An Effective Methodology to Multi-objective Design of Application Domain-specific Embedded architectures

Vincenzo Catania Alessandro G. Di Nuovo Maurizio Palesi Davide Patti
Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
Università di Catania
Viale A. Doria 6, 95125 Catania, Italy
{vcatania,adinuovo,mpalesi,dpatti}@diit.unict.it

Gianmarco De Francisci Morales
IMT - Institutions, Markets, Technologies
Institute for Advanced Studies
Piazza S. Ponziano, 6, 55100, Lucca, Italy
gianmarco.dfmorales@imtlucca.it

***Abstract*—Today's computer systems have become unbelievably complex. Nowadays register-level design is an overwhelming task, especially in the embedded system area where the time-to-market is very short. Platform Based Design shifts the challenge on how to tune parametric platforms to achieve the best performance at the smallest cost. This task, called multi-objective Design Space Exploration, requires accurate strategies because the design space is too vast to be exhaustively evaluated. Even using efficient exploration strategies proposed in the literature, simulation times can become a bottleneck in the design flow.**

In this work we propose a novel approach to application-domain Design Space Exploration using a Multi-Objective Genetic Algorithm and employing HPC to reduce exploration times. The genetic algorithm is preceded by a correlation analysis of the different objectives. The search space is thus reduced by combining highly correlated objectives from different domains. We describe the steps needed to parallelize the exploration on the Grid, and present the results of extensive testing of the proposed approach. We obtained over one order of magnitude reduction in exploration times without hampering the quality of the solutions. Shorter simulation times allow more ideas to be explored in less time. This leads to shorter product time-to-market and a more thorough design space exploration. Furthermore the combination of correlated objectives favors the design of modern multi-purpose devices.

I. Introduction

Looking to the past, embedded systems have been characterized by a common denominator: they were so specialized that the entire logic could be implemented as a single program. For instance the functionality of a mobile phone was "just" making a phone call. Today, it is universally acknowledged that the purpose of an embedded system has been largely extended. In the era of the convergence, devices are becoming more and more general-purpose. The word "application" is being replaced by the word "domain" as embedded systems are designed to support not just one single application but a set of applications often belonging to a given application domain. For instance, the newest mobile phones are complex systems able to connect to the internet, to take pictures, to encode and decode video streams, to implement sophisticated data security mechanisms, and so on. For these applications, there exist two extreme implementations. One is software implementation running on a general purpose processor and the other is hardware implementation in the form of ASIC. In the first case, it is flexible enough to support various applications but may not yield sufficient performance to cope with the complexity of application. In the second case, we can optimize it better in respect of both power and performance but only for a specific application. For this reasons, nowadays the design of an embedded system is no more driven by the optimization of the metrics of interests for a certain application, but more often the goal is optimizing a complex embedded system to support a set of applications belonging to a certain domain [1], [2], [3]. Unfortunately, this problem is not trivial as it could appear, especially when more than one objective metric has to be optimized. Therefore a system configuration which is optimal for a given application could be bad for another application in the given domain, so it could be hard to find a good trade-off to meet all requirements.

The problem of defining the best system configuration (or the Pareto set of system configurations) for a given application is known as *design space exploration* (DSE). The majority of DSE techniques proposed in literature [4], [5], [6] tackle the problem under the simplificative hypothesis that the system will run only a single application.

To perform application-domain DSE many simulations are to be done in order to evaluate just one candidate configuration. In this paper, we propose a methodology to application-domain DSE, that runs in an HPC environment and uses a Multi-Objective Evolutionary Algorithm (MOEA) to additionally reduce the computational effort. In order to assess the effectiveness of our methodology we consider the design of a complex embedded system based on a parameterized VLIW microprocessor along with a highly parameterized memory hierarchy tailored for supporting a class (or domain) of applications, taking also into account the effects of advanced ILP-oriented compilation strategies. More precisely, we define a methodology to search for a Pareto set of system configurations, composed by processor, memory hierarchy and compiler parameters, able to optimize

978-0-7695-3782-5/09 $25.00 © 2009 IEEE
DOI 10.1109/DSD.2009.205

both performance and power dissipation when any of the applications in a given domain is executed.

II. The proposed Methodology for Multi-objective Application Domain-specific DSE

This section will present the proposed methodology to application domain DSE. Starting from the results obtained using well-known DSE techniques in literature [4], [5], [6], the main problem in designing an application domain approach is how to merge the different Pareto sets obtained from the separate evaluations of different benchmarks (i.e. applications) into a single Pareto set from which the designer can choose the configuration that best suits his needs.

