        if P_i ∈ G^k then
            if D^k_i < 0 then
                G_P = sortDescending (P_i)
            end if
            for j ← 1 to n do
                if P_j ≠ G_P then
                    δs^k_{G'_P} = δs^k_{G'_P} ∪ {B^k_j}
                end if
            end for
            for l ← 1 to L_i do
                s^k_G = δs^k_{G_P'} ∪ {G_{P(l)}}
                Explore k^th benchmark using configuration s^k_G
                Calculate F^k_{s_G}
                if F^k_{s_G} < F^k_{s_b} then
                    F^k_{s_b} = F^k_{s_G}
                    B^k_i = G_{P(j)}
                else
                    break
                end if
            end for
        end if
    end for
end for
```

Phase IV: Greedy Search Optimization: **Algorithm 1** describes the final phase of our methodology which involves greedy search. For each parameter in the greedy search set $\mathcal{G}$, the set of possible settings for that parameter is either sorted in descending order or left unchanged (i.e., default ascending order) depending on the sign of the significance value of that parameter. That is, negative sign of the significance value for a parameter results in the sorting of that parameter's settings set in descending order since the last parameter setting of that tunable parameter yields a lower objective function value. This ordering ensures that when the search progresses for each tunable parameter setting, the best initial setting for the parameter is encountered first.In the search

Table I
LIST OF TUNABLE PARAMETERS AND SETTINGS

Parameter Name	Set of Settings
Cores (PARSEC)	2, 4, 8
Cores (SPLASH2)	2, 4
Frequency (MHz)	1700, 2200, 2800, 3200
L1 I-Cache Size (KB)	8, 16, 32, 64, 128
L1 D-Cache Size (KB)	8, 16, 32, 64, 128
L2 Cache Size (KB)	256, 512, 1024
L3 Cache Size (KB)	2048, 4096, 8192

process for each parameter, all the parameters except for the parameter currently being explored are assigned settings using the set of best settings B_i^k (Section III-D).

The greedy search algorithm explores each setting of the parameter under consideration in the greedy search set, starting from the best initial setting of that parameter, to generate a new test design configuration. The algorithm compares the new test design configuration with previous configurations generated by changing the same greedy parameter settings. If there is an improvement in the objective function value obtained with the new configuration, then the next parameter setting for that tunable parameter is explored, otherwise, the search is terminated for that parameter and the next parameter in the greedy search set is explored.

IV. EXPERIMENTAL SETUP

To evaluate our parameter optimization methodology, we have simulated the design configurations dictated by our methodology using the ESESC [11] (Enhanced Super ESCalar) simulator. The test benchmarks used in our evaluation are from the PARSEC and SPLASH2 [12] benchmark suite. We have implemented our methodology phases in Perl [13]. The results from each of the simulations carried out were gathered in MS Excel using the tool Excel-Writer-XLSX [14] for Perl. The ARM-binaries for all the test benchmarks were compiled using arm-linux-gnueabihf toolchain [15]. We tested our parameter optimization methodology on a design space consisting of six parameters with possible settings listed in **Table I**. This yields a state space of 2700 configurations for PARSEC and 1800 configurations for SPLASH2 benchmarks. We used the following benchmarks from the PARSEC and SPLASH2 suites to test our algorithm:

PARSEC Benchmarks: Blackscholes, Canneal, Facesim, Fluidanimate, Freqmine, Streamcluster, Swaptions, x264

SPLASH2 Benchmarks: Cholesky, FFT, LU-cb, LU-ncb, Ocean-cp, Ocean-ncp, Radiosity, Radix, Raytrace

We define two sample application domains which we use to evaluate our methodology: a low-power application domain ($w_{\mathcal{P}} = 0.9$ and $w_E = 0.1$) and a high-performance application domain ($w_{\mathcal{P}} = 0.1$ and $w_E = 0.9$), where $w_{\mathcal{P}}$ and w_E denote weight factors for total power and execution time metrics, respectively.

The linear objective function for our evaluation tests is formulated by the weighted sum of design metrics, and is given by $\mathcal{F} = w_{\mathcal{P}} \cdot \mathcal{P} + w_E \cdot E$ where, $\mathcal{P}$ denotes the sum of the dynamic power ($\mathcal{P}_d$) and the leakage power ($\mathcal{P}_l$), that is, $\mathcal{P} = \mathcal{P}_d + \mathcal{P}_l$, and E denotes the total execution time.

