# Scenario-Based Design Space Exploration of MPSoCs

Peter van Stralen and Andy Pimentel
*Computer System Architecture Group*
*Informatics Institute, University of Amsterdam*
{*p.vanstralen,A.D.Pimentel*}*@uva.nl*

*Abstract*— **Early design space exploration (DSE) is a key ingredient in system-level design of MPSoC-based embedded systems. The state of the art in this field typically still explores systems under a single, fixed application workload. In reality, however, the applications are concurrently executing and contending for system resources in such systems. As a result, the intensity and nature of application demands can change dramatically over time. This paper therefore introduces the concept of workload scenarios in the DSE process, capturing dynamic behavior both within and between applications. More specifically, we present and evaluate a novel, scenario-based DSE approach based on a coevolutionary genetic algorithm.**

## I. INTRODUCTION

Modern embedded systems are increasingly based on heterogeneous MultiProcessor SoC (MPSoC) architectures, integrating components that range from fully programmable processors to dedicated hardware blocks for time-critical application tasks. The application behavior of these, typically software-centric, MPSoC-based embedded systems also becomes more and more dynamic. Here, one can distinguish two types of dynamic behavior: intra-application and inter-application dynamic behavior. Intra-application dynamic behavior relates to the different behaviors, or operation modes, of a single application. For example, a video application could dynamically lower its resolution (and thus its QoS) to decrease its computational demands in order to save battery life. Inter-application dynamic behavior, on the other hand, is caused by the fact that modern embedded systems often require to support an increasing number of applications and standards. Also, todays embedded systems become more and more "open systems" for which third-party software applications can be downloaded and installed. As a consequence, the application workload in such systems (i.e., the applications that are concurrently executing and contending for system resources), and therefore the intensity and nature of the application demands, can change very dramatically over time.

To cope with the design complexities of MPSoC-based embedded systems, system-level design has become a promising approach for raising the abstraction level of design, and thereby increasing the design productivity. Early design space exploration (DSE) is an important ingredient of such system-level design, which has received significant research attention in recent years. However, the majority of these DSE efforts still evaluate and explore MPSoC architectures under single-application workloads. In this paper, we therefore build upon the concept of *workload scenarios* [1], capturing dynamic application behavior at both inter and intra application levels, and present a novel, *scenario-based DSE* approach.

An important problem that needs to be solved by such scenario based DSE is the fact that the number of possible workload scenarios usually is too large for an exhaustive evaluation of *all* the design points with *all* workload scenarios during the MPSoC DSE. Therefore, a representative subset of scenarios need to be selected for the evaluation of MPSoC design points. The most attractive way of such scenario selection is to obtain a representative subset of scenarios statically, before the DSE is performed. This could, e.g., be done by clustering the scenarios, based on architecture dependent metrics [1]. However, because our goal is to perform MPSoC DSE, implying that the target MPSoC as well as the mapping of applications onto the underlying MPSoC are still unknown, a static clustering based on architecture dependent metrics cannot be done. For this reason, a dynamic selection method is required. During the DSE itself, a representative subset of scenarios must be dynamically identified based on the MPSoC design points evaluated by the DSE process. The consequence is that the DSE algorithm must solve two problems simultaneously: searching the MPSoC design space as well as searching the scenario space to find a representative subset of scenarios. To this end, we use the concept of coevolution, in which the evolution of a certain type of individual is closely related to the evolution of another type of individual [2], [3]. More specifically, we present and evaluate a *coevolutionary genetic algorithm* to perform scenario-based DSE of MPSoC architectures.

The remainder of this paper is organized as follows. In the next section, we briefly outline the high-level simulation framework Sesame which allows for simulating and evaluating MPSoC design instances. Additionally, a brief description of workload scenarios is given. Section 3 describes our scenario based DSE approach, based on a coevolutionary genetic algorithm. In Section 4, we present experimental results which show that our coevolutionary DSE approach clearly outperforms DSE in which the representative subset of scenarios has been selected statically using random selection. Section 5 discusses related work, after which Section 6 concludes the paper.

## II. SCENARIO BASED MPSOC SIMULATION

To evaluate design points during system-level design space exploration, we deploy the Sesame high-level simulation framework [4]. This framework, as illustrated in Figure 1,
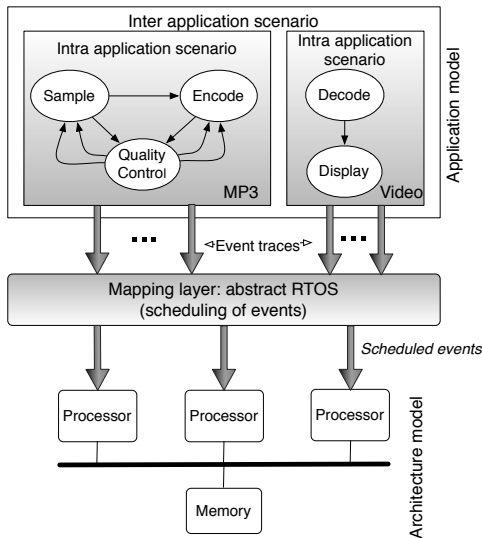
Fig. 1. High level scenario based MPSoC simulation in Sesame in combination with workload scenarios



Fig. 2. The coevolutionary genetic algorithm

allows for rapid performance evaluation of different MPSoC architecture designs, application to architecture mappings, and hardware/software partitioning. Key to this flexibility is the separation of application and architecture models, together with an explicit mapping step to map application models onto an architecture model. This mapping step has been implemented using trace-driven co-simulation of the application and architecture models. The input to Sesame is a set of event traces, which are an abstract representation of the workload imposed on the architecture. After processing the events, the architecture model provides us with the quality (e.g. performance, power) of the mapping.