A. *Formulation of the problem*

In the general case we want to optimize the configuration parameters of a platform with respect to m objectives $(o_1, o_2, \ldots, o_m)$ running q benchmarks $(b_1, b_2, \ldots, b_q)$.

A trivial test that can be done is to search Pareto optimal points for application domain optimization among the ones already obtained from single-application optimization, running a new exploration with the other benchmarks. This approach, we called Re-Evaluation of Pareto (REP), can be very fast as we need to simulate only a small number of points belonging to the Pareto set, instead of re-exploring the whole design space again. It works as follows: first we take the configurations belonging to the Pareto sets obtained from a single-benchmark optimization for b_i and run a re-evaluation with all the other benchmarks b_j in cascade, propagating to the next benchmark only the Pareto optimal points obtained at each stage. At the end of this process we will have a reduced Pareto set that contains only those configurations that are Pareto optimal for every benchmark. The process is then repeated taking a different starting Pareto set for all the other benchmarks. Taking the minimum Pareto front (in a $q \times m$ dimensional space) from the union of the resulting multi-sets produces the final application domain Pareto set.

While this approach guarantees to find at least a solution, it is clear that the order in which we run the re-evaluations will affect the final outcome. Let us suppose we have 2 configurations c_i and c_j both present in the starting Pareto set so that $c_i \succ_{b_k} c_j$ and $c_j \succ_{b_l} c_i$, that is, c_i dominates c_j when running b_k and c_j dominates c_i when running b_l. If we first run through b_k then c_j will be removed from the Pareto set while c_i will be kept, and vice-versa. Furthermore the number of re-evaluations needed to explore all the possibilities is $q \times (q-1)$, which, being $O(N^2)$ in complexity, can rapidly grow too large. Unfortunately, even if this approach is still feasible with $q = 2$ as it requires only 2 re-evaluations, the quality of its results with respect to the single-benchmark run can be disappointing.

The main problem with this approach is that it lacks generality because it examines only a small part of the design space obtained from previous explorations. A robust application domain methodology will require to carry out a new exploration that takes into considerations all the objectives from the beginning. Nevertheless it may be a quick way of discovering if the problem under exam needs a more powerful approach. As shown in Section IV-B we obtained interesting results with this method under well defined conditions.

Let us state the DSE problem again taking into considerations the application domain issue. Let m be the number of objectives to be optimized and q the number of benchmarks $b \in \mathcal{B}$ taken into consideration. We define $r = m \times q$ to be the number of total objectives to be optimized. An evaluation function $E : \mathcal{C}^*(S) \times \mathcal{B}^q \longrightarrow \Re^r$ is a function that associates each feasible configuration of the design space S with an r-tuple of values corresponding to the objectives to be optimized *for each* application belonging to the set of benchmarks $\mathcal{B}$.

This way we are stating the will to optimize the parameters *concurrently* for all the benchmarks. Stating the problem in this manner raises the new question as what is the relationship between the newly defined objectives. While it is consolidated and universally acknowledged that performance indexes in traditional DSE are usually conflicting (e.g. power and performance) and related in difficult to express ways, this is not necessarily true for all of the newly defined r objectives $o_1, o_2, \ldots, o_r$. Some of them may be linearly dependent, as an example we can take into consideration the silicon area occupied by a configuration which is clearly unaffected by the benchmark running on it. Thus we may employ some knowledge about the objectives to reduce the space and help the exploration algorithm in its task. We can thus define an objective reduction function $F : \Re^r \longrightarrow \Re^{\overline{r}}$ that maps the r-dimensional objective space defined for the application domain exploration to a smaller $\overline{r}$-dimensional space.

Another reason for applying this reduction function comes from the exploration algorithm we want to employ, that is MOEA. In fact, adding unneeded dimensions to the problem can damage the algorithm's performances because of the creation of new local minima, the added complexity in Pareto extraction and the interference in distance evaluation for niching techniques.

A simple reduction function F can map the r separate objectives again onto m compound objectives using a linear combination (i.e. mean value) of every original objective across the q benchmarks. This way traditional exploration techniques can be employed unaltered, as the dimensionality of space to be explored remains the same. The linear combination can use different coefficient values to bias the search towards more important benchmarks giving them a bigger weight, or evenly distribute the weights among the benchmarks (weight sum).

B. Proposed approach

While the aforementioned method can give good results, it implicitly presupposes that the same kinds of objective for different benchmarks (e.g. execution time) are in a somewhat linear relationship among them. If this is not the case the results can become negatively affected by this "forced" relationship imposed to the objectives, as an optimization of a linear combination of the objectives can never reach points in the non-convex part of the Pareto. Because the problem considered is inherently coarse grained, "steps" and "holes" that make the Pareto front non-convex are frequent.