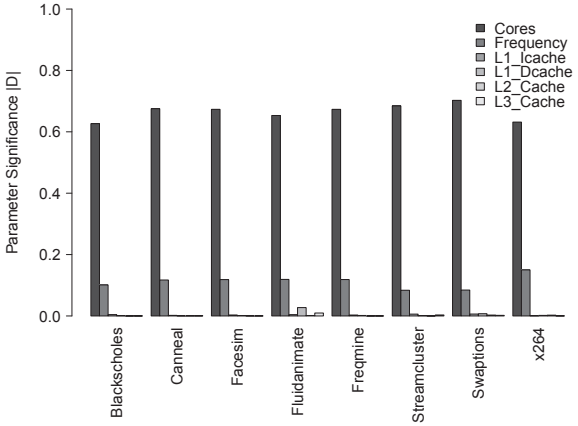

Figure 1. Significance of tunable parameters for PARSEC benchmarks for applications with low-power requirement

V. RESULTS

We verified our methodology by comparing our results with the results obtained from a fully exhaustive search over the design space. We execute our methodology with an arbitrary exhaustive search threshold value of 150 (i.e., T = 150). We clarify that the threshold value of T = 150 means that the maximum number of design configurations that can be explored by exhaustive search phase in our methodology is upper bounded by 150.

A. Parameter Significance

Figure 1 presents the normalized values of parameter significance obtained for various PARSEC benchmarks for applications with low-power requirement. These parameter significance values were calculated based on the simulation results obtained from the initial parameter setting selection and parameter significance phase of our methodology. **Figure 1** reveals that for each test benchmark, there are at most three tunable parameters that have high significance with respect to the objective function. We observe that for low-power requirements, number of cores followed by the core frequency are the parameters that have a significant impact on the objective function. For applications with high-performance requirements (not shown here for brevity), cache sizes (in particular L1-D and L2 cache sizes) also impact the objective function as larger cache sizes can accommodate more of the recently used data and help mitigate cache thrashing, which provides performance improvement. This observation is intuitive because the PARSEC benchmarks are highly data-parallel (and hence performance improvement with an increase in number of cores) and have medium to large working sets (and hence performance improvement with an increase in cache sizes).

B. Pareto Fronts

Figure 2 presents the Pareto fronts for applications with low-power requirement for various PARSEC benchmarks.

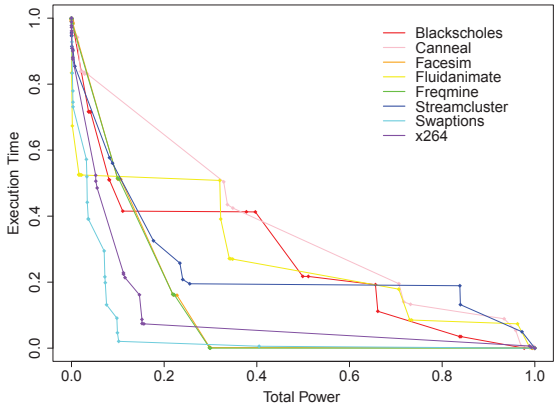

Figure 2. Pareto fronts for PARSEC benchmarks with exhaustive search threshold of $T = 150$ for applications with low-power requirement [total power and execution time values normalized]

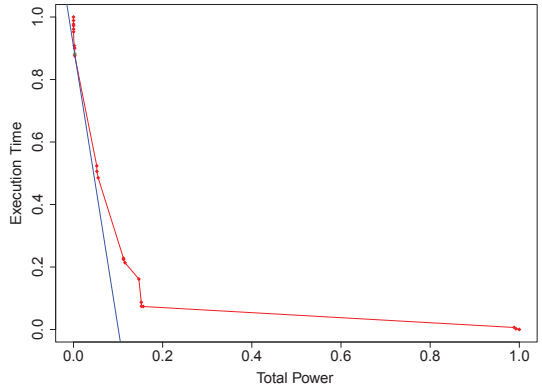

Figure 3. Linear objective function plotted along with the Pareto front for the x264 (PARSEC) benchmark for an application with low-power requirement [total power and execution time values normalized]

The Pareto fronts, which are generated using normalized values of total power and execution time, clearly exhibit the conflicting interdependency between the two design metrics considered , i.e., decreasing execution time results in an increase in total power consumption, and vice versa. Hence, it is not possible to have a single solution to the optimization problem, which gives minimum values for both of these metrics. A favourable trade-off solution between power and performance is the only result that can be obtained from the optimization process. Each of the points on the Pareto front represents a favourable trade-off solution between the conflicting power and performance metrics.

