Robust and Flexible Mapping for Real-time Distributed Applications during the Early Design Phases

Junhe Gan[1], Paul Pop[1], Flavius Gruian[2], and Jan Madsen[1]

[1]Department of Informatics and Mathematical Modelling Technical University of Denmark, Denmark
[2]Department of Computer Science, Lund University, Sweden

Abstract–We are interested in mapping hard real-time applications on distributed heterogeneous architectures. An application is modeled as a set of tasks, and we consider a fixed-priority preemptive scheduling policy. We target the early design phases, when decisions have a high impact on the subsequent implementation choices. However, due to a lack of information, the early design phases are characterized by uncertainties, e.g., in the *worst-case execution time*s (*wcet*s), or in the functionality requirements. We model uncertainties in the *wcet*s using the "percentile method". The uncertainties in the functionality requirements are captured using "future scenarios", which are task sets that model functionality likely to be added in the future. In this context, we derive a mapping of tasks in the application, such that the resulted implementation is both robust and flexible. Robust means that the application has a high chance of being schedulable, considering the *wcet* uncertainties, whereas a flexible mapping has a high chance to successfully accommodate the future scenarios. We propose a Genetic Algorithm-based approach to solve this optimization problem. Extensive experiments show the importance of taking into account the uncertainties during the early design phases.

I. Introduction

There is a lot of research on embedded systems design [1], but very few researchers have addressed the early design phases. The decisions taken during the early design phases, e.g., architecture selection and task mapping, have a high impact on the subsequent implementation choices [2]. However, early phases are characterized by many uncertainties, e.g., in terms of the hardware components available, functionality that has to be implemented and attributes such as the *wcet*.

We address hard real-time applications, modeled as a set of tasks, where timing constraints are of utmost importance. We are interested in tackling uncertainties in the functionality requirements and the *wcet*s of tasks. We model the uncertainties in the *wcet*s using the "percentile method" [3], which captures the *wcet* of a task by two values, the 50^{th} and the 90^{th} percentile. These numbers are chosen by the designers based on the best available information and their experience.

Today, most systems are engineered in an evolutionary fashion: introducing a new version of an existing product, introducing new features—possibly as part of a planned evolution of a product line, performing a design-iteration, etc. Functionality requirements often change during these iterations. For example, new tasks may be introduced to update the existing functionality or add a new feature, etc. Several approaches to generating realistic product scenarios and how these product scenarios should be prioritized, are discussed in [4]. Thus, we capture the uncertainties in functionality requirements using "future scenarios" [4], which are task sets that model functionalities likely to be added in the future.

We assume that the hardware architecture is fixed, and we want to decide the mapping of tasks to processing elements (PEs), such that the application is schedulable, even considering the uncertainties. A straightforward solution to tackle uncertainties is to over-design the system, e.g., to build a lot of spare capacity, but this is often prohibitively expensive. As an alternative, researchers have proposed *adaptive systems*, which change at runtime their configuration (e.g., re-mapping tasks using task migration) in response to changes in the requirements, execution times, environment, etc. However, many hard real-time applications are safety critical, where online task migration is not feasible, and an offline reconfiguration, e.g., task re-mapping, may be very costly. Hence, we want to derive, early on, a mapping of tasks to PEs, which is both *robust* and *flexible*. In our case, robust means that the application has a high chance of being schedulable, considering the *wcet* uncertainties, whereas a flexible mapping has a high chance to successfully accommodate future scenarios.

For time-triggered systems using static-cyclic scheduling, researchers have proposed approaches for the synthesis of flexible schedules, which can accommodate future changes. In [5], the future applications are captured using a set of possible *wcet*s and their probability distributions, and the flexibility is measured as the likelihood of successfully adding new functionality in the future. In [6], the flexibility is defined as both "extensibility", i.e., the maximum increase in the *wcet* of a task that can be handled without rescheduling, and "scalability", i.e., the maximum *wcet* of a new task that a schedule can accommodate without change. The work in [7] uses the info-gap decision theory to synthesize robust FlexRay bus schedules, considering uncertainties in design parameters, such as the size of a message.

