ReSPIR: A Response Surface-Based Pareto Iterative Refinement for Application-Specific Design Space Exploration

Gianluca Palermo, Member, IEEE, Cristina Silvano, Senior Member, IEEE, and Vittorio Zaccaria, Member, IEEE

Abstract—Application-specific multiprocessor systems-on-chip (MPSoCs) are usually designed by using a platform-based approach, where a wide range of customizable parameters can be tuned to find the best tradeoff in terms of the selected figures of merit (such as energy, delay, and area). This optimization phase is called design space exploration (DSE), and it usually consists of a multiobjective optimization problem with multiple constraints. So far, several heuristic techniques have been proposed to address the DSE problem for MPSoC, but they are not efficient enough for managing the application-specific constraints and for identifying the Pareto front. In this paper, an efficient DSE methodology for application-specific MPSoC is proposed. The methodology is efficient in the sense that it is capable of finding a set of good candidate architecture configurations by minimizing the number of simulations to be executed. The methodology combines the design of experiments (DoEs) and response surface modeling (RSM) techniques for managing system-level constraints. First, the DoE phase generates an initial plan of experiments used to create a coarse view of the target design space to be explored by simulations. Then, a set of RSM techniques is used to refine the simulation-based exploration by exploiting the application-specific constraints to identify the maximum number of feasible solutions. To trade off the accuracy and efficiency of the proposed techniques, a set of experimental results for the customization of a symmetric shared-memory on-chip multiprocessor with actual workloads has been reported in this paper.

Index Terms—Application-specific processors, chip multiprocessors, design space exploration.

I. INTRODUCTION

NOWADAYS, multiprocessor systems-on-chip (MPSoCs) and chip multiprocessors (CMPs) represent the de facto standard for both embedded and general-purpose architectures. In particular, customizable MPSoCs supported by parallel programming have become the dominant computing paradigm for application-specific processors. In fact, they represent the best compromise in terms of a stable hardware platform which is software programmable, thus customizable, upgradable, and extensible. In this sense, the MPSoC paradigm minimizes the risk of missing the time-to-market deadline while ensuring greater efficiency due to architecture customization and software compilation techniques. For these architectures, the platform-based design approach [1] is widely used to meet time-to-market constraints. In this scenario, configurable simulation models are used to accurately tune the on-chip architectures and to meet the target application requirements in terms of performance, battery lifetime, and area.

The design space exploration (DSE) phase is used to tune the configurable system parameters, and it generally consists of a multiobjective optimization (MOO) problem. The DSE problem consists of exploring a large design space consisting of several parameters at system and microarchitectural levels. Although several heuristic techniques have been proposed so far to address this problem, they are all characterized by low efficiency to identify the Pareto front of feasible solutions. Among those heuristics, evolutionary or sensitivity-based algorithms represent the most notable state-of-the-art techniques.

The emphasis on the efficiency of the DSE algorithms is due to the fact that, in computer architecture research and development, simulation still represents the main tool to predict the performance of alternative architectural design points. If we consider a cycle-accurate system-level simulation, it requires a lot of simulation time, and the exploration of the design alternatives can exceed the practical limits. Furthermore, the growing trend toward CMP architectures amplifies this problem because the simulation speed linearly decreases by increasing the number of cores to be simulated (as shown in Fig. 1). While, from the simulation point of view, the use of statistical sampling techniques seems to represent the best solution [2], from the DSE point of view, efficient solutions that reduce the number of architectural alternatives to be analyzed are still missing.

In this paper, an efficient DSE methodology is proposed to tune MPSoCs and CMPs subject to application-specific constraints to identify the Pareto front of feasible solutions. The methodology combines the design of experiments (DoEs) and response surface modeling (RSM) techniques for managing system-level constraints. First, the DoE phase generates an initial plan of experiments used to create a coarse view of the target design space to be explored by simulations. Then, a set of RSM techniques is used to refine the simulation-based exploration by exploiting the application-specific constraints to identify the maximum number of feasible solutions.
constraints (such as total system area, energy consumption, or throughput). The overall goal is to minimize the number of simulations to be executed during the exploration phase. The methodology is based on the design of experiments (DoE) and response surface modeling (RSM) techniques. First, the DoE phase defines an initial plan of experiments to create a coarse grain view of the target design space. Second, a set of RSM techniques is used to refine the exploration and to identify a set of feasible configurations. This process is iteratively repeated to derive a set of Pareto points. Then, a technique is proposed to deal with the application-specific constraints expressed at the system level.

To the best of our knowledge, while there have already been some applications of DoEs and RSM techniques to the field of performance analysis and optimization, the work proposed in this paper represents the first in-depth comprehensive application of the DoE and RSM techniques to the field of multiobjective DSE for MPSoCs characterized by application-specific constraints.

This paper is organized as follows. Section II discusses the state of the art related to DSE, while Section III introduces the DSE problem. Section IV describes the application-specific DSE methodology proposed in this paper, and Section V validates the proposed methodology applied for the customization of an on-chip multiprocessor over a set of standard benchmarks. Finally, Section VI shows the application of the proposed methodology to design a customized MPEG decoder for an MPSoC architecture. Finally, Section VII contains some concluding remarks.

II. STATE OF THE ART

Several methods have been recently proposed in literature to reduce the DSE complexity by using traditional statistic techniques and advanced exploration algorithms.

Among the most recent heuristics for power/performance architectural exploration, we can find [5] and [6]. In [5], the authors compare Pareto simulated annealing, Pareto reactive taboo search, and random search exploration to identify energy-performance tradeoffs for a parametric superscalar architecture executing a set of multimedia kernels. In [6], a combined genetic–fuzzy system approach is proposed. The technique is applied to a highly parameterized SoC platform based on a very long instruction word processor in order to optimize both power dissipation and execution time. The technique is based on a strength Pareto evolutionary algorithm coupled with fuzzy system rules in order to speedup the evaluation of the system configurations.

State-of-the-art system performance optimization is presented in [7]–[10]. A common trend among those methods is the combined use of RSM and DoEs methodologies.

In [7], a radial basis function (RBF) has been used to estimate the performance of a superscalar architecture; the approach is critically combined with an initial training sample set that is representative of the whole design space in order to obtain a good estimation accuracy. Joseph et al. propose to use a variant of the Latin hypercube method in order to derive an optimal initial set. In [8] and [9], linear regression has been used for the performance prediction and assessment. Joseph et al. and Lee and Brooks analyze the main and interaction effects among the processor architectural parameters. In both cases, random sampling has been used to derive an initial set of points to train the linear model. A different approach is proposed in [10], where McKee et al. tackle performance prediction by using an artificial neural network (ANN) paradigm to estimate the system performance of a CMP.

In [11] and [12], a Plackett–Burman DoEs is applied to the system architecture to identify the key input parameters. The exploration is then reduced by exploiting this information. The approach shown in [11] is directed toward the optimization of a field-programmable-gate-array soft-core-based design, while, in [12], the target problem is more oriented to a single superscalar processor.

This paper represents a revised and extended version of the papers previously published in [13] and [14]. With respect to these works, this paper proposes an overall generalization and formalization of the proposed exploration strategy based on several RSMs and a refined technique to deal with the application-specific constraints.