To avoid this problem the designer has to identify which objectives can be aggregated, as they exhibit linear behavior, and which have to be taken into consideration separately. A statistical analysis can be done on the objectives using correlation to find how linear is the relationship between objectives. In probability theory and statistics, correlation (often measured as a correlation coefficient) indicates the strength and direction of a linear relationship between two random variables. In general statistical usage, correlation refers to the departure of two variables from independence and is defined as $Corr(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}}$, where Cov is the *covariance* and Var is the *variance* of the random variables.

The correlation is 1 in the case of an increasing linear relationship, -1 in the case of a decreasing linear relationship, and some value in between in all other cases, indicating the degree of linear dependence between the variables. The closer the coefficient is to either -1 or 1, the stronger the correlation between the variables. If the variables are independent then the correlation is 0, but the converse is not true because the correlation coefficient detects only linear dependencies between two variables. So, aggregating only strongly correlated objectives we are keeping on the conservative side.

To evaluate the degree of correlation between the objectives we have to randomly sample the design space and consider the objectives as random variables. The number of samples needed varies according to the precision we require and to the correlation value we get: the higher the correlation, the smaller the number of samples needed. Using Fisher's Z transformation to get the confidence intervals for correlation values, as we are only interested in large or very large correlations, 100 or 200 samples are usually good enough.

In general for r objectives we will obtain an $r \times r$ correlation matrix Q whose i,j element is $Corr(o_i, o_j)$. If $Q_{i,j}$ is high, i.e. over a defined threshold (in our work we found that the suitable value is 0.8), we can merge the two objectives using a linear weight sum to represent them without loosing generality.

Algorithm 1: application domain (multi-benchmark) evaluation algorithm

```
1  extract configurations from GA population;
2  foreach i in benchmarks do
3      split configuration_space among processes;
4      foreach j in processes do
5          send benchmark i to processor j;
6          send configuration_space part j to processor j;
7          simulate_space;
8          collect results;
9      end
10     merge results;
11 end
12 update objective values of GA population;
```

Whichever the reduction function applied, the exploration algorithm from this point on retains the common structure sketched in Algorithm 1, and only differs in how results are merged. Obviously also the MOEA evolution and fitness updating are different as the number of objectives varies. The application domain evaluation algorithm is executed at every generation of the MOEA, to evaluate the newly created population. Using a master-slave approach, the algorithm iterates over the selected benchmarks, for every benchmark the master splits the configuration space to be executed in N parts, where N is the number of processes, and sends the configurations to be evaluated to the slaves together with the benchmark to be used. After every slave (and the master itself) completes the simulations, results are collected and merged according to the chosen strategy. For aggregated objectives the weighted sum is calculated from its components, while separated objectives can be directly used.

At the end of the exploration a cumulative Pareto will be obtained. This Pareto set of configurations is directly usable by the decision maker because, even if the MOEA used objectives that are in some part composite values of the real ones, all the original r objectives were evaluated and saved anyway. To compare the results obtained by the application domain approach with the ones from the single-benchmark case, the Pareto set needs to be "projected" on the objective space of every benchmark, so we need to take into consideration only the subset of the objectives relative to that specific benchmark.

III. Simulation Environment

To evaluate and compare the performance indexes of different architectures for a specific application, one needs to simulate the architecture running the code of the application. In this work we use the Epic Explorer [7] simulation environment. To make architectural exploration possible both the compiler and the simulator have to be retargetable. Trimaran [8] provides these tools and thus represents the pillar of the Epic Explorer. Epic Explorer is an environment that not only allows us to evaluate any instance of a platform in terms of area, performance and power, but also implements various techniques for exploration of the design space. To achieve high levels of parallelism, the compiler should be allowed to schedule instructions belonging to different basic blocks in parallel. This extremely complex task, that involves moving instructions over branches (speculative execution), makes the presence of several basic blocks an intrinsic limit to the amount of parallelism that can be extracted from the application code. A solution to overcome this limit and achieve an effective ILP-oriented scheduling is to extend the scope of action of the compiler to wider regions of code such as the hyperblocks. Thus, hyperblock formation allows the VLIW compiler to act on much larger regions of code and thus to radically transform the code in order to maximize ILP.

In this section, we briefly describe the parameterized VLIW platform used as testbed for the experiments, the general evaluation flow along with the high-level estimation models used to evaluate the performance indexes to be optimized and the set of applications used as benchmarks.