To systematically specify and generate different multi-application workloads for Sesame, so that these workloads can be used during system-level design space exploration, we use the concept of *workload scenarios* [1]. More specifically, we distinguish two types of application scenarios within a workload scenario: *intra and inter application scenarios*. Intra-application scenarios are scenarios which describe the different behaviors, or operation modes, of a single application. For example, an application may feature various operation modes to maintain a certain level of QoS under changing circumstances. The inter-application scenarios, on the other hand, describe the behavior of multiple applications. More specifically, inter-application scenarios indicate which applications can run concurrently. In Figure 1, for example, it can be seen that an intra-application scenario corresponds to a single application specification, which in our case is a (Kahn) Process Network. We capture an intra-application scenario by the set of trace events that were dispatched by the processes in a Process Network [5] during a particular execution phase of the application. The behavior of a complete system with multiple applications is described by an inter-application scenario. Such an inter-application scenario actually bundles the description of all the individual intra-application scenarios for each of the Process Networks (an MP3 recorder and Video player in
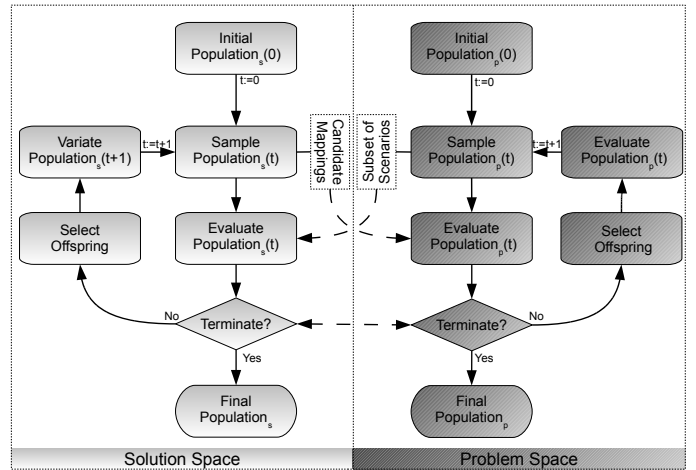
Figure 1). Subsequently, we use a scenario database to compactly store all possible workload scenarios, consisting of both intra-application and inter-application scenarios [6]. This scenario database allows for on-demand generation of application workloads, in the form of event traces, according to the stored scenarios for the purpose of scenario-based DSE using the Sesame framework.

## III. COEVOLUTIONARY ALGORITHM

Our DSE problem consists of finding optimal mappings (also called design points) of the applications onto the architecture, given a subset of application scenarios that is representative for all the scenarios in the scenario database. Here, a *mapping* defines both the allocation of resources in the MPSoC platform and the binding of the tasks and communication channels onto the allocated resources. The difficulty is the dependence of the representativeness of a subset of scenarios on the application mapping. Based on the architecture instances that are explored, the behavior of an application scenario can vary. An obvious example is an architecture that may be optimized for a certain scenario. On the optimized architecture it may execute quite efficiently, whereas in general the scenario is quite expensive. As the architectures used in the DSE are not known beforehand, the representative subset is also not known in beforehand. Consequently, the problem is ill-defined.

A technique to solve ill-defined problems is a coevolutionary genetic algorithm (GA). The coevolutionary GA algorithm can be built in two ways [3]: a combined and a separate genotype approach. This paper presents a coevolutionary DSE method using the separate genotype approach. The implementation of this coevolutionary genetic algorithm is based on the widely-used SPEA2 GA [7].

The separate genotype solution corresponds to two GAs which are running in parallel: a GA in the solution space and a GA in the problem space (shown in Figure 2). In our case, the solution space consists of the different mappings of the applications onto the underlying platform architecture specifying the MPSoC instance. The goal of the solution
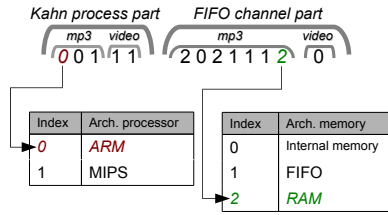
Fig. 3. Chromosome representation of the mapping

space is to optimize the application mapping. As a result, the solution space will try to identify the best resource allocation and binding. The problem space, on the other hand, completes the ill-defined problem by filling in some remaining details of the given problem. It tries to find the best representative subset of the scenarios for the mappings that are evaluated in the solution space. The MPSoC designer is supposed to limit the size of the subsets. As a consequence, the problem space is spanned by all possible valid subsets of application scenarios. In this case, a valid subset has a size which is smaller or equal to the defined limit.

The most important aspect of a coevolutionary genetic algorithm is the interaction between the two populations. This interaction can be symbiotic or competitive [2]. In a symbiotic interaction two species cooperate to obtain a high fitness. For competitive interaction the behavior is the other way around. Between the two populations there is a competition; what is good for one population is probably bad for the other. For our DSE problem, the interaction between the solution and problem space is symbiotic. The better the subsets in the problem space become, the better the fitness of the mappings in the solution space can be predicted. As a consequence, we obtain better mappings since the right mappings will be selected for future generations of the GA based search process.