III. INTRODUCTION TO DSE

The intellectual property (IP) reuse and platform-reconfigurability approaches are converging into a new design paradigm (platform-based design [1]), which is strongly influencing today’s automatic system synthesis. In this context, a microprocessor-based platform is composed of a number of IP blocks which are integrated, extended, and customized for a particular application. The IP-based design flow dramatically decreases the risk of subsystem integration and configuration errors, reducing up to 60% [15] the platform design time while achieving the highest quality of results in the implementation of the design.

In general, DSE consists of an optimization process which takes into account a typical set of IP parameters mainly associated with the memory subsystem configuration (e.g., cache size), the parallelism of the processor (e.g., number of processors and issue width), and the on-chip interconnect configuration. The optimization problem involves the minimization (maximization) of multiple objectives (such as latency, energy, area, etc.), making the definition of optimality not unique [16]. In fact, a system, which is the best from the performance point of view, can be the worst in terms of power consumption and vice versa.

In our context, the MOO targets a set of \( n \) system configuration parameters grouped on a configuration vector

\[
\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in X
\]  

(1)

where \( X \) is usually a finite discrete domain (subset of \( N_0^n \)). The MOO problem is defined as a set of \( m \) objective functions to be minimized (or maximized)

\[
\min_{\mathbf{x} \in X} \mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^m
\]  

(2)

Authorized licensed use limited to: Politecnico di Milano. Downloaded on December 1, 2009 at 12:38 from IEEE Xplore. Restrictions apply.
subject to a set of \( k \) constraints which, without loss of generality, can be expressed as

\[
e(x) = \begin{bmatrix} e_1(x) \\ \vdots \\ e_k(x) \end{bmatrix} \leq \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}.
\]

The set of all the admissible or feasible solutions of an optimization problem is called the feasible region, and it is defined as

\[
\Phi = \{ x \in X | e_i(x) \leq 0, 1 \leq i \leq k \}.
\]

In single-objective optimization problems, the feasible set is totally ordered according to the objective function \( f(x) \); thus, a single exact solution \( x \) does exist. However, when several objectives are involved, the feasible set is partially ordered by a dominance relation; thus, multiple exact solutions exist.

The multiple-solution set is called the exact Pareto set of the problem. In order to formally define the concept of Pareto set, we need to introduce the dominance relation. We say that \( f \) dominates \( g \) when

\[
f < g \iff \left\{ \begin{array}{l} f_i \leq g_i, \quad \forall i = 1, \ldots, m \\ f_i < g_i, \quad \text{for at least one } i \end{array} \right. \]  

(4)

Given a subset of feasible configurations \( \Omega \subseteq X \), we define the Pareto set \( \Psi \) associated to \( \Omega \) as

\[
\Psi(\Omega) = \left\{ x \in \Omega \cap \bigwedge x \in (\Phi \cap \Omega) \land \exists y \in \Psi(\Omega) \text{ s.t. } f(y) < f(x) \right\}.
\]

The Pareto set \( \Psi(\Omega) \) is the exact Pareto set of the problem. Based on set theory, the projection of \( \Psi(\Phi) \) in the objective space is called the exact Pareto front.

Whenever the considered solution set \( \Omega \) is a subset of the feasible solution space \( \Phi \), the Pareto set \( \Psi(\Omega) \) is called an approximate Pareto set of the problem.

Fig. 2(a) shows a feasible set of solutions \( \Omega = \Phi = \{ x_a, x_b, x_c \} \) for an unconstrained minimization problem for \( [f_1, f_2] \). Point \( x_b \) is dominated by point \( x_c \) since both \( f_1(x_b) \) and \( f_2(x_c) \) are greater than \( f_1(x_b) \) and \( f_2(x_b) \). In this case, we thus have \( \Psi(\Omega) = \{ x_a, x_b \} \).

MOO heuristics generally try to identify the approximate Pareto sets which are as close as possible to the exact Pareto set. For evaluating the quality of the approximate Pareto sets, a measure of the distance between the exact and the approximate Pareto sets should be introduced. In literature, many quality functions have been proposed to tackle this problem [17]; we present here the average distance from reference set (ADRS). The ADRS is used to measure the distance between the exact Pareto set \( \Pi = \Psi(\Phi) \) and the approximate Pareto set \( \Lambda = \Psi(\Omega) \) [18]

\[
\text{ADRS}(\Pi, \Lambda) = \frac{1}{|\Pi|} \sum_{x_R \in \Pi} \left( \min_{x_A \in \Lambda} \{ \delta(x_R, x_A) \} \right)
\]

where \( \delta \) is a measure of the normalized distance in the objective function space of two configurations

\[
\delta(x_R, x_A) = \max_{j=1,\ldots,m} \left\{ 0, \frac{f_j(x_A) - f_j(x_R)}{f_j(x_R)} \right\}.
\]

The ADRS is usually measured in terms of percentage; the higher the ADRS, the worst is \( \Lambda \) with respect to \( \Pi \).

A useful measure of the relative goodness of a Pareto front with respect to another is the coverage function \( \chi \). The coverage function is defined as the percentage of points of an approximated Pareto set \( \Lambda' = \Psi(\Omega') \) which is dominated by another Pareto set \( \Lambda'' = \Psi(\Omega'') \) [19]

\[
\chi(\Lambda'', \Lambda') = \left\{ \frac{|\{ x_A' \in \Lambda' : \exists x_A'' \in \Lambda'' \land x_A'' \prec x_A' \}|}{|\Lambda'|} \right\}.
\]

(8)

From the previous definition, it follows that \( \chi(\Lambda'', \Lambda') = 0 \) (i.e., the self-coverage of an approximated Pareto set is zero) and that \( 0 \leq \chi(\Lambda'', \Lambda') \leq 1 \). \( \chi(\Lambda'', \Lambda') = 1 \) corresponds to the case when all the configurations in \( \Lambda' \) are dominated by those in \( \Lambda'' \), whereas \( \chi(\Lambda'', \Lambda') = 0 \) represents the case when none of the points in \( \Lambda' \) are dominated by those in \( \Lambda'' \). Given the definition in (8), it follows that \( \chi(\Lambda'', \Lambda') \) is not necessarily equal to \( 1 - \chi(\Lambda', \Lambda'') \).

Fig. 2(b) shows an example computation of coverage \( \chi(\Lambda'', \Lambda') \) by considering two objective functions \( f_1 \) and \( f_2 \). Only two points \( (x_a, x_b) \) out of the three in \( \Lambda'' \) are dominated by those in \( \Lambda' \). The coverage \( \chi(\Lambda'', \Lambda') \) is thus \( 2/3 \) (~66%), while \( \chi(\Lambda', \Lambda'') \) is 0%.

IV. APPLICATION-SPECIFIC DSE METHODOLOGY

In this paper, we propose an application-specific DSE strategy leveraging DoEs and RSM techniques combined with a constraint handling heuristic. Given the objective functions associated to the system, the proposed methodology enables the efficient identification of an approximate Pareto set of the candidate architectures by minimizing the number of simulations of system configurations. This is a notable achievement since, nowadays, evaluating the objective function \( f(x) \) of a single system configuration \( x \) (being it either performance or power consumption) means hours or days of simulations under a realistic workload for complex SoCs.