A. Reference Architecture

The parameterized system architecture used in this work is based on HPL-PD [9] which is a parametric processor meta-architecture designed for research in instruction-level parallelism of EPIC/VLIW architectures. The HPL-PD opcode repertoire, at its core, is similar to that of a RISC-like load/store architecture, with standard integer, floating point (including fused multiply-add type operations) and memory operations.

Architectural parameters can be classified in three main categories: *register files*, *functional units* and *memory subsystem*. The design space is shown in Table I.

Table I
DESIGN SPACE OF THE PARAMETERIZED VLIW BASED SYSTEM ARCHITECTURE.

Parameter	Parameter space
Integer Units	1,2,3,4,5,6
Float Units	1,2,3,4,5,6
Memory Units	1,2,3,4
Branch Units	1,2,3,4
GPR/FPR	16,32,64,128
PR/CR	32,64,128
BTR	8,12,16
L1D/I cache size	1KB,2KB,...,128KB
L1D/I cache block size	32B,64B,128B
L1D/I cache associativity	1,2,4
L2U cache size	32KB,64KB...,512KB
L2U cache block size	64B,128B,256B
L2U cache associativity	2,4,8,16
Space size	7.739×10^{10}

Table II
EVALUATION TIME FOR A SIMULATION (COMPILATION + EXECUTION) FOR SEVERAL MULTIMEDIA BENCHMARKS ON A OPTERON 2.6 GHZ LINUX WORKSTATION.

Benchmark	Description	Input size (KB)	Evaluation time (sec)
`wave`	Audio Wavefront computation	625	7.7
`jpeg-codec`	jpeg compression and decompression	128	33.2
`mpeg2-dec`	MPEG-2 video decoding	400	143.7
`adpcm-enc`	Adaptive Differential Pulse Code Modulation speech encoding	295	22.6
`adpcm-dec`	Adaptive Differential Pulse Code Modulation speech decoding	16	20.2
`fir`	FIR filter	64	9.1

IV. EXPERIMENTAL RESULTS

In this section we perform an evaluation of the proposed framework with an exploration of the parameterized VLIW based system architecture being studied. In addition, they all demonstrate that DSE is a computationally intensive task, in fact even using efficient exploration strategies proposed in literature, simulation times can become a bottleneck in the design flow. In [10] an extensive analysis aimed at showing the scalability of a single-application DSE in an HPC environment, was carried out demonstrating that the use of a combination of statistical and/or machine learning approaches in HPC environments are able to reduce of $1000x$ the exploration time. Table II reports the computational effort needed for one evaluation (i.e. simulation) of just a single system configuration for several media and digital signal processing application benchmarks taken from the MediaBench benchmark suite [11].

A. Correlation analysis

Basing the decision on a benchmark profile analysis work, four benchmarks were chosen for multi-benchmark exploration because of their features and complexity: `rawcaudio`, `rawdaudio`, `fir`, `jpeg`. The benchmarks were taken in couples for multi-benchmark exploration. The first two (`rawcaudio` and `rawdaudio`) were chosen because they may represent a typical application domain, for example a VoIP gateway. The other two (`fir` and `jpeg`) were chosen because their profile hinted they need very different architectures to perform well, so it would be a good challenge for the algorithm. This choice was validated through a correlation analysis.

For the multi-benchmark optimization problem we decided to take into consideration the power dissipation and execution time objectives, as the area only depends from the configuration. The test were executed for every benchmark with hyperblock formation disabled and enabled, we will refer to them respectively as *Normal* and *Hyperblock* compilation profiles.

For every benchmark 10000 random configurations were

Table III
CORRELATION FOR THE POWER OBJECTIVE, AND FOR THE EXECUTION TIME OBJECTIVE WHEN HYPERBLOCK FORMATION IS DISABLED AND ENABLED.

Hyperblock formation disabled								
	Power				Execution time			
Benchmark	`rawc`	`rawd`	`fir`	`jpeg`	`rawc`	`rawd`	`fir`	`jpeg`
`rawc`	#	**0.9810**	0.9815	0.9160	#	**0.5802**	0.8907	0.3636
`rawd`	**0.9810**	#	0.9455	0.9225	**0.5802**	#	0.4697	0.5998
`fir`	0.9815	0.9455	#	**0.8838**	0.8907	0.4697	#	**0.2355**
`jpeg`	0.9160	0.9225	**0.8838**	#	0.3636	0.5998	**0.2355**	#

Hyperblock formation enabled								
	Power				Execution time			
Benchmark	`rawc`	`rawd`	`fir`	`jpeg`	`rawc`	`rawd`	`fir`	`jpeg`
`rawc`	#	**0.8790**	0.9216	0.9172	#	**0.6401**	0.8013	0.9061
`rawd`	**0.8790**	#	0.8613	0.9046	**0.6401**	#	0.4770	0.7113
`fir`	0.9216	0.8613	#	**0.8857**	0.8013	0.4770	#	**0.7200**
`jpeg`	0.9172	0.9046	**0.8857**	#	0.9061	0.7113	**0.7200**	#

simulated. Even if much less samples would have been enough, this way we obtained extremely narrow confidence intervals. We can estimate with 99% confidence that the actual correlation values are in a range of ± 0.01 from their nominal value. In practical application a very few simulations (e.g. about one hundred) will be enough to test the correlation between the objectives.