C. Selecting a Favourable Trade-off Solution

Figure 3 illustrates how a favourable trade-off solution is selected from the set of Pareto-optimal trade-off points on the Pareto front. The objective function forms a straight line in the power-performance graph with slope $-w_{\mathcal{P}}/w_E$, where $w_{\mathcal{P}}$ and w_E are the weights associated with total power and execution time design metrics, respectively. These weights indicate the preference/weightage of design metrics with respect to each other. This is seen in the objective function line having a smaller intercept on the horizontal axis (total power) for low-power requirement and having a smaller intercept on the vertical axis (execution time) for high-performance requirement (not shown here for brevity).

Table II
COMPARISON OF SETTINGS FOR BLACKSCHOLES (PARSEC) BENCHMARK FOR OUR PROPOSED METHODOLOGY VERSUS FULLY EXHAUSTIVE SEARCH

	High-Performance		**Low-Power**	
Cores	**8**	8	**2**	2
Frequency	**3200**	3200	**1700**	1700
L1-ICache Size	**64**	64	**8**	16
L1-DCache Size	**32**	128	**16**	16
L2-Cache Size	**512**	256	**512**	1024
L3-Cache Size	**4096**	8192	**8192**	8192
Total Power [W]	**4.309**	4.549	**0.658**	0.660
Execution Time [ms]	**28.153**	28.139	**144.625**	144.809

Figure 3 reveals that for low-power requirement, the favourable trade-off power-performance pair of the form (total power (W), execution time (ms)), obtained from the point of intersection of the objective line with the Pareto front, is (0.904, 68.614). The weight-balanced result for the power-performance pair that is obtained from the fully exhaustive search is (0.909, 65.321). This results in a difference of -0.55% for the total power and 4.79% for the execution-time/performance results. Both configurations yielding these power-performance pairs have 2 processor cores and 1700 MHz operating frequency. The configurations differ slightly on cache sizes.

We obtain the favourable trade-off power-performance pair for the applications with high-performance requirement in a similar manner. Results indicate that our methodology attains results that are consistent with the results obtained from fully exhaustive search results while only exploring 52 configurations (i.e., 1.92% of the design space) for low-power requirement; and 77 configurations (i.e., 2.85% of the design space), for high-performance requirement. These results verify that our proposed methodology explores the design space of multicore/manycore processors in a highly efficient manner.

D. Comparison with Exhaustive Search

To verify the solution quality (design configuration) obtained by our methodology, we compare the results obtained from our methodology with the results obtained from fully exhaustive search for the given application requirements. **Table II** lists details of the design configuration selected by our methodology alongside the one obtained from fully exhaustive search for Blackscholes (PARSEC) test benchmark. The boldface values in the table are parameters settings obtained from fully exhaustive search and the values adjacent to these boldface values are parameter settings obtained from our methodology. **Table II** also presents the power-performance values for configurations obtained from fully exhaustive search as well as from our methodology. We observe that for high-performance requirement, our methodology provides a trade-off solution with execution-time/performance within -0.05% and total power within

5.27% of the solution obtained from fully exhaustive search. We also note that the configuration obtained for high-performance requirement has a higher core count (i.e., 8 cores) and a higher operating frequency (i.e., 3200 MHz) as compared to the configuration obtained for low-power requirement (i.e., core count of 2 and the operating frequency of 1700 MHz). For low-power requirement, our methodology provides a trade-off solution with total power within 0.3% and execution time/performance within 0.12% of the solution obtained from fully exhaustive search. These configuration settings can be explained intuitively. Having a large number of cores working at high frequencies favours high performance whereas having fewer cores operating at low frequencies result in reduced power consumption.