### A. Chromosome representations

As mentioned, our coevolutionary approach features two separate chromosome representations. For the solution geno-type, there are two types of genes as illustrated in Figure 3. The first part of the chromosome contains the mapping of the processes in all the applications. In the second part, the mappings of the communication channels of all the applications are described. Implicitly, this also contains the resource allocation for the platform. Resources that are not used in any binding (processing or memory) are also not allocated on the platform.

The chromosome representation of the problem genotype represents a complete representative subset. Each individual gene encodes a single scenario of the representative subset. The scenario is described using an integer number referring to the scenario number in the scenario database. This means that it is possible that a single scenario occurs multiple times in the same representative subset. In our point of view, this does not cause any problems because the fitness function will filter the weak subsets out of the population. If a representative subset with multiple occurrences of a single scenario is not filtered out, then it is probably a quite good

|  | A | B | C |
|---|---|---|---|
| $s_1$ | 1.5 | 3.3 | 3.5 |
| $s_2$ | 1.75 | 3.6 | 3.6 |
| $s_3$ | 1.85 | 3.5 | 4 |
| $s_4$ | 2.5 | 6.4 | 5.3 |
| $F_{sol}$ | 1.9 | 4.2 | 4.1 |
| $R_{sol}$ | **1** | **3** | **2** |
| $F_{sol}^s$ | 1.68 | 3.40 | 3.75 |
| $R_{sol}^s$ | **1** | **2** | **3** |
| $d$ | 0.23 | 0.80 | 0.35 |
| $d_N$ | 0.23 | 0.26 | 0.19 |

$$D([s_1, s_3]) = \tfrac{1}{3}(0.23 + 0.26 + 0.19) \approx 0.23$$
$$R([s_1, s_3]) = 1 - \rho = 0.5$$

distribution of scenarios.

### B. Fitness functions

An exhaustive evaluation, using all the possible scenarios, of the individuals in the solution space is infeasible. Consequently, we need to estimate the fitness of the application mappings. The work of Jin and Branke [8] describes several ways of estimating the fitness of a solution: problem approximation, data-driven functional approximation and fitness inheritance. In our work, we use problem approximation. *Problem approximation* tries to replace the original statement of the problem by one that is approximately the same as the original problem, but which is easier to solve. In our case, this means that we are using a representative subset of scenarios instead of all the application scenarios.

Evaluating the estimated fitness of solution individuals is performed using the Sesame framework. The mapping specified in each solution individual is simulated and evaluated using the representative subset of scenarios $s$ received from the problem population (see Figure 2).

Next to this, we need to determine the fitness of the representative subsets. In order to keep the fitness evaluation of the problem space affordable, no additional mappings need to be simulated by Sesame during coevolution. Instead, we are using a *trainer*. The trainer consists of a small number of training mappings that have already been exhaustively (i.e. using all the scenarios) evaluated. As a consequence, the exact fitness of these training mappings is known and can be used to obtain the quality of a subset in the problem population. The quality (or problem fitness) $F_p(s)$ of subset $s$ is calculated for each individual objective, like performance and power, and given in the following equations:

$$D(s) = \frac{1}{|T|} \sum_{i=0}^{|T|} \mathrm{norm}\left(\mathrm{abs}\left(F_{sol}(T[i]) - F_{sol}^s(T[i])\right)\right) \quad (1)$$

$$R(s) = 1 - \rho(T, F_{sol}, f_{sol}^s) \quad (2)$$

$$F_p(s) = \alpha * D(s) + (1 - \alpha) * R(s) \quad (3)$$

The problem fitness $F_p$ is composed of two metrics. We have chosen for two metrics, because in the case of a small trainer these metrics are not capable of selecting the best subset of scenarios. The two metrics are not inherently conflicting. If a subset has the optimal value on one of the metrics, it also has

the optimal value on the other metric. The weighting factor $\alpha$ allows us to give a preference on one of the two metrics. Currently, we have only used $\alpha = 0.5$. The individual metrics, obtained using the trainer $T$, are as follows:

1) The first metric is the deviation of the predictor. The deviation $D(s)$ of subset $s$ is obtained by taking the absolute difference between the real fitness $F_{sol}$ and the estimated fitness $F_{sol}^s$ for each of the training mappings. The difference for each training mapping is normalized such that each training mapping gets a similar weight during the evaluation of the subset. This is done by dividing it by the highest possible difference in the estimated fitness of the training mapping, which is equal to the difference of the largest scenario fitness and the lowest scenario fitness.

2) The second metric is Spearman's rank correlation $\rho$ [9]. Spearman's rank correlation measures the correlation between two variables based on their ranking. For the quality of the subset, the ranking based on the real and estimated fitness is compared. When the ranking of a subset is the same as the exact ranking of the trainer mappings, it is a good subset. So, the goal of the fitness prediction is to obtain the correct trend. It is not about the absolute predicted fitness, but about the relative fitness value. The predictor needs to make sure that the Pareto optimal mapping solutions are found. This can only be the case when the Pareto optimal solutions have the highest predicted fitness. For the problem fitness function, the rank correlation is subtracted from 1.0 such that, analogous to the deviation, this metric must be minimized.