The term DoEs [20] is used to identify the planning of an information-gathering experimentation campaign where a set of variable parameters can be tuned. In this paper, we define an experiment as the simulation of the system with the target parameter configuration.

The reason for DoEs is that, very often, the designer is interested in the effects of some parameter’s tuning on the system response. DoEs is a discipline that has very broad application across natural and social sciences and encompasses a set of techniques whose main goal is the screening and analysis of the system behavior with a small number of simulations.
Each DoE plan differs in terms of the layout of the selected design points in the design space. In this paper, four DoEs techniques (random, full factorial, central composite design, and Box–Behnken) have been selected to generate the initial set of design points to be analyzed.

RSM techniques are typically introduced to decrease the time due to the evaluation of the system-level objective function \( f(x) \) for each architecture \( x \). In fact, for applications of commercial interest, evaluating \( f(x) \) can involve one or more simulations which can take several hours, depending on the platform complexity and the system resources dedicated to the simulation.

A response surface model for the function \( f(x) \) is an analytical function \( r(x) \) such that

\[
 f(x) = r(x) + \epsilon
\]

where \( \epsilon \) is the estimation error. Typically, an appropriate RSM for \( f(x) \) is such that \( \epsilon \) has some desired statistical properties such as a mean of zero and small variance. The working principle of RSM is to use a set of simulations generated by DoE in order to build our response model of the system. A typical RSM-based flow involves a training phase, in which known data (or training set) are used to identify the RSM configuration, and a prediction phase, in which the RSM is used to forecast the unknown system response. RSMs are an effective technique for analytically predicting the behavior of the system platform without resorting to a system simulation. RSM-based techniques represent the kernel of the proposed methodology. In this paper, four techniques (linear regression, Shepard’s interpolation, neural networks, and RBFs) have been selected and tuned to build the response surface models.

A. Notation and Preliminary Definitions

Before introducing the methodology, let us present some notations and definitions used in the rest of this paper for the formal description of the proposed methodology.

As described before, our target problem is to find an approximate Pareto set of configurations \( x \) which minimizes a vector objective function \( f(x) \). In our problem domain, \( f(x) \) represents a vector of system metrics which are evaluated by system simulation; \( f(x) \) is, thus, a black-box function since there does not exist an exact analytical representation for it.

The proposed method is based on a number of archives of tuples \( (x, f(x)) \) referred to as \( F \) which are updated every time a successful simulation is carried out for a new configuration \( x \). We will use the symbol \( F_i \) to indicate the \( i \)th update to the archive \( F \) constructed in such way. Each archive represents the information about a set of configurations \( \Omega \) and the corresponding projection in the objective function space \( f(x) \in \Omega \).

On the other hand, as, during the optimization, the proposed methodology is based on the prediction of the system performance, the heuristics need to maintain also a set of archives of tuples \( (x, r(x)) \) referred to as \( R \), where \( r(x) \) is the estimate of the objective vector \( f(x) \) derived from the RSM model. Again, we will use the symbol \( R_i \) to indicate the \( i \)th update of the archive \( R \). In this case, each archive represents the information about a set of configurations \( \Omega \) and the corresponding projection into the approximate objective function space \( r(x) \in \Omega \), where \( r \) is expected to be close to \( f(x) \).

Based on the given definitions, we can easily extend (5) to the archives: Given an archive \( F_i \), the following equation (10) can be used to obtain all the tuples \((x, f(x))\) containing the Pareto set associated with the target problem. The result is a new archive

\[
\Psi(F_i) = \{ (x, f(x)) | x \in \Psi(\Omega) \}
\]

where \( \Omega \) is the set of configurations \( x \) contained in \( F_i \). A similar definition of the Pareto set can be done for \( R_i \).

B. How to Manage the Constraints

A key concept of the proposed methodology is how the Pareto sets are built. While conventional techniques filter the set of configurations by considering only the feasible solutions [see (5)], we allow promising nonfeasible solutions to be maintained in the archives throughout the optimization process; they will be discarded only at the end. We define a promising nonfeasible solution as a solution which violates as few constraints as possible with minimum penalties; in principle, promising nonfeasible solutions can lead to feasible solutions in the successive steps of the exploration process. This principle is borrowed from evolutionary heuristics, where even nonfeasible solutions can contribute to the determination of the final result.

Each nonfeasible solution \( x \) has a rank and a penalty. The rank \( \rho(x) \) of a nonfeasible solution is the number of constraints \( (\epsilon_k(x) \leq 0) \) which it violates. The penalty \( \pi(x) \) is the geometric average\(^1\) of the slacks with respect to the constraints

\[
\pi(x) = \prod_{\epsilon_k(x) > 0} \epsilon_k(x) \frac{1}{\pi(x)}.
\]

While the approximated solution given by the proposed heuristic will be filtered from the nonfeasible solutions, we redefine the concept of Pareto set (5) which is used internally in the algorithm as follows:

\[
\Psi_\phi(\Omega) = \{ x | (\exists y \in \Psi_\phi(\Omega) \text{ s.t. } f(y) < f(x)) \}
\]

where “\( \phi \)” is a constrained dominance operator. The constrained dominance operator is introduced to include in the Pareto set the most promising configurations which violate the application-specific constraints. We say that \( f(x) \) constraint dominates \( f(y) \) when

\[
 f(x) < f(y), \quad x \in \Phi \\
 f(y) < f(x), \quad y \notin \Phi; \quad y \in \Phi
\]

where “\( \ll \)” is defined as

\[
 f(x) \ll f(y) = \begin{cases} 
 f(x) < f(y), & x \in \Phi \\
 f(y) < f(x), & y \notin \Phi; \quad y \in \Phi
\end{cases}
\]

\(^1\) A geometric mean, unlike an arithmetic mean, tends to dampen the effect of very high or low values, which might bias the mean if a straight average (arithmetic mean) were calculated.
In other words, the “≺” operator combines the rank, penalty, and the pure dominance of the system responses of two non-feasible solutions into a single extended-dominance value. It is essentially used for not discarding the promising nonfeasible solutions which are good either from the point of view of the number of constraints violated or the overall penalty associated with them.

Then, the archive filtering for the Pareto sets [as expressed in (10)] is modified accordingly to reflect the new Pareto set definition

\[ \Psi_\delta(F_i) = \{ \langle x, f(x) \rangle \mid \text{s.t. } x \in \Psi_\delta(\Omega) \} \]  

where \( \Omega \) is the set of configurations \( x \) contained in \( F_i \). The dual definition of the Pareto set can be done for \( R_i \).