In this paper we consider that the tasks are scheduled using *fixed-priority preemptive scheduling* (*fpps*). In this context, robustness has been addressed using "sensitivity analysis" [8], where the attributes of an implementation, e.g., worst-case response times, are evaluated against changes in system prop-

978-3-9810801-8-6/DATE12/©2012 EDAA

erties, e.g., *wcet*s. In [3], the mapping is fixed, and the uncertainties in the *wcet*s are captured using the "percentile method". The author proposes a Monte Carlo simulation approach to evaluate the likelihood that tasks will meet their deadlines. Other researchers [9] quantify the robustness in terms of a "revision cost", and they aim to provide robust designs, which minimize the revision cost.

Regarding flexibility, [10] proposes a mapping technique to increase the chance of successfully accommodating future applications. The future applications are captured similar to [5], using probabilities for *wcet* values. The approach in [11] defines flexibility in terms of the amount of functionality that the design is able to implement, and proposes a flexibility vs. cost trade-off model to search for Pareto-optimal solutions. In [4], researchers model the uncertainty in functionality requirements using scenarios, i.e., prioritized task sets, and derive an architecture and a mapping of tasks such that the flexibility is maximized. The flexibility is defined as the likelihood of all tasks being schedulable when adding a scenario to the existing mapping.

In this paper we are interested to derive a robust and flexible mapping solution during early design phases, taking into account the uncertainties. The problem is defined in Section III, and the uncertainty models are presented in Section II. Section V presents a motivational example which shows the importance of considering the uncertainties. In order to address both robustness and flexibility simultaneously, we propose a Genetic Algorithm-based approach to search for a *Pareto-front* of solutions (Section VI). The evaluation of the proposed approach is presented in Section VII. In the last section, we draw the conclusion of our work.

II. System model

In this paper, a system is composed of software applications, modeled as a set of tasks, and a hardware architecture consisting of a set $\mathcal{N}$ of PEs, interconnected by a communication channel, which is a bus on which messages are exchanged between tasks mapped on different PEs.

The mapping of a task τ_i to a PE N_j is captured by a mapping function: $\mathcal{M}(\tau_i) = N_j$. This mapping is not yet known and will be decided by our proposed approach. For each task τ_i, we assume that the period T_i and deadline D_i are known and given such that $D_i \leq T_i$. Tasks are scheduled using *fpps*.

A. Modeling wcet uncertainties

In this paper, we are not interested in modeling the variability of the execution time e_i of a task τ_i, but in modeling the uncertainties of the *wcet* c_i. The variability of e_i is typically captured by a probability mass function [12] and is due to, for example, the variations in the input data of tasks or the speculative features of modern processors.

In our case, the uncertainties in a *wcet* c_i come from the lack of information during the early design stages, when, for example, the algorithm used to implement a task τ_i or the parameters of the architecture are not yet known. Similar to [3], we use the "percentile method" to model the *wcet* uncertainties. Thus, we capture the *wcet* of a task by two values, the 50^{th} and 90^{th} percentiles. This means that in 50% of the possible future implementations, a task will have a *wcet* smaller or equal to the 50^{th} percentile value, while in the majority of the cases, i.e., 90%, the *wcet* will not exceed the 90^{th} percentile value.

Tasks	N_1 50^{th}	N_1 90^{th}	N_2 50^{th}	N_2 90^{th}	$T_i = D_i$
τ_1	10	20	15	30	50
τ_2	25	50	37.5	75	100
τ_3	40	60	60	90	150
τ_4	60	72	90	108	300

Fig. 1 Uncertainties in *wcet*s

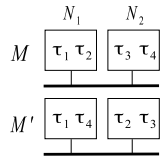

Fig. 2 Two mapping alternatives

Fig. 1 presents an example in which four tasks, τ_1 to τ_4, can be mapped on two PEs, N_1 and N_2. For each task τ_i and its mapping on each PE N_j, two *wcet* values are given, i.e., the 50^{th} and 90^{th} percentile, respectively. The more information is available about a task and the PEs where it can potentially be mapped, the smaller the difference between the two percentiles. A *wcet* can also be a fixed value, e.g., for legacy tasks mapped on legacy PEs. In our example, τ_4 is an updated task, thus designers believe that its 90^{th} percentile is only 20% larger than its 50^{th} percentile. For τ_3, this difference is 50%. However, tasks τ_1 and τ_2 are new tasks to be introduced, hence designers have lower confidence and thus their 90^{th} percentiles are twice the 50^{th} percentiles.