In Table I, an example fitness evaluation is given with a single objective (a lower fitness value is better). In this case, the trainer consists of three mappings: $A$, $B$ and $C$. The application has four different kinds of behavior, which are stored at the scenario database in the four scenarios: $s_1$, $s_2$, $s_3$ and $s_4$. The function $F_{sol}$ gives the exhaustive evaluation (the average of scenario $s_1$, $s_2$, $s_3$ and $s_4$) for each of the training mappings. Based on the exhaustive evaluation, we can conclude that mapping $A$ gets rank 1, mapping $C$ gets rank 2 and mapping $B$ gets rank 3. In this example, the fitness of the subset $s = [s_1, s_3]$ is determined. For mapping $C$, the predicted fitness becomes $(3.5 + 4)/2 = 3.75$. The deviation to the real fitness is equal to 0.35. In order to normalize the value, it is divided by $5.3 - 3.5$, which gives a deviation of approximately 0.19. When the results of all the training mappings are combined, the average deviation is 0.23. The ordering quality is equal to 0.5 due to the differences in the ranks of training mappings $B$ and $C$.

### C. Trainer selection

The next challenge is to obtain a good trainer. During the selection of the trainer, there are two conflicting requirements. The first requirement is that the trainer is as compact as possible. The fewer application mappings in the trainer, the fewer exhaustive evaluations for training mappings are required. On the other hand, the set of training mappings
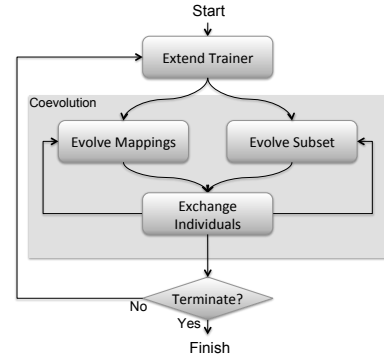


Fig. 4.   Trainer selection during the coevolution

must reflect the solution design space as good as possible. The more training mappings in the trainer, the higher the probability that the trainer provides a good reflection of the solution design space.

Based on these observations, we have chosen to update the trainer dynamically during the coevolution. Because during the coevolution different parts of the solution design space are traversed, it is more likely that the dynamically updated trainer reflects the relevant part of the solution design space. In order to limit the total evaluation time, the trainer is only updated periodically. After a fixed number of generations, only a small number of mappings is added to the trainer.

The procedure is illustrated in Figure 4. The main part of the procedure is the coevolution process. Periodically, the coevolution process is paused to extend the list of mappings in the trainer. The length of the period is given as a parameter of the scenario-based design space exploration.

The extension procedure of a trainer starts by getting a list of candidate mappings. As can be seen in Figure 2, in each generation the solution space will supply a list of candidate mappings. If the trainer is not updated, this list is simply discarded. Otherwise, the candidate mappings are analyzed to identify the most interesting mappings. To identify the most interesting mappings, the candidate mappings are partially evaluated by Sesame. In this partial evaluation, the scenarios used in the subsets of the problem population (without the offspring) are used. For each of the candidate mappings, the standard deviation of the estimated fitness of all the subsets in the problem population is calculated. This standard deviation is normalized. The mappings with the highest normalized standard deviation are chosen. The rationale behind this idea is that the larger the standard deviation, the higher the uncertainty about the predicted fitness. As our fitness predictor must reduce the uncertainty of the predicted fitness, it makes sense to exhaustively evaluate the uncertain mappings and use them for training.

Finally, scenario-based DSE allows to throttle the execution time. As the Sesame simulations are the most significant part of the execution time, the designer can specify the maximum number of times Sesame is invoked (both in the solution and problem space). The coevolutionary procedure guarantees that this amount of evaluations is not exceeded. This is done by adjusting the number of mappings added

| Population size | 200 | Population size | 150 |
|---|---|---|---|
| Offspring size | 30 | Offspring size | 25 |
| $P_{crossover}$ | 0.8 | $P_{crossover}$ | 0.8 |
| $P_{mutate}$ | 0.01 | $P_{mutate}$ | 0.02 |
| Generations | 600 | Gen-train | 10 |
| **Solution Space** | | **Problem Space** | |

to the trainer during the trainer extension. For this purpose, the number of allowed Sesame calls per generations is determined. During the extension of the trainer, the difference between the number of allowed Sesame calls until now and the executed Sesame calls is used for the trainer extension. The higher the maximum amount of Sesame evaluations, the higher the quality of the trainer will be. To a certain extent, this will improve the result of the scenario-based DSE.

## IV. EXPERIMENTAL RESULTS

To evaluate our coevolutionary DSE approach, we have studied the design space exploration of two types of multi-application workloads. The first multi-application workload is composed of two real applications, whereas the second multi-application workload only contains synthetic applications giving us more control about its behavior.

In both experiments, the design space exploration entails the search of the optimal design instances – in terms of cost and execution time objectives – of a heterogeneous MPSoC platform target architecture consisting of at most eight processing elements: three MIPS processors, three ARM processors and dedicated DCT and VLE hardware blocks. For communication, the platform architecture allows for using a crossbar with private memory buffers, four dedicated point-to-point FIFOs and / or a bus connected to a shared memory. As the mapping determines the resource allocation, not all these elements need to be used in the final platform instance.