Table III reports the correlation matrix for the power objective and for the execution time objective of the four benchmarks when hyperblock formation is enabled and disabled. From these results it is clear that the power objective is highly correlated even across different benchmarks, while the execution time objective is from medium to lowly correlated, depending on the benchmarks taken into consideration. Thus three objectives were defined for the correlation based approach, the execution times of the two benchmarks separately and a weighted sum of their power. The results also confirm the hypothesis that `fir` and `jpeg` have very different requirements, as their correlation values are the lowest. The power objective plot is very narrow with a positive slope, the time objective plot instead is more spread apart in accord with correlation values, this confirms that the power objective is almost linearly related. Hyperblock formation introduces some changes. As reported in Table III the power objective is still highly correlated. The time objective this time is more correlated, even if not as much as power. Hyperblock in fact induces a higher exploitation of the available functional units, so the execution time is now more influenced by the underlying hardware.

From this data it is difficult to state how many objectives we should use for the hyperblock case, as the times are correlated, but not too much. We chose to use the same strategy as for the normal case and compare the results obtained with WS. In fact the 3D approach would be identical to WS if all the objectives were considered as correlated and thus aggregated.

B. Empirical tests and results

This section compares the resulting Pareto sets obtained with the various approaches discussed in Section II. The

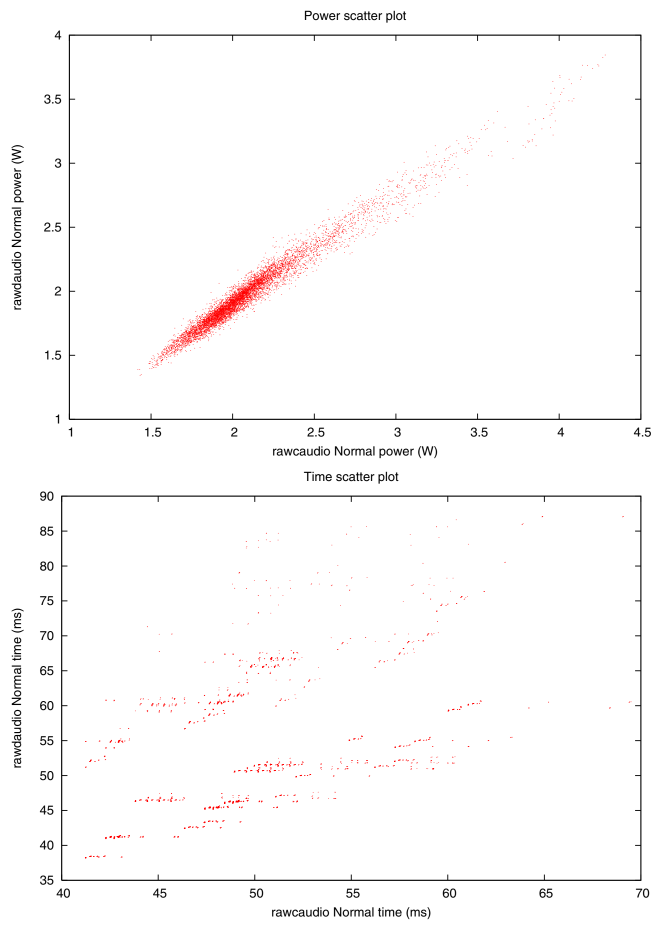
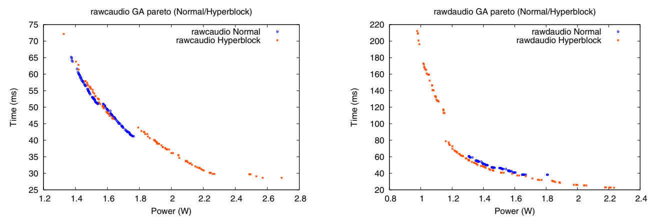


Figure 1. Scatterplots for the rawcaudio/rawdaudio benchmarks: Power and Execution Time

SPEA2 parameters were fixed as follows: the internal and external population for the evolutionary algorithm were set as comprising 30 individuals, using a crossover probability of 0.8 and a mutation probability of 0.1. These values were set in order to maximize the performance over the short run experiments, following the indications given in [6], where the convergence times and accuracy of the results were evaluated with various crossover and mutation probabilities.