Fig. 3 shows an example of a set of solutions \( \Omega = \{ x_a, x_b, x_c \} \) for a constrained minimization problem of \([ f_1, f_2 ]\) with a single constraint. The constraint is such that \( \{ x_a, x_b \} \) are nonfeasible solutions, with \( \pi(x_a) = 2 \) and \( \pi(x_b) = 1 \). Although traditional Pareto dominance would have given \( \Psi(\Omega) = \{ x_c \} \), the constrained-Pareto dominance relation \( \triangleleft \) is such that

\[ \Psi_\delta(\Omega) = \{ x_a, x_b, x_c \} \]  

As a matter of fact, although \( x_c \) dominates \( x_b \) from the point of view of the problem objectives, it has a higher penalty \( \pi \) and the same rank \( \rho \). Thus, both \( \{ x_a, x_b \} \) are considered promising and not filtered out by \( \Psi_\delta \). Moreover, although point \( x_c \) is dominated by \( x_a \), it still belongs to the Pareto set \( \Psi_\delta(\Omega) \) since it is the only feasible solution.

C. Proposed Exploration Strategy

The proposed exploration strategy is called response surface-based Pareto iterative refinement (ReSPIR). The approach focuses on the concept of iterative simulation-based refinements of the approximate Pareto set derived from the predictions provided by the RSM model, starting from an initial DoE (see Fig. 4).  

The methodology is parametric in terms of DoE and RSM techniques to be used, as well as in terms of the maximum number of simulations to be run (see Algorithm 1).

\[ \text{Algorithm 1: The ReSPIR DSE Algorithm} \]

\[ 1: \text{nsim} = 0 \]
\[ 2: \text{Generate and run the simulations from DoE. Update nsim accordingly. Put results into } F_0. \]
\[ 3: \text{cov} = 100\% \]
\[ 4: F_1 = \emptyset \]
\[ 5: \text{while } (\text{cov} > 0) \land (\text{nsim} < \text{maxnsim}) \text{ do} \]
\[ 6: F_0 = F_0 \cup F_1 \]
\[ 7: \text{Train RSM with the content of } F_0 \text{ and compute a prediction } R_0 \forall x \in X \]
\[ 8: R_1 = \Psi_\delta(R_0) \]
\[ 9: \text{Generate and run the simulations associated with the configurations in } R_1. \text{ Update nsim accordingly. Put results into } F_1. \]
\[ 10: \text{cov} = \chi(\Psi_\delta(F_1), \Psi_\delta(F_0)) \]
\[ 11: \text{end while} \]
\[ 12: \text{return } \Psi_\delta(F_0) \text{ by pruning the nonfeasible configurations.} \]

Initially (step 2), the DoE plan is used to pick up the set of initial design configurations corresponding to the plan of simulations to be executed. This step provides an initial coarse view of the target design space by running the simulations to obtain the actual measurements \( f \) associated with \( F_0 \). In the next steps, \( F_0 \) represents the archive containing the significant information about all the architectural configurations simulated so far.

At the first iteration, provided that the \( \text{maxnsim} \) value is greater than the DoE size, the condition in step 5 is met, and the \text{while} loop body is entered. The RSM technique (step 7) is thus trained with the current archive \( F_0 \). The response surface model generates a prediction archive \( R_0 \) considering all the possible system configurations, which is then Pareto filtered at step 8, obtaining \( R_1 \), the intermediate Pareto set shown in Fig. 4. Then (step 9), the refining simulations associated with the intermediate Pareto set \( R_1 \) are run; the result is put into the intermediate archive \( F_1 \) containing the actual measurements.

The last step of the loop checks that an improvement of the solutions has been obtained in this iteration by computing the coverage between the two Pareto sets of \( \Psi_\delta(F_1) \) and \( \Psi_\delta(F_0) \).

We would like to remind that the coverage value \( \chi(\Psi_\delta(F_1), \Psi_\delta(F_0)) \) is the percentage of points of \( \Psi_\delta(F_0) \) which are dominated by \( \Psi_\delta(F_1) \) [see (8)]. In other words, the coverage \( \text{cov} \) measures how much the Pareto set \( F_1 \) improves
the current archive of solutions \( F_0 \). The coverage is updated at each iteration of the loop.

The termination condition states that the main loop body of the methodology is executed as far as the improvement \( \text{cov} \) is greater than zero and a maximum number of simulations \( \text{maxnsim} \) have not been reached. The latter condition allows for stopping the algorithm before it has reached a steady state, avoiding the case of long explorations. In other words, increasing the \( \text{maxnsim} \) parameter improves the quality of the solution but increases the methodology run time.

We remark that \( F_1 \) is the archive containing the \( \langle x, f(x) \rangle \) tuples of the Pareto points found with the RSM (contained in the intermediate Pareto set \( R_1 \)). In other words, the RSM predictions (computed in step 7) have been used to drive the new simulation campaign (step 9) for constructing \( F_1 \). In the subsequent iteration of the loop, \( F_1 \) will be merged with \( F_0 \) (step 6), and a new RSM will be computed.

Finally (step 12), \( F_0 \) is Pareto filtered by pruning all the non-feasible configurations; the resulting archive is the approximate solution of our DSE problem.

The following sections describe several DoE techniques and RSM methods used in the proposed DSE methodology. Some experimental results have also been reported to show how the performance characteristics of each strategy can vary with respect to the selected DoE and RSM.

D. Design of Experiments

The selection of the DoEs [20] presented in this paper has been guided by some considerations about the impact of parameter interaction on the system-level metrics [9], [22]. The selected random, Box–Behnken, central composite, and full factorial DoEs allow generating experiments to capture those interactions. We excluded from the analysis some DoEs such as Plackett–Burman [12] which are not suitable for constructing interaction-based or higher level linear models. We also excluded fractional factorial designs which, although efficient, require iterating over the experimental design to resolve aliasing problems [23]. We thus selected the following four DoEs to build our efficient DSE methodology.

1) Random: In this case, design space configurations are picked up randomly by following a probability density function (PDF). In our methodology, we will use a uniformly distributed PDF.

2) Full Factorial: In statistics, a two-level full factorial experiment is an experiment whose design consists of two or more parameters, each with discrete possible values or “levels,” and whose experimental units take on all possible combinations of the minimum and maximum levels for such parameters. Such an experiment enables the evaluation of the effects of each parameter on the response variable, as well as the effects of interactions between parameters on the response variable. If, for example, we have only two tunable parameters, i.e., cache size and associativity, where size \( \in \{2K, 4K, 8K, 16K, 32K\} \) and associativity \( \in \{2, 4, 8\} \), then a two-level full factorial design will be composed of the following design vectors: \( \langle \text{size, associativity} \rangle \in \{\langle 2K, 2 \rangle, \langle 2K, 8 \rangle, \langle 32K, 2 \rangle, \langle 32K, 8 \rangle\} \).

3) Central Composite Design: A central composite design [Fig. 5(a)] is an experimental design specifically targeted to the construction of response surfaces of the second order (quadratic) without requiring a three-level full or fractional factorial DoE.

The design consists of the following three distinct sets of experimental runs: 1) a two-level full or fractional factorial design; 2) a set of center points, i.e., experimental runs whose values of each parameter are the medians of the values used in the factorial portion; and 3) a set of axial points, i.e., experimental runs that are identical to the center points except for one parameter. In the general central composite design, this parameter will take on values both below and above the median of the two levels. In this paper, we use a face centered central composite design where the remaining parameters assume the upper and lower bounds of the parameter range.