We compare our coevolutionary approach against a *static approach* to select a representative subset of scenarios. In this static approach, we have randomly selected a sample of the workload scenarios in the scenario database. This fixed, random sample of scenarios has subsequently been used for the evaluation of solution individuals (i.e., mappings) in a traditional DSE process using a single (SPEA2 based) GA for searching the solution space. Additionally, the coevolutionary approach is compared to an exhaustive DSE where the complete scenario database is used for the evaluation of each solution individual. The general settings of the GA for the solution and problem spaces are given in Table II. The table on the right is only applicable for the coevolutionary GA. Here, the Gen-train is the interval after which the trainer is updated. Each experiment is repeated several times to take the randomness of a GA into account. The realistic experiment is repeated twelve times, whereas the synthetic approach is repeated six times.

Since the coevolutionary and static approaches select different representative subsets of workload scenarios, we cannot directly compare their obtained solution populations.

The problem is that the objective values, such as execution time, have been calculated on different scenario subsets. To resolve this, the real objective values for each solution must be calculated using exhaustive simulation. In order to analyze the convergence of the GAs, this has been done at every 15th population.

The experiments result into Pareto fronts (as illustrated in Figure 6D). A *Pareto front* shows the designer the different tradeoffs that can be made with respect to execution time and cost. However, Pareto fronts cannot directly be used for a quantified comparison between the different methods. There are several known metrics that are capable of quantifying the difference between the Pareto fronts [10]. In our case, we are using the hypervolume indicator (or S-metric). The hypervolume indicator considers the volume of the objective space dominated by the total Pareto front. This volume can be used to compare the results of the different approaches and to analyze the convergence of the GA.The larger the hypervolume, the better the Pareto front.

### A. Real multi-application workload

The real multi-application workload contains two applications: A Motion JPEG (M-JPEG) encoder and a MPEG4 Simple Profile decoder. The M-JPEG application contains 11 intra-application scenarios, whereas the MPEG4 application is composed of 10 scenarios. For the inter-application scenarios all possible combinations are chosen: the M-JPEG application can run concurrently with the MPEG4 application and both applications can run on their own. As a result, 131 different workload scenarios are obtained.

In this experiment, the statically selected subset contains 5 scenarios and the maximal size of the dynamically selected subset in our coevolutionary approach) is equal to 4 scenarios. In this way, the additional Sesame calls performed by the trainer are compensated for and both methods are using the same number of simulations. The results of the experiments are shown in Figure 5A. On average, the hypervolume of the Pareto fronts of the coevolution approach is better than the static method. Both methods converge quite quickly. Starting from the 100th generation, the hypervolume increases very slowly. The final result is that the hypervolume of the coevolution is 0.02 larger. For the designer, this means that, on average, the found solutions of the coevolutionary approach are more than 2 percent faster.

Figure 5B shows the quality of the representative subset of scenarios using Spearman's rank correlation. For this purpose, a large set of exhaustively evaluated solution individuals are used. This set is unequal to the trainer. The higher the rank correlation, the better the ranking of the subset matches with the real ranking. In the graph, the lines show the averaged quality of the subsets over all the runs. Extrema over the different runs are shown in two ways. As the subset of the static approach does not change over time, the quality is also constant. Therefore, the extrema are shown as an error bar. The quality of the coevolution approach changes over time, thus the extrema are shown using a shaded region to prevent clutter.
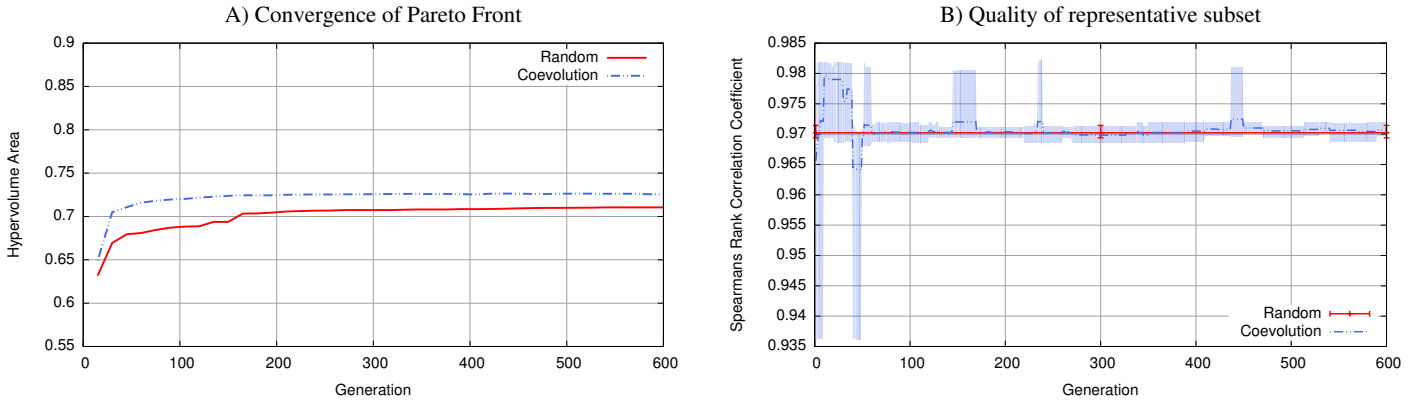
Fig. 5. The results of the coevolutionary GA on the real multi-application workload

The quality of the subsets are more or less equal for both the coevolutionary approach and the random approach. A lack of diversity in the scenario database is the cause of the similar quality. Since the diversity is so small, there is a large probability that a randomly picked subset will be representative for the complete scenario database.