We use a Gumbel distribution, with the *cumulative distribution function* (*cdf*) defined as [13]: $P(c_i \leq x) = e^{-e^{-\frac{x-\mu}{\beta}}}$, where P is the probability that the *wcet* c_i will have a value smaller or equal to x. Using the two percentile values, corresponding to 0.5 and 0.9 probability, we can determine the distribution parameters μ and β, i.e., determine the *cdf* of the *wcet*. For example, for τ_1 mapped on N_1 in Fig. 1, we have $\mu = 8.05$ and $\beta = 5.31$. Knowing the *cdf* for a task τ_i, we can also determine its *probability density function* (*pdf*) σ_i.

B. Modeling functionality uncertainties

Additionally, product requirements often change during later design and development phases. We use the approach from [4] to describe an application as one baseline functionality, S_0, and a set of future scenarios, $\mathcal{S}_f = \{S_1, S_2, \ldots\}$. Each future scenario is associated a weight w_i, which reflects the probability of this future scenario becoming a reality.

Tasks	N_1 50^{th}	N_1 90^{th}	N_2 50^{th}	N_2 90^{th}	$T_i = D_i$
S_0: Example in Fig. 1					
$S_1 = S_0 \setminus \tau_1 \cup \tau_5$ ($w_1 = 0.8$)					
τ_5	10	20	15	30	50
$S_2 = S_0 \cup \tau_6$ ($w_2 = 0.4$)					
τ_6	25	50	37.5	75	100
$S_3 = S_0 \cup \tau_7 \cup \tau_8$ ($w_3 = 0.6$)					
τ_7	30	60	45	90	300
τ_8	50	75	75	102.5	300
$S_4 = S_0 \setminus \tau_1 \cup \tau_5 \cup \tau_6$ ($w_4 = 0.2$)					
τ_5	10	20	15	30	50
τ_6	25	50	37.5	75	100

Fig. 3 Modeling uncertainties in functionality requirements

Let us consider the example in Fig. 3, which includes one baseline functionality S_0 (tasks τ_1 to τ_4 from Fig. 1), and four future scenarios S_1 to S_4. We consider an architecture of two PEs, N_1 and N_2 (the same as in Fig. 1) . S_1 replaces τ_1 with τ_5 due to a functionality update. S_2 introduces τ_6 for enhancing the application performance. In S_3, a new application is added, which is modeled by τ_7 and τ_8. S_4 is the combination of S_1 and S_2, which captures the case when both S_1 and S_2 happen at the same time. Next to each scenario S_i, we also specify its weight w_i. These scenarios, and their associated weights, are determined using the methods presented in [4].

III. Problem Formulation

As an input to our problem, we have the hardware architecture $\mathcal{N}$, the baseline functionality S_0 and the set of future scenarios $\mathcal{S}_f$. For each task, we know the two *wcet* percentile values, on every PE where it is considered for mapping.

We are interested to determine the mapping M_0 of the baseline functionality S_0 on the given architecture $\mathcal{N}$, such that the robustness and flexibility of M_0 is maximized. These two metrics will be formally defined in the next subsection. A mapping M_0 is robust if the tasks in S_0 have a high chance of being schedulable. M_0 is flexible if it has a high chance to successfully accommodate the future scenarios from $\mathcal{S}_f$, such that they are also likely to be schedulable. This is a two-objective optimization problem (robustness and flexibility). Our optimization strategy, presented in Section VI, will produce a Pareto-front of solutions.