Considering the previous example (cache size and associativity), a face centered central composite design is composed of the following design vectors:

\[
\{\langle 2K, 2 \rangle, \langle 2K, 8 \rangle, \langle 32K, 2 \rangle, \langle 32K, 8 \rangle, \\langle 8K, 4 \rangle, \langle 2K, 4 \rangle, \langle 32K, 4 \rangle, \langle 8K, 2 \rangle, \langle 8K, 8 \rangle\}. \tag{17}
\]

4) Box–Behnken: The Box–Behnken design is suitable for quadratic models where parameter combinations are at the center of the edges of the process space in addition to a design with all the parameters at the center [Fig. 5(b)]. The main advantage is that the parameter combinations avoid taking extreme values taken at the same time (in contrast with the central composite design). This may be suitable to avoid singular points in the generation of the response surface, which would deteriorate it. Considering the previous example (cache size and associativity), a Box–Behnken design is composed of the following design vectors:

\[
\{\langle 8K, 4 \rangle, \langle 2K, 4 \rangle, \langle 32K, 4 \rangle, \langle 8K, 2 \rangle, \langle 8K, 8 \rangle\}. \tag{18}
\]

E. Response Surface Methods and Model Selection

As explained in Section IV, we introduce RSM to provide an analytical representation \( r(x) \) of the given vector objective function \( f(x) \). To achieve this task, we consider each vector component of the objective function \( f(x) \) separately by individuating a suitable \( r(x) \) for each \( f_i(x) \).

Each RSM \( r(x) \) is tuned/trained with a set of observations derived from an actual simulation of a system configuration \( x \). We will use the symbol \( y_k \) to indicate the \( k \)th observation of

![Graphical representations in the parameter space of the central composite and Box–Behnken DoEs.](image-url)
the target objective function obtained by simulating the $k$th configuration (or $x_k$).

In this section, we present the four RSMs used in the proposed methodology and the corresponding phases of the model selection process. The model selection is very important to identify a suitable model configuration by maximizing the accuracy and minimizing the associated complexity.

The RSM models used in this paper are specifically directed toward predicting the execution time and energy consumption of a target MPSoC architecture. The model selection has been carried out by considering a target system architecture that is composed of a shared-memory multiprocessor with private L2 caches. In the target architecture, a MESI snoopy-based coherence protocol acts directly among the L2 caches, requiring additional invalidates/writes between L1 and L2 caches to ensure coherence of the data. Cache inclusion is maintained explicitly by using the mechanism adopted for propagating the coherence events in the cache hierarchy [24].

To provide a comprehensive yet affordable validation of the proposed methodology, we identified a design space that is composed of a set of nine independent platform parameters (corresponding to the configuration vector $x$ and presented in Table I). Moreover, we used the issue width as an index of the core complexity by scaling accordingly an additional set of microarchitectural parameters (see Table II for details).

The assumptions about the parameter dependences are derived from other works that appeared in literature [25], [26]. The corresponding design space size consists of $|X| = 2^{17}$ configurations. The minimum and maximum values for each parameter have been set based on the knowledge of the target embedded domain which is characterized by tight area constraints and low-power budget.

The selected target applications have been derived from the SPLASH-2 [4] benchmark suite ($U = \{\text{FFT, OCEAN, LU, RADIX}\}$); for each application, we considered three different input data sets, resulting into 12 application data set scenarios. For the evaluation of the system metrics (mainly in terms of execution time and energy consumption), an SESC [3] simulator has been used. SESC is a fast simulator for CMP architectures with out-of-order processors that can provide energy and performance values for a given application. The evaluation of the energy consumption of the memory hierarchy is supported by CACTI [27], while the energy consumption due to the core logic is based on the Wattch models [28]. The SESC simulator has been chosen because it offers a good tradeoff between simulation speed and modeling accuracy (less than 5% with respect to an actual implementation of the MIPS R10K architecture [3]) and it is widely adopted by the scientific community for multiprocessor design space modeling and exploration [26], [29]–[32].

In the following sections, we introduce the four RSM models used in the proposed methodology: linear regression, Shepard-based interpolation, ANNs, and RBFs. Each RSM model has been used to predict both the execution latency and the energy for each target application and given data set. For each RSM model, the model selection phase has been described as well.

1) Linear Regression: Linear regression is a technique for building and tuning an analytic model $r(x)$ as a linear combination of $x$’s parameters in order to minimize the prediction residual $\epsilon$.

We apply regression by taking into account also the interaction between the parameters and the quadratic behavior with respect to a single parameter. We thus consider the following general model:

$$r(x) = \alpha_0 + \sum_{j=1}^{n} \alpha_j x_j^2 + \sum_{l=1}^{n} \sum_{j=1}^{n} \beta_{l,j} x_l x_j + \sum_{j=1}^{n} \gamma_j x_j$$

(19)

where $x_j$ is the level associated with the $j$th parameter of the system configuration.

A least square analysis can be used to determine a suitable approximation of the parameters. The least square technique determines the values of the unknown quantities in a statistical model by minimizing the sum of the squared residuals (the difference between the approximated and observed values).

A measure of the quality of fit associated with the resulting model is called the coefficient of determination and defined as follows:

$$R^2 = \frac{\sum_{k}(y_k - \bar{y})^2}{\sum_{k}(r_k - \bar{y})^2}$$

(20)

where $y_k$ is the $k$th observation, $\bar{y}$ is the average of the observations, and $r_k$ is the prediction for the $y_k$ observation. $R^2$ corresponds to the ratio of variability in a data set that is accounted for by the statistical model. Usually, the higher is $R^2$, the better is the quality of fit (with $0 < R^2 < 1$).

Although adding parameters to the model can improve $R^2$, there is a risk of exceeding the actual information content of the data, leading to arbitrariness in the final (fit) model parameters (also called overfitting). This reduces the capability of the model to generalize beyond the fitting data while giving very good results on the training data.

To this purpose, we introduce an “adjusted” definition of $R^2$.

This term adjusts for the number of explanatory terms in a
model; it increases only if the terms of the model improve it more than expected by chance and will always be less than or equal to $R^2$; it is defined as

$$1 - (1 - R^2) \frac{N - 1}{N - p}$$

where $p$ is the total number of terms in the linear model (i.e., the set of coefficients $\alpha, \beta, \gamma$), while $N$ is the sample set size. The adjusted $R^2$ is particularly useful in the model selection stage of model building.

**Linear regression model selection:** In order to understand the optimal number of terms of the linear model and the corresponding model order, we analyzed the behavior of the RSM cross-validation error and adjusted $R^2$ by varying the number of random training samples (derived from the simulations of the target applications) from 200 to 1800. The cross-validation consists of predicting both energy consumption and execution cycles for a set of 15 000 random system configurations not used during the training of the model.

Equation (21) represents an improved measure of the overall quality of fit of linear regression: It is inversely proportional to the model’s degrees of freedom (i.e., $N - p$) which, in turn, depend on the order of the chosen polynomial $r(x)$.