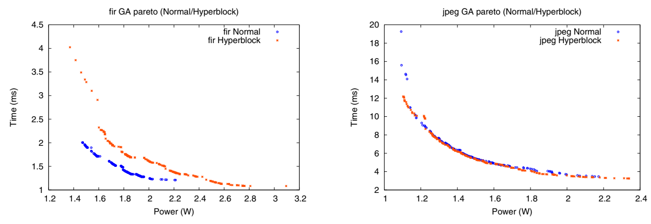
For the sake of brevity we will refer to the Pareto sets as follows: MOEA is obtained from single-benchmark exploration, REP is obtained using `rep` from the single-benchmark Pareto set (MOEA) of the corresponding coupled benchmark, WS using the multi-benchmark approach and the weight sum aggregation, 3D using the correlation based approach and 3 objectives as discussed in Section IV-A. The WS weights were chosen without bias as 0.5. The same applies to the power objective in the 3D approach (i.e. mean values were employed).

Comparing results in a multi-dimensional space is a complex task by itself that has already drawn a lot of



(a) (b)

Figure 2. rawcaudio and rawdaudio reference MOEA Pareto sets.



(a) (b)

Figure 3. fir and jpeg reference MOEA Pareto sets

Table IV
SUMMARY METRICS

	Normal			Hyperblock		
	REP	WS	3D	REP	WS	3D
	`rawdaudio`					
$\mathcal{D}$	0.333131	0.457594	**0.0882834**	33.1124	10.3939	**4.76823**
Cov	0	0.112903	**0.177419**	0.047619	**0.295238**	0.0952381
$Cont$	0.0746269	0.188406	**0.242647**	0.107143	**0.486207**	0.181034
$Entr$	0.281217	0.32845	**0.368619**	0.20414	**0.385881**	0.303914
	`rawcaudio`					
$\mathcal{D}$	0.176271	0.259744	**0.0595826**	1.18928	**0.175876**	0.822846
Cov	0.182609	**0.286957**	0.234783	0.177083	**0.364583**	0.197917
$Cont$	0.290076	0.295652	**0.408784**	0.294643	**0.595395**	0.384
$Entr$	0.262983	0.294741	**0.364141**	0.337999	**0.446739**	0.35621
	`fir`					
$\mathcal{D}$	0.0197497	0.0346367	**0.00460449**	0.533096	**0.14934**	0.310364
Cov	0.12844	0.00458716	**0.316514**	0	0.0110497	**0.0441989**
$Cont$	0.194915	0.03125	**0.43346**	0.0163044	0.0427808	**0.0748663**
$Entr$	0.209722	0.19324	**0.360886**	0.178013	**0.257882**	0.23842
	`jpeg`					
$\mathcal{D}$	2.29449	1.53472	**0.430155**	0.658617	**0.251496**	0.580902
Cov	0	**0.368098**	0.214724	0.172414	0.413793	**0.423645**
$Cont$	0	**0.533937**	0.309783	0.0718232	**0.458716**	0.327586
$Entr$	0.122465	**0.415031**	0.31516	0.179608	**0.393319**	0.329214

attention from the scientific community. There is no consensus on how to compare Pareto sets, but a plethora of approaches have been proposed in literature [12]. It is established [13] that more than one metric is necessary to evaluate the performance of MOEA, among the available ones the following performance indexes were used : Size of the dominated space, Coverage, Contribution, and Entropy. These metrics were chosen because they are meant to work for high dimensional spaces, they are not sensitive to scaling problems, and they measure different features of the Pareto sets. While they are not without defects [13], they should give us a general view of how well the various approaches perform.

Figures 2 and 3 present the reference Pareto fronts used for testing, obtained through single benchmark exploration. We can also see how hyperblock formation influences the objectives: the `rawcaudio` and `rawdaudio` benchmarks, being complex multimedia applications, heavily exploit the added ILP introduced by hyperblock formation. This allows faster execution times to be achieved and also lower power consumption in some cases.

Looking at Figure 2 we can see that the hyperblock Pareto set is far wider than the normal one. On the other side, in Figure 3 the simple `fir` filter is negatively affected, even though some boundary solution can only be obtained with hyperblock formation enabled. The `jpeg` codec is only slightly positively influenced by hyperblock, due to the small default input used (128 KB).