On average, our methodology attains power values within 1.35% for low-power requirement and the performance values within 3.69% for high-performance requirement as compared to fully exhaustive search. Our methodology explores 593 configurations in total for all PARSEC test benchmarks (Section IV) for low-power requirement, and 648 configurations in total for high-performance requirement. The average percentage of the design space explored by our methodology is within 2.74%–3%. Hence, our methodology provides a speedup of 35.32× as compared to fully exhaustive search exploration of the design space. These evaluation results verify that our methodology explores the design space in a highly efficient manner.

VI. Conclusions

In this paper, we proposed a four-phase methodology for efficient design space exploration and tunable parameter optimization for multicore/manycore architectures. The first phase determined good initial settings (within 51.26% of the best settings) for each of the tunable parameters before beginning the search processes. The second phase consisted of a set partitioning algorithm that separated the parameters into different ordered sets based on the significance of the parameters with respect to the objective function and the exhaustive search threshold value supplied by the designer. The third and fourth phases explored the subsets obtained from step two by exhaustive search and greedy search, respectively. To verify our methodology, we compared the results obtained from our methodology with the results from fully exhaustive search. The results revealed that our methodology provided a solution quality within 1.35%–3.69% of the solution quality obtained from fully exhaustive search, while only exploring 2.74%–3% of the design space on average (a speedup of 35.32× as compared to fully exhaustive search). These results verified that our methodology explored the design space efficiently and provided a high quality solution.

In the future, we plan to improve various phases of our methodology. In particular, we intend to investigate the full-factorial design method to improve the initial parameter setting selection phase. We further aim to improve the set partitioning phase by using a better cut-off value for set partition instead of using the exhaustive search threshold value. Furthermore, we plan to compare our methodology with other parameter optimizations techniques that leverage genetic-evolution and machine-learning algorithms.

References

[1] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, *Multiobjective Optimization - Interactive and Evolutionary Approaches*. Verlag Berlin Heidelberg: Springer, 2008.

[2] F. Vahid and T. Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*, 1st ed. New York, NY, USA: John Wiley and Sons, Inc., 2001.

[3] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[4] M. Monchiero, R. Canal, and A. Gonzalez, "Power/performance/thermal design-space exploration for multicore architectures," *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, no. 5, pp. 666–681, 2008.

[5] J. Michanan, R. Dewri, and M. Rutherford, "Understanding the power-performance tradeoff through pareto analysis of live performance data," in *Proc. of the International Green Computing Conference (IGCC),*, Dallas, Texas, USA, Nov 2014, pp. 1–8.

[6] Q. Guo, T. Chen, Y. Chen, Z. H. Zhou, W. Hu, and Z. Xu, "Effective and efficient microprocessor design space exploration using unlabeled design configurations," *ACM Trans. on Intelligent Systems and Technology*, vol. 5, no. 1, pp. 20:1–20:18, Jan 2014.

[7] T. Givargis and F. Vahid, "Platune: A tuning framework for system-on-a-chip platforms," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 11, pp. 1317–1327, Nov 2002.

[8] M. Palesi and T. Givargis, "Multi-objective design space exploration using genetic algorithms," in *Proc. of the 10th International Symposium on Hardware/Software Codesign (CODES)*, Estes Park, CO, USA, May 2002.

[9] C. Silvano, W. Fornaciari, and et al., "MULTICUBE: Multi-objective design space exploration of multi-core architectures," in *Proc. of the International Symposium on VLSI (ISVLSI)*, Lixouri, Kefalonia, July 2010.

[10] A. Munir, A. Gordon-Ross, S. Lysecky, and R. Lysecky, "A lightweight dynamic optimization methodology and application metrics estimation model for wireless sensor networks," *Elsevier Sustainable Computing: Informatics and Systems*, vol. 3, no. 2, pp. 94 – 108, Jun 2013.

[11] E. K. Ardestani and J. Renau, "ESESC: A fast multicore simulator using time-based sampling," in *Proc. of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Washington, DC, USA, Feb 2013.

[12] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Department of Computer Science, Jan 2011.

[13] (2015) Perl reference. [Online]. Available: http://perlmaven.com/

[14] J. McNamara. (2015, Apr) Excel-writer-XLSX. [Online]. Available: http://search.cpan.org/dist/Excel-Writer-XLSX/

[15] Q. Zhang. (2013, Jun) Build PARSEC for ARM. [Online]. Available: http://worklogtk.blogspot.com/2012/06/build-parsec-for-arm.html