As a matter of fact, higher degrees of freedom increase the chance of reduced variance of the model coefficients, thus improving model stability while avoiding overfitting [21], [33]. Our heuristic model selection tries to maximize the number of degrees of freedom and, at the same time, to minimize (on the order of 200) the number of simulations needed to build the model in the initial iterations of the methodology. Thus, as a “rule of thumb,” we decided to set the maximum number of terms to 50 to increase the chance of good quality of fit. Eventually, we limited the set of considered models to the following configurations:

1) first-order model, without any interaction between parameters (nine terms);
2) first-order model, with interaction between parameters ($9 + \binom{2}{1} = 45$ terms included parameter cross-products);
3) second-order model, without any interaction between parameters (18 terms).

Fig. 6(a) shows the average adjusted $R^2$ for the considered models (computed across the 12 introduced scenarios) by varying the number of simulations. The first-order model (with interaction between parameters) is the best among all the possible configurations in terms of ability to explain the linear dependence among all the parameters. The second-order model (without interaction between parameters) is the second best, representing a good tradeoff between the adjusted $R^2$ and the number of parameters. Fig. 6(b) shows that the first-order model (with interaction) and the second-order model present an average prediction error between 35% and 45% over all the target prediction space and scenarios, while the first-order model (without interaction) is between 55% and 60%.

The previous experimental considerations suggest to choose the first-order model (with interaction) as the candidate model. To further improve the prediction accuracy, we introduce a Box–Cox power transform on each sample $y_k$ of the observed data [8]. The Box–Cox transform is a useful data (pre)processing technique used to reduce data variation, make the data more normal distributionlike, and improve the correlation between variables. The power transformation is defined as a continuously varying function with respect to the power parameter $\lambda$

$$y_k^{(\lambda)} = \begin{cases} (y_k^\lambda - 1)/\lambda, & \text{if } \lambda \neq 0 \\ \log y_k, & \text{if } \lambda = 0. \end{cases}$$

We considered a family of transformations as potential candidates $\lambda \in \{1, 0.5, 0, -1\}$. For each transformation, we selected a set of random configurations as input to the linear regression model and computed the average normalized error, a varying-size training set. Fig. 6(c) shows that, among all the possible Box–Cox transformations, the “log” transformation is the most stable and provides the best approximation error for the considered model configurations. On the other hand, the $\lambda = -1$ transformation decreases the stability of the prediction model despite of decreasing the average error. We remark that an inverse Box–Cox transformation has been applied on the model predicted data to keep it consistent with the target problem objective functions.

2) Shepard-Based Interpolation: The Shepard’s technique is a well-known method for multivariate interpolation. This technique is also called the inverse distance weighting method because the values of the response function in the unknown points are the sum of the value of the response function in the known points weighted with the inverse of the distance. In particular, the value of a response function $r(x)$ for an unknown
Box–Cox power transform can also be applied to the configurations used as a training set (known points). In the figure, the Box–Cox power transform can also be applied to the training set.

Shepard-based interpolation model selection: Fig. 7(a) shows the cross-validation results of Cascade ANNs over the set of 15 000 configurations selected from the target architecture by varying the number of random training configurations from 200 to 1800 (as in the case of linear regression). Although, for every Box–Cox transform, the selected ANNs perform very well, with an error that is under 20%, the log Box–Cox transform is the one that presents the most evident advantages. We will use this transform for the experimental results of our methodology.

4) RBFs: RBFs represent a widely used interpolation/approximation model [36]. The interpolation function is built on a set of training configurations \( x_k \) as follows:

\[
\phi(z) = \begin{cases} 
  z, & \text{linear} \\
  z^2 \log z, & \text{thin plate spline} \\
  (1 + z^2)^{1/2}, & \text{multiquadric} \\
  (1 + z^2)^{-1/2}, & \text{inverse multiquadric} \\
  e^{-z^2}, & \text{Gaussian}
\end{cases}
\]

The weights \( \lambda_k \) are the solutions of a matrix equation which is determined by the training set of configurations \( x_k \) and the associated observations \( y_k \):

\[
\begin{bmatrix}
A_{11} & A_{12} & \ldots & A_{1n} \\
A_{21} & A_{22} & \ldots & A_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
A_{n1} & A_{n2} & \ldots & A_{nn}
\end{bmatrix}
\begin{bmatrix}
\lambda_1 \\
\lambda_2 \\
\vdots \\
\lambda_n
\end{bmatrix}
= 
\begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{bmatrix}
\]

where

\[
A_{jk} = \phi(\|x_j - x_k\|), \quad j, k = 1, 2, \ldots, N.
\]
RBF model selection: Fig. 7(c) shows the cross-validation results of the RBF interpolation model over a set of 15,000 configurations selected from the target architecture by varying the number of random training configurations \( N \) from 200 to 1800 (as in the case of linear regression). Although we tested every combination of Box–Cox transform and \( \phi \) function, Fig. 7(c) shows only the best transform for each \( \phi \). It can be seen that the thin plate spline with the log Box–Cox transform is the most significant in terms of reduction in the estimation error. In the next sections, we will use this combination for the validation of our methodology.

V. VALIDATION OF ReSPIR

To validate the proposed ReSPIR methodology, we applied it to the customization of a symmetric shared-memory multiprocessor architecture for the execution of a set of standard benchmarks.

A. Experimental Setup

The target architecture is the one already presented in Section IV-E. Furthermore, in this case, the analysis focuses on the architectural parameters listed in Table I, representing a design space that is composed of \(|X| = 2^{17}\) configurations to be simulated by SESC [3]. The target applications are still the same derived from the SPLASH-2 [4]; for each application, three different input data sets have been considered, thus resulting into 12 application data set scenarios.

The actual system response consists of the average execution time and mWatt/MIPS system response over the set of 12 different scenarios

\[
\min_{\xi \in X} \left[ \frac{\text{avg\_time\_\mu s}(x)}{\text{avg\_mW\_per\_MIPS}(x)} \right] \quad (30)
\]

where

\[
\text{avg\_time\_\mu s}(x) = \prod_{\xi_k \in U} \text{time\_\mu s}(x, \xi_k) \quad (31)
\]

\[
\text{avg\_mW\_per\_MIPS}(x) = \prod_{\xi_k \in U} \text{mW\_per\_MIPS}(x, \xi_k) \quad (32)
\]

subject to the following constraint:

\[
\text{total\_cache\_size}(x) \leq 1 \text{ MB}. \quad (33)
\]

The average execution time and mWatt/MIPS are computed by using the geometric average. This type of aggregation technique, which tends to dampen the effect of very high or low values, is commonly used within the performance evaluation community. As an example, the CPU SPEC 2006 [37] uses geometric mean to aggregate the information on execution time ratios and throughput.

The total cache size of the architecture \( x \) is defined as the sum of the L1 I/D cache size and the private L2 cache for each processor times the number of processors.

B. Comparison of ReSPIR With State-of-the-Art Heuristics

To provide a fair comparison of ReSPIR with respect to the state-of-the-art heuristics, we introduce a multiobjective simulated annealing (MOSA) derived from [38] and a nondominated sorting genetic algorithm (NSGA-II) derived from [39]. Both heuristics are parameterizable in terms of parameters such as the initial population, the number of iteration steps, the set of permutation probabilities, and other specific parameters.