We will now show summary tables of the performance indexes for the benchmarks. Every metric is applied to the set obtained with the specified approach compared to the set obtained with the single-benchmark EA. Best values are highlighted in bold. Even if numerical data is useful, visual data (where available) can convey to the user more information at a glance. Therefore some plots comparing the Pareto fronts obtained are shown next.

Form Figure 4 we can visually see that the 3D approach is actually better than the WS in approximating the `rawcaudio` Pareto. Not only the 3D points cover a bigger fraction of the reference Pareto and are more evenly distributed than in WS, as detailed in Table IV, but we also find some new boundary solutions. For `fir` the same considerations still hold, even if this time the difference is sharper. Almost all of the solutions of WS are dominated by 3D. Furthermore 3D is much more widely spread and very close to the reference Pareto, even outclassing in some points. At the same time REP fails to reach a good performance with `jpeg` because of the innate difference from the coupled benchmark used as source (`fir`).

In the hyperblock version in the same Figure 4 instead, images confirm what Table IV said: the WS approach is better. Better coverage, especially on the left (low power) side, and a wider front can be found. The reason is the higher correlation for the time objective that favors WS. The 3D approach offers a good approximation annyway, confirming its high generality.

Metrics sometimes fail to accurately represent the situation, as for `rawdaudio`. Referring again to Figure 4, REP focuses on the left (low power) side of the Pareto. WS approach finds a good approximation only for the low Pareto section (low time). The linear combination has thus excluded from the search some points of interest. Conversely 3D finds good tradeoffs for the whole surface even if the point density is inferior.

Finally, with regard to `jpeg`, despite what found in Table IV, WS is just marginally better than 3D: on the left side WS gets better convergence while on the right side 3D finds new boundary solutions. WS Pareto front is more densely populated while still being evenly spread, this may give it an edge over 3D in the performance indexes. In

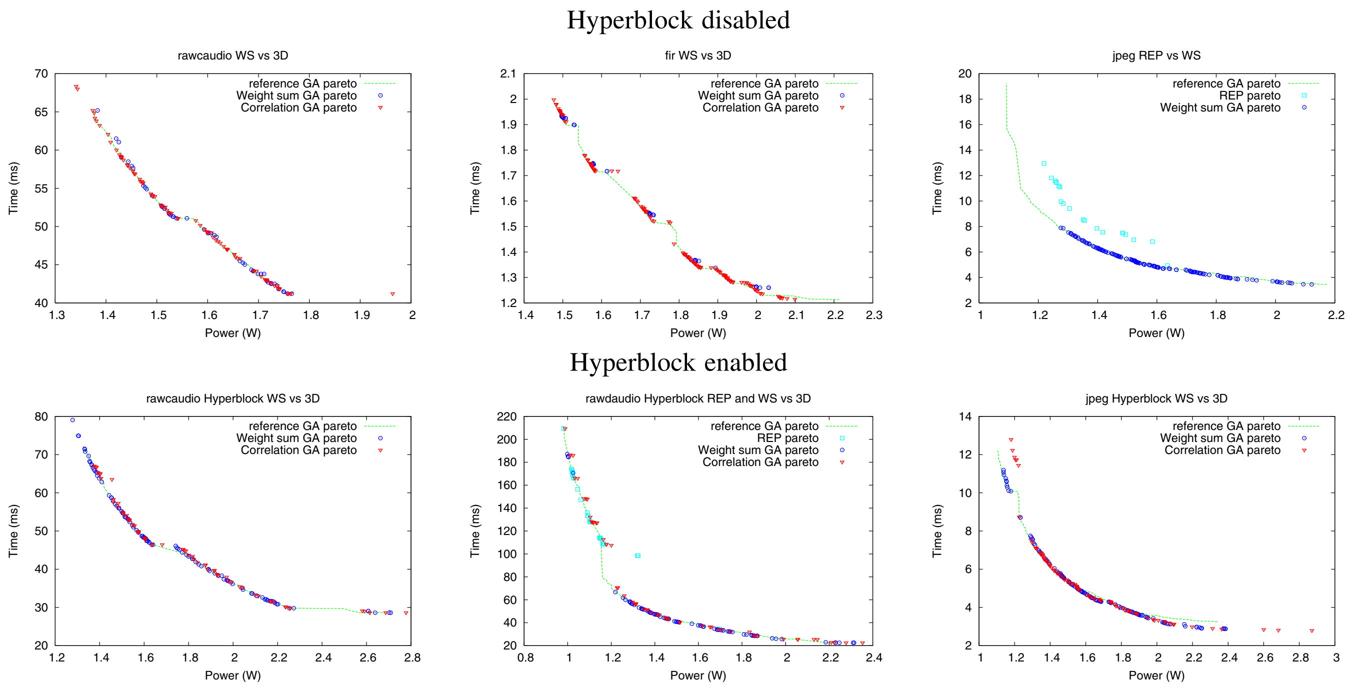


Figure 4. Pareto fronts computed by WS and 3D for different benchmarks with hyperblock formation disabled (top) and enabled (bottom).

both cases however, the diversity introduced by the multi-benchmark optimization has surprisingly beneficial results, allowing to find solutions that are better than the single-benchmark ones.