### B. Synthetic multi-application workload

In order to assess a larger diversity of multi-applicational workloads, the coevolutionary GA is also tested on a synthetic workload. The synthetic workload contains 18 applications with a total of 72 process nodes. The total number of intra-application scenarios is 51, which in combination with 27 inter-application scenarios results into a total number of 514 scenarios. In order to make a fair comparison, we choose to use the same amount of evaluations for the random and the coevolutionary genetic approach. This means that the random approach will use a subset of 10 scenarios, whereas the coevolutionary approach only takes 6 scenarios in the representative subset. The remaining Sesame calls in the coevolutionary approach are used for the evaluation of the trainer. As a consequence, the coevolutionary approach needs to work extra hard in order to let the representative subset outweigh the random subset.

The results of the experiment are given in Figure 6A. Evidently, the best method is the exhaustive evaluation of the complete workload. One can see that for this approach the hypervolume of the Pareto front is only increasing during the evolution. The reason is obvious, as it selects individuals based on the real fitness, it will only take the strong individuals for use in the next generation[1]. However, the exhaustive method is only run once and for 420 generations. The evaluation effort of the exhaustive evaluation is impractical, as can be seen in Figure 6B. Whereas the exhaustive approach has a (wall clock) execution time of 163 hours, the coevolution and static approaches only take approximately 6 hours.

The evaluation effort of the random and coevolutionary approaches are more or less equal. In this view, it can be seen that the coevolutionary approach clearly outperforms the random approach. The average hypervolume of the coevolutionary approach is almost monotonically increasing. From this observation, we can conclude that the fitness prediction

of the representative subset is good enough to keep the strong individuals in the population. However, the subset only spans 1.2 percent of the scenario database. This is the reason why it is not always capable of selecting all individuals that improve the Pareto front, and as a result, the hypervolume of the exhaustive evaluation is still somewhat larger. On the other hand, the random approach makes wrong decisions resulting into a decreasing hypervolume.

The coevolutionary approach does not only obtain better results, but the variance in the results over different runs is also decreased. In Figure 6A, the extrema of the different experiments are highlighted using a shaded area. These areas partially overlap, as can be seen by the darker region. Since the exhaustive evaluation is only done once, the exhaustive evaluation is not extended with a colored area. In the initial part of the evolution, the fitness differences between the individuals in the solution population are still large enough to make an easy distinction. As a result, all methods have a similar hypervolume and the variance is not too large. During the coevolution, the dynamic improvement of the representative subset takes care that the variance is slowly decreased. On the other hand, the variance of the hypervolume of the random approach is much larger. As this random approach has no clever way of selecting the scenarios, the representativeness of the subsets is extremely uncertain. The uncertainty of the representativeness of the subset gives problems in the later stages in the evolution. Due to the imprecise prediction of the fitness, the fitness predictor based on the subset is not capable of selecting the Pareto optimal solution individuals.

Next, the dynamic improvement of the subset used in the fitness prediction of the solution individuals is demonstrated. The quality of the subsets is obtained in a similar fashion as we described in the previous subsection and can be seen in Figure 6C. A first observation is that, in contrary to the previous experiment where the scenario database had much less diversity, the quality of the subset generated by the coevolutionary approach is much better. As the diversity in the scenario database makes it harder to statically select a representative subset, the coevolution approach is now capable of dominating the static approach. Additionally, the variance of the subset quality in the random approach is both for the deviation and the ranking much larger than the variance of the coevolutionary approach. In fact, the random approach has a variance which is no less than

---

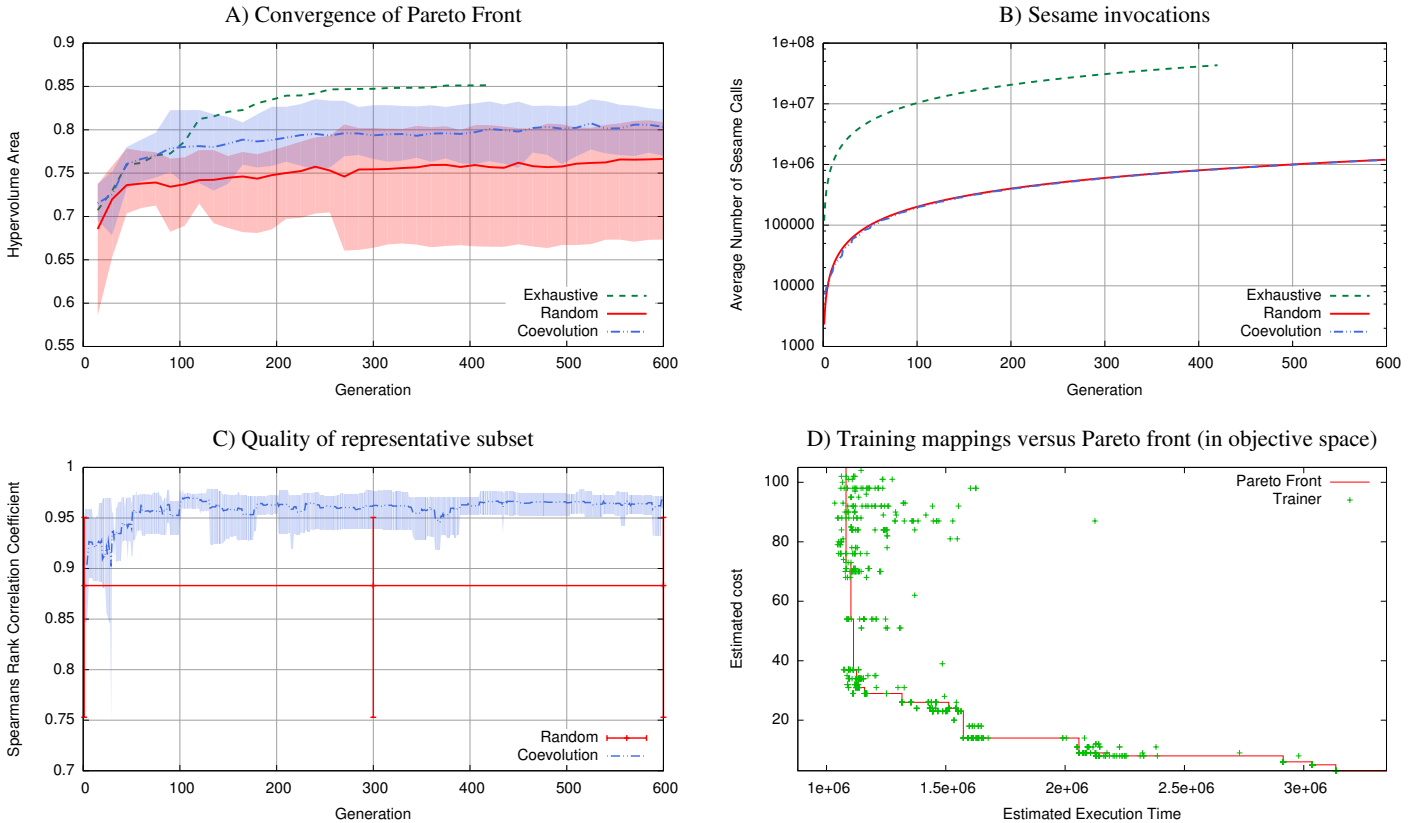[1]SPEA2 is an selection algorithm which makes use of elitism.