Because both NSGA-II and MOSA are inherently random, each combination of heuristic parameters should be evaluated more than once to infer a more general trend. A similar consideration also applies for some parameters of our proposed strategy (ReSPIR); in particular, this happens whenever we use a random DoE or a neural network (e.g., the learning set is chosen at random from the training set). This issue motivates us in comparing ReSPIR with MOSA and NSGA-II by using multiple sample runs of each algorithm. The number of runs for each algorithm is set such that the actual performance of the algorithm (in terms of ADRS) reaches an average asymptotic value. This generated more than 100 evaluations for each heuristic.

Each strategy has been run by considering an upper bound on the number of simulations which is 3.5% of the whole design space since we are focused on obtaining a good approximation of the exact Pareto set (\( \text{ADRS} \leq 1\% \)) by executing less than three times the number of simulations of the model selection presented in the previous section.

Thus, we pruned accordingly the parameter space of each heuristic to obtain those combinations, leading to a suitable number of simulations. The resulting Pareto fronts have been validated against the reference exact Pareto front of the target architecture.\(^3\)

Concerning the ReSPIR methodology, we first focused on the evaluation of the overall performance by presenting a collapsed view for all the combinations of DoE and RSMs; then, we detailed the algorithm performance for each DoE and RSM.

1) Statistical and Probabilistic Analysis of the Heuristics: Fig. 8(a)–(c) shows the average and standard deviation values of the ADRS for the approximated Pareto fronts found by using MOSA, NSGA-II, and ReSPIR with respect to the exact Pareto by varying the percentage of the design space analyzed from 1% to 3.5%.

Comparing the three figures, the MOSA algorithm presents the worst values of the ADRS. As a matter of fact, considering 1%, 2.5%, and 3.5% of the design space configurations, the MOSA algorithm presents the average values in terms of ADRS that is equal to 18%, 10%, and 7%, respectively. Those values

\(^3\)The reference Pareto front has been computed with a full-search algorithm; thus, it is the exact Pareto front.
are significantly higher than the corresponding values presented
by NSGA-II (6%, 4%, and 3%, respectively) and ReSPIR (4%,
2%, and 0.5%, respectively). Furthermore, from the point of
view of the standard deviation, the MOSA algorithm is worst
than NSGA-II and ReSPIR. The MOSA reaches a value around
4% when the percentage of the design space analyzed is equal
to 3.5%; NSGA-II and ReSPIR reach the values of 0.5% and
1% as lower bound in the analyzed range of the design space,
respectively.

The results of this initial comparison suggested us to focus
on the comparison of the probabilistic performance of the three
heuristics. For each algorithm, we evaluated the probability to
obtain an ADRS under a given threshold (6%, 3%, and 1%)
by varying the percentage of the design space analyzed. The
probability has been estimated by using multiple runs of each
heuristic as described in the previous paragraph. Fig. 9(a)–(c)
shows the estimated cumulative probability to obtain an ADRS
under the given thresholds for MOSA, NSGA-II, and ReSPIR,
respectively.

Concerning the 6% threshold, a steepest climb can be noted
for the ReSPIR algorithm, which overcomes 60% for 2.5% of
the design space, while NSGA-II in under the 40% for the
same design space size. A 10% gap is still present at 3.5%
of the design space, where the ReSPIR reaches almost 90% of
the probability. As expected from Fig. 8(a), the probabilistic
performance of the MOSA algorithm confirms the worst values.

Considering the lower threshold values, the performance of
both MOSA and NSGA-II becomes worse than ReSPIR. In
particular, while ReSPIR reaches an ADRS that is less than 1%
with a 40% probability, both MOSA and NSGA-II were not
able to reach such lower ADRS values during our experimental
campaign. These results represent a significant achievement
in proving that the ReSPIR algorithm outperforms two well-
known state-of-the-art heuristics when considering a so limited
number of simulations. This is essentially due to the fact that
both MOSA and NSGA-II algorithms require a large number of
simulations to converge to good ADRS values with a significant
probability.

Fig. 10 presents a direct comparison of the approximated
Pareto sets found by the MOSA, NSGA-II, and ReSPIR for 3.5% of
the explored design space.
mWatt/MIPS, the heuristic solutions tend to have similar execution time, increasing the mWatt/MIPS ratio highlights the ability of ReSPIR to find a faster architecture configuration (up to 16%).

2) Analysis of ReSPIR DoEs and RSMs: In this section, we focus the analysis on the 1% ADRS threshold introduced in the previous paragraph. For such a small ADRS value, the ReSPIR algorithm demonstrated an asymptotic probability of 40%. However, this result represents the average behavior of all the combinations of DoE and RSMs presented in this paper.

Fig. 11(a) shows the conditional probability for each DoE used during the experimental campaign. Although the random and the full factorial DoEs show a significant asymptotic probability, the central composite design is the definitive winner for constructing a better RSM for our target application domain. In fact, while, for less than 2% of the analyzed design space, all the DoE techniques perform similarly, for values that are bigger than 2%, the central composite design outperforms all the other DoEs, reaching an asymptotic value of more than 70%.

Fig. 11(b) shows a similar analysis but focusing on each RSM used. For the reported results, the neural networks represent the most promising RSM, with a probability that is close to 100% for the 3.5% of the analyzed design space. Considering the other RSM techniques, the Shepard interpolation and the RBFs show a fair asymptotic probability around 40%, while the linear regression methods are close to 0% for all the analyzed design space.

Given the general-purpose features of the benchmarks used in this analysis, we have confidence that the ReSPIR with a central composite design DoE and the cascade neural network RSM is the best methodology to be applied for the target architecture. This configuration of the ReSPIR methodology has been used in the next section for the MPEG decoder case study.

VI. MPEG DECODER CASE STUDY

In this section, the ReSPIR methodology has been applied to the customization of an MPSoC architecture for the execution of an MPEG2 decoder application. Given the promising results presented in the previous section, we will leverage ReSPIR with a central composite design as DoE and the cascade neural network as RSM. As before, an SESC [3] simulator has been used as the target MPSoC architecture model extended with an area model (for 70-nm process technology) derived from [25] for the IBM Power5 architecture. We decided to prune the initial design space (described in Table I) from those configurations corresponding to an area exceeding 100 mm$^2$. The resulting design space is composed of 23K architectural configurations.

For the MPEG application-specific customization, we formalized the MOO problem as follows:

$$\min_{x \in X} \begin{bmatrix} \text{total_system_area}(x) \\ \text{energy_per_frame}(x) \\ \frac{1}{\text{frame_rate}}(x) \end{bmatrix}$$

subject to the following constraints:

$$\text{total_system_area}(x) \leq 85 \text{ mm}^2$$

$$\text{frame_rate}(x) \geq 25 \text{ ft/s}.$$  

The minimization problem consists of three objective functions which are the total system area, the energy consumption per frame, and the inverse of the frame rate. The area constraint has been defined to represent an overall upper bound to the cost of manufacturing and packaging. The minimum frame rate constraint has been introduced as a quality of service (QoS) constraint considering a standard 50 half-frame per second.