Overall results evidence that the proposed approach gives very good results both in terms of convergence and diversity. The correlation based approach (3D) seems better suited for the normal compilation profile, while the weight sum approach (WS) performs better with hyperblock formation enabled. The reason is that hyperblock enabled objectives are more correlated, therefore the WS aggregating approach finds a good approximation while at the same time reducing the objective space dimension. This confirms that a reduction in the space dimensionality can help the EA to converge faster. At the same time the 3D approach proved to have general applicability and give good results also for the hyperblock case where objective identification was hard (whichever the number of objectives chosen), so its application is recommended.

V. Conclusions and Future Works

This work has proposed a application domain multi-objective methodology for design space exploration of embedded systems, employing genetic algorithms to drive the exploration in which every application objective is treated as a new dimension in the solution space. Furthermore, the proposed methodology has been analyzed through extensive testing and proven to be effective compared to the single-benchmark case. Future developments will include DSE with more than two application benchmarks and with consideration for more advanced compilation strategies. Also the use of different weights in the aggregating approaches will be examined.

References

[1] M. Arnold and H. Corporaal, "Designing domain-specific processors," in *CODES '01: Proceedings of the ninth international symposium on Hardware/software codesign*. New York, NY, USA: ACM, 2001, pp. 61–66.

[2] P. Schaumont and I. Verbauwhede, "Domain-specific codesign for embedded security," *Computer*, vol. 36, no. 4, pp. 68–74, 2003.

[3] I. Verbauwhede, P. Schaumont, C. Piguet, and B. Kienhuis, "Architectures and design techniques for energy efficient embedded dsp and multimedia processing," in *DATE '04: Proceedings of the conference on Design, automation and test in Europe*. Washington, DC, USA: IEEE Computer Society, 2004, p. 20988.

[4] W. Fornaciari, D. Sciuto, C. Silvano, and V. Zaccaria, "A sensitivity-based design space exploration methodology for embedded systems," *Design Automation for Embedded Systems*, vol. 7, pp. 7–33, 2002.

[5] T. Givargis, F. Vahid, and J. Henkel, "System-level exploration for Pareto-optimal configurations in parameterized System-on-a-Chip," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 10, no. 2, pp. 416–422, Aug. 2002.

[6] G. Ascia, V. Catania, and M. Palesi, "A multi-objective genetic approach for system-level exploration in parameterized systems-on-a-chip," *IEEE Transactions on Computer-Aided*

Design of Integrated Circuits and Systems, vol. 24, no. 4, pp. 635–645, Apr. 2005.

[7] G. Ascia, V. Catania, M. Palesi, and D. Patti, “EPIC-Explorer: A parameterized VLIW-based platform framework for design space exploration,” in *First Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia)*, Newport Beach, California, USA, Oct. 3–4 2003, pp. 65–72.

[8] “An infrastructure for research in instruction-level parallelism,” http://www.trimaran.org/.

[9] V. Kathail, M. S. Schlansker, and B. R. Rau, “HPL-PD architecture specification: Version 1.0,” Compiler and Architecture Research HP Laboratories Palo Alto HPL-93-80, Tech. Rep., 2000.

[10] V. Catania, G. De Francisci Morales, A. G. Di Nuovo, M. Palesi, and D. Patti, “High performance computing for embedded system design: A case study,” in *Proceedings of 11th EUROMICRO CONFERENCE on DIGITAL SYSTEM DESIGN Architectures, Methods and Tools*. IEEE Computer Society Press, 2008.

[11] C. Lee, M. Potkonjak, and W. Mangione-Smith, “MediaBench: a tool for evaluating and synthesizing multimedia and communicatons systems,” *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*, pp. 330–335, 1997.

[12] T. Okabe, Y. Jin, and B. Sendhoff, “A critical survey of performance indices for multi-objective optimisation,” *Evolutionary Computation, 2003. CEC’03. The 2003 Congress on*, vol. 2, 2003.

[13] J. Knowles, L. Thiele, and E. Zitzler, “A tutorial on the performance assessment of stochastive multiobjective optimizers,” Computer Engineering and Networks Laboratory, ETH Zurich, Tech. Rep. 214, Feb. 2006.