Fig. 6. The results of the coevolutionary GA on the synthetic multi-application workload

ten times as large as the coevolutionary approach. Another observation is that in the beginning of the DSE, the search of a representative subset in the problem space is not partially converged yet. As a result, the quality of the dynamically selected subset is within the range of the statically selected subset. After a certain number of generations, however, the dynamically updated subset of 6 scenarios reigns supreme over the statically selected subset of 10 scenarios. The consistent quality of the coevolutionary selected subset outperforms the uncertain quality of the random subset of scenarios. Even the worst subset in the coevolution approach is eventually better than the best subset in the random approach.

Finally, Figure 6D shows one of the trainers. The line is the Pareto front of non-dominated design points which are found in the specific DSE. For the designers, this is the main result of the DSE where they can see the found trade off between performance and cost (in terms of area). Within this graph, all the trainer mappings are plot using their real fitness. It can be observed that the trainer mostly picks the mappings with fitness values close to Pareto optimal design points. In these areas the mappings are cluttered and therefore hard to differentiate.

## V. RELATED WORK

Much research has been performed on high-level modeling and simulation for MPSoC performance evaluation as well as on GA based DSE [11], [12]. However, the majority of the work in this area still evaluates and explores systems under a single (fixed) application workload. Only very recently, research has been initiated on recognizing workload scenarios [13], [1] and making DSE scenario aware [14], [15].

In [1], for example, different single-application scenarios are used for DSE. Another type of scenario is the use-case. A use-case can be compared with what we call inter-application scenarios, and consequently, a use-case describes which applications are able to run concurrently. Examples of frameworks utilizing use-cases for mapping multiple applications are MAMPS [16] and the work of Benini et al. [17]. MAMPS is a system level synthesis tool for mapping multiple applications on a FPGA, whereas Benini et al. use logic programming to reconfigure an embedded system when a new use-case is detected. Another way of describing the use of multiple applications is a multimode multimedia terminal [18], in which the inter-application behavior is captured in a single, heterogeneous model of computation (MoC) combining dataflow MoCs and state machine MoCs. These approaches describe inter-application dynamic behavior, but do not capture intra-application dynamism. To the best of our knowledge, our approach is the first to address both intra and inter application dynamism during DSE of MPSoCs with multi-application workloads.

Moreover, to the best of our knowledge, coevolution is also not used yet in the field of DSE of MPSoCs. However, in other research areas there has been done some work in applying coevolution to reduce the number of scenarios or test cases during the execution of a GA to improve the efficiency. Branke and Rosenbusch [19] apply coevolution for worst-case optimization. They try to optimize a mathematical function $F(s, t)$. In this function $s$ is the solution parameter, whereas $t$ is a test case. The goal is to find the optimal $s$ such that the worst case value of all the possible tests $t$ is as high as possible. Since the worst case performance is an individual

metric (as long as the worst case test is in the population, the real fitness is obtained, irrespective of the other tests in the population), they do not need an exhaustively evaluated trainer.

The work of Schmidt and Lipson [20], on the other hand, uses an exhaustively evaluated trainer. Similarly to our work, they want to increase the efficiency of a GA by using a fitness predictor. The fitness predictors are coevolving with the traditional GA and are composed of a subset of all the samples in the problem definition. In order to evaluate the fitness predictor, a set of exhaustively evaluated training individuals is used. The lower the absolute prediction error on the trainer, the better the fitness predictor is.