In our case study, we used the ALPBench MPEG2 decoder [40]. The objective functions have been derived as a geometric average of the system metrics over a set of five input data sets composed of ten frames at a resolution of 640 × 480. We set an upper bound on the number of simulations of less than 4% of the whole design space for the ReSPIR algorithm. The optimization phase required approximately nine days by using a server with two Xeon quad-cores at 3 GHz. We would like to remark that 4% corresponds to the percentage of the design space that is suitable for deriving a reasonable training set for this particular case. Concerning other architectures where the design space is larger, the size of the training set is expected to remain constant due to the convergence of the response surface
models; thus, the percentage is likely to decrease. Moreover, since the examined configurations are independent, the ReSPIR algorithm simulations can be parallelized to further reduce the overall simulation time. The average simulation time for each data set was 18 min, and the number of inner iterations of the ReSPIR algorithm was four. This enabled the simulation scheduler to run the simulations on the parallel cores.

The final Pareto front found by applying ReSPIR is composed of 163 configurations, and it is shown in Fig. 12.

To help the system architect to select among the large number of feasible solutions composing the Pareto front, we can envision an approach that starts by clustering the architectural configurations and then selects a champion solution for each cluster by using a decision-making mechanism. In this case, we used a $k$-means clustering algorithm applied to the frame-rate metric to create three groups of architectural configurations, representing low, medium, and high frame-rate solutions. Fig. 12 shows the scatter plot of the clustered configurations with cluster centroids centered around the value of the frame rate that is equal to 26, 33, and 48 [ft/s]. For each cluster, we identified the best configuration by using as a decision-making mechanism the optimization of the product

$$\text{total\_system\_area}(x) \times \text{energy\_per\_frame}(x).$$

The three selected system configurations found are shown in Table III. First of all, we note that the configurations found are very close to the clustering centroids in terms of frame rate, representing low, medium, and high frame rates. We also can note that the frame rate is a monotonic function of the area, while the energy per frame is a relatively constant value.

The results reported in Table III suggest that, given the application-specific problem and the decision-making mechanism expressed in (37), the performance of the application (frame rate) depends on the exposed parallelism (both at task and instruction levels) but not on the memory subsystem. This consideration can foster the investigation toward approaches exploiting the parallelism both at task and instruction levels.

Overall, the proposed methodology has been able to find a set of candidate implementations with a very low number of system simulations (corresponding to $\leq 4\%$ of the design space). Each implementation trades off the area for increased QoS with a relatively constant power consumption while meeting decoding constraints and manufacturing costs.

VII. CONCLUSION

In this paper, we have proposed ReSPIR, a DSE methodology that leverages traditional DoE paradigms and RSM techniques combined with a powerful way of considering customized application constraints. The DoEs phase generates an initial plan of experiments which are used to create a coarse view of the target design space; then, a set of response surface extraction techniques is used to identify the nonfeasible configurations and refine the Pareto configurations. This process is repeated iteratively until a target criterion, e.g., number of simulations, is satisfied. The proposed methodology has been used to customize an MPSoC architecture for the execution of an MPEG application with QoS and manufacturing cost constraints. Overall, the proposed methodology has been able to find a set of candidate implementations with a very low number of simulations by trading off the area for increased QoS at a constant power consumption cost.

REFERENCES


Fig. 12. Final Pareto set clustered in three sets.

<table>
<thead>
<tr>
<th>Parameter/metric</th>
<th>Cluster 0</th>
<th>Cluster 1</th>
<th>Cluster 2</th>
</tr>
</thead>
<tbody>
<tr>
<td># Processors</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Processor issue width.</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>L1 instruction cache size</td>
<td>16K</td>
<td>16K</td>
<td>16K</td>
</tr>
<tr>
<td>L1 data cache size</td>
<td>8K</td>
<td>8K</td>
<td>8K</td>
</tr>
<tr>
<td>L2 private cache</td>
<td>128K</td>
<td>128K</td>
<td>128K</td>
</tr>
<tr>
<td>L1 instruction cache assoc.</td>
<td>1w</td>
<td>1w</td>
<td>1w</td>
</tr>
<tr>
<td>L1 data cache assoc.</td>
<td>8w</td>
<td>8w</td>
<td>8w</td>
</tr>
<tr>
<td>L2 private cache assoc.</td>
<td>2w</td>
<td>2w</td>
<td>2w</td>
</tr>
<tr>
<td>I/D/L2 block size</td>
<td>16B</td>
<td>16B</td>
<td>16B</td>
</tr>
<tr>
<td>total_system_area [mm²]</td>
<td>38.74</td>
<td>57.66</td>
<td>73.33</td>
</tr>
<tr>
<td>energy_per_frame [J/J]</td>
<td>0.58</td>
<td>0.53</td>
<td>0.59</td>
</tr>
<tr>
<td>frame_rate [fps]</td>
<td>25.62</td>
<td>35.82</td>
<td>49.06</td>
</tr>
</tbody>
</table>


Gianluca Palermo (M’06) received the M.S. degree in electronic engineering and the Ph.D. degree in computer engineering from Politecnico di Milano, Milano, Italy, in 2002 and 2006, respectively.

Previously, he was a Consultant Engineer with the Low Power Design Group, Advanced System Technology, STMicroelectronics, where he worked on network-on-chip, and also a Research Assistant with the Advanced Learning and Research Institute, University of Lugano, Lugano, Switzerland. He is currently an Assistant Professor with the Dipartimento di Elettronica e Informazione, Politecnico di Milano. His research interests include design methodologies and architectures for embedded systems, focusing on low-power design, on-chip multiprocessors, and network-on-chip.

Dr. Palermo is a member of HiPEAC2 European Network of Excellence.

Cristina Silvano (S’97–M’99–SM’09) received the M.S. degree in electronic engineering from Politecnico di Milano, Milano, Italy, in 1987 and the Ph.D. degree in computer engineering from the University of Brescia, Brescia, Italy, in 1999.

From 1987 to 1996, she was a Senior Design Engineer with the R&D Labs, Groupe Bull, Pregnana, Italy. From 2000 to 2002, she was an Assistant Professor with the Department of Computer Science, University of Milano, Milan. She is currently an Assistant Professor in computer engineering with the Dipartimento di Elettronica e Informazione, Politecnico di Milano. She has published one scientific international book and more than 70 papers in international journals and conference proceedings, and she is the holder of seven international patents. Her primary research interests are in the area of computer architectures and computer-aided design of digital systems, with particular emphasis on low-power design and embedded systems-on-chip.

Dr. Silvano is a member of HiPEAC2 European Network of Excellence.

Vittorio Zaccaria (M’08) received the Laurea and Ph.D. degrees in computer engineering from Politecnico di Milano, Milano, Italy, in 1998 and 2002, respectively.

From 2003 to 2007, he held various positions with the Advanced System Technology Group, STMicroelectronics, where he was involved in the architectural specification and design of multithreaded processors with particular emphasis on emerging stream-programming paradigms. He has also been involved in the development of tools and methodologies for designing low-power industrial networks-on-chip. He is currently a Research Associate with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, where he is overseeing the research and development of design space exploration tools for embedded systems. He has published a book and over 20 papers in international journals and conference proceedings, and he is the holder of two European and two U.S. patents. His primary research interests are computer-aided design of digital systems, with particular emphasis on low-power design and embedded systems-on-chip.

Dr. Zaccaria is a member of HiPEAC2 European Network of Excellence.