In contrast to our work, the fitness predictor is evaluated differently. We are not only normalizing the prediction error to get a fair weight for each individual training element, but also the relative ordering quality is taken into account using Spearman's rank correlation. The relative ordering that a fitness predictor gives is in our point of view the most important quality of a fitness predictor. At the end of the DSE, the best solutions need to be found, and from this perspective it does not matter if the predicted fitness has a systematic error. The combination of Spearman's rank correlation and the prediction error gives a complete view of the quality of a fitness predictor. As a small trainer is not capable of getting a complete view of the ordering requirements, comparable ordering qualities can be discriminated using the prediction error. The smaller the prediction error, the higher the probability that the relative ordering of the fitness predictor is representative for the rest of the design space.

To summarize, our work is the first to exploit the co-evolution technique in order to make the DSE of MPSoCs more efficient. In this way, we are able of exploring a multi-application workload taking both the intra- and inter-application dynamism into account.

## VI. CONCLUSION

To address the increasingly dynamic behavior of application workloads in modern MPSoC-based embedded systems, this paper has introduced the concept of workload scenarios in system-level DSE of such systems. More specifically, we have proposed a novel scenario-based DSE approach, based on a co-evolutionary algorithm. This algorithm simultaneously searches for optimal MPSoC design instances and for representative subsets to evaluate these design instances.

We have shown that, as long as the diversity in the scenario database is large enough, the coevolutionary DSE clearly outperforms the DSE in which the representative subset of scenarios has been selected statically using random selection. The improvement comes in two ways. The first improvement is that the hypervolume of the resulting Pareto fronts is larger. This means that the fully dominated area is larger and we have obtained design points which are faster and cheaper. Secondly, there is a higher probability that the coevolution gives a good outcome. The diversity in the hypervolume of the Pareto fronts and the quality of the representative subset is much smaller than in the random approach. As

a consequence, the results are more consistent than it would be with a randomly selected subset.

Thus, the coevolution approach supports the designer by obtaining a Pareto front of near-optimal design points much faster than the traditional approach that exhaustively evaluates all the scenarios. Additionally, the results are more reliable as the consistency over different runs is much larger than with a randomly selected subset.

## REFERENCES

[1] S. V. Gheorghita *et al.*, "System-scenario-based design of dynamic embedded systems," *ACM Trans. on Design Automation of Electronic Systems*, vol. 14, no. 1, pp. 1–45, 2009.

[2] J. Paredis, "Coevolutionary computation," *Artificial Life*, vol. 2, no. 4, pp. 355–375, 1995.

[3] M. Maher and J. Poon, "Modeling design exploration as co-evolution," *Microcomputers in Civil Engineering*, 1996.

[4] A. D. Pimentel, C. Erbas, and S. Polstra, "A systematic approach to exploring embedded system architectures at multiple abstraction levels," *IEEE Trans. on Computers*, vol. 55, no. 2, pp. 99–112, 2006.

[5] G. Kahn, "The semantics of simple language for parallel programming," in *IFIP Congress*, 1974, pp. 471–475.

[6] P. van Stralen and A. D. Pimentel, "A trace-based scenario database for high-level simulation of multimedia mp-socs," in *Proc. of the Int. Conference on Embedded Computer Systems: Architectures, MOdeling and Simulation (SAMOS '10)*, Samos, Greece, July 2010.

[7] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation, and Control*. CIMNE, Barcelona, Spain, 2002, pp. 95–100.

[8] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Trans. on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.

[9] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, January 1904.

[10] E. Zitzler *et al.*, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[11] M. Gries, "Methods for evaluating and covering the design space during early design development," *Integration, the VLSI Journal*, vol. 38, no. 2, pp. 131–183, 2004.

[12] C. Erbas, S. Cerav-Erbas, and A. D. Pimentel, "Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design," *IEEE Trans. on Evolutionary Computation*, vol. 10, no. 3, pp. 358–374, June 2006.

[13] J. Paul, D. Thomas, and A. Bobrek, "Scenario-oriented design for single-chip heterogeneous multiprocessors," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 8, pp. 868–880, August 2006.

[14] G. Palermo, C. Silvano, and V. Zaccaria, "Robust optimization of soc architectures: A multi-scenario approach," in *Proceedings of ESTIMedia 2008 - IEEE Workshop on Embedded Systems for Real-Time Multimedia. Atlanta, Georgia, USA*, October 2008.

[15] "Eu fp7 multicube project," http://www.multicube.eu/.

[16] A. Kumar, S. Fernando, Y. Ha, B. Mesman, and H. Corporaal, "Multiprocessor systems synthesis for multiple use-cases of multiple applications on fpga," *ACM Transactions on Design Automation of Electronic Systems*, vol. 13, no. 3, pp. 1–27, July 2008.

[17] L. Benini, D. Bertozzi, and M. Milano, "Resource management policy handling multiple use-cases in mpsoc platforms using constraint programming," in *Logic Programming*, ser. Lecture Notes in Computer Science, vol. 5366, December 2008, pp. 470–484.

[18] S. Ha, C. Lee, Y. Yi, S. Kwon, and Y.-P. Joo, "Hardware-software codesign of multimedia embedded systems: the peace approach," in *Proc. of the 12th IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications*, August 2006, pp. 207–214.

[19] J. Branke and J. Rosenbusch, *New Approaches to Coevolutionary Worst-Case Optimization*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2008, vol. 5199.

[20] M. Schmidt and H. Lipson, "Coevolution of fitness predictors," *IEEE Trans. on Evolutionary Computation*, vol. 12, no. 6, pp. 736–749, 2008.