We target safety-critical hard real-time applications, so we consider that M_0 is fixed when adding a future scenario. Our optimization strategy will produce the mappings M_i of future scenarios S_i, as a byproduct of evaluating the flexibility of M_0. In the later design stages, when a scenario S_i has become a reality, we use our proposed mapping optimization strategy to decide the mapping M_i of S_i, while keeping the mapping of tasks in S_0, decided during the early design phases, fixed.

A. Robustness and Flexibility

We use the "degree of schedulability" to characterize the schedulability of a given mapping alternative M,

$$r_M = \begin{cases} d_1 = \sum_i \max(0, r_i - D_i) & if \quad d_1 > 0 \\ d_2 = \sum_i (r_i - D_i) & if \quad d_1 = 0 \end{cases} \tag{1}$$

where r_i is the *worst-case response time* (*wcrt*) of a task τ_i and D_i is its deadline. If a mapping is not schedulable, there exists at least one r_i greater than the deadline D_i, therefore the term d_1 of the function will be positive. In this case r_M is equal to d_1. However, if a mapping is schedulable, then each r_i is smaller than its corresponding deadline D_i. In that case $d_1 = 0$ and we use d_2 as the r_M, to be able to differentiate between two mapping alternatives, both leading to feasible schedules. $r_M \leq 0$ means the mapping is schedulable. A larger negative value of r_M indicates the mapping is "more schedulable", i.e., the *wcrt*s are smaller.

Note that r_M is a stochastic variable, since it is calculated based on *wcrt*s r_i, and each r_i is determined by the related *wcet*s c_i, see Section IV.

The robustness of a mapping M_k ($k = 0, 1, \ldots$), for the tasks in a task set S_k, is defined as the probability of all tasks in S_k being schedulable,

$$\mathcal{R}_{M_k} = P(r_{M_k} \leq 0) \tag{2}$$

where r_{M_k} is the degree of schedulability from Eq. 1.

Let us denote M_i, the mapping of the tasks in a future scenario $S_i \in \mathcal{S}_f$, on top of the mapping M_0 of the baseline functionality S_0, such that the robustness $\mathcal{R}_{M_i}$ is maximized. Then, the flexibility $\mathcal{F}_{M_0}$ of M_0 is defined as,

$$\mathcal{F}_{M_0} = \frac{\sum_{i=1}^{|\mathcal{S}_f|} w_i \times \mathcal{R}_{M_i}}{\sum_{i=1}^{|\mathcal{S}_f|} w_i} \tag{3}$$

where w_i is the weight of scenario S_i, and $|\mathcal{S}_f|$ is the number of future scenarios. To calculate $\mathcal{F}_{M_0}$, we need first to determine the mapping M_i of each S_i, such that $\mathcal{R}_{M_i}$ is maximized (See Section VI-A).

IV. Schedulability Analysis

We are interested to determine the probability $\mathcal{R}_{M_k}$ of a mapping M_k to be schedulable. This value is used in both metrics, robustness ($\mathcal{R}_{M_0}$) and flexibility ($\mathcal{R}_{M_i}$, $i = 1, 2, \ldots$). In this paper, we assume that tasks are scheduled using a fixed-priority preemptive scheduling policy, and we use a Response Time Analysis (RTA) [14] to determine the *wcrt* r_i of a task τ_i, according to the recurrence equation:

$$r_i^{n+1} = c_i + \sum_{\forall \tau_j \in hp(\tau_i)} \left\lceil \frac{r_i^n}{T_j} \right\rceil c_j \tag{4}$$

where $hp(\tau_i)$ is the set of tasks that have a priority higher than the priority of τ_i.

This basic analysis has been extended over the years [14] to take into account blocking times, arbitrary deadlines and release times, jitter, offsets, etc. Our analysis for uncertain *wcet*s uses a RTA inside an iterative loop. For simplicity, in this paper, we have decided to consider the case when $D_i \leq T_i$ and ignore the messages. The RTA is orthogonal to our analysis, and can be extended to consider a more general case.

In [12], a stochastic schedulability analysis is used to handle the variability in e_i. Each job $J_{i,j}$ of a task τ_i may have different execution times, depending on the probability distribution function ξ_i of e_i. Thus, for calculating r_i, the updated response time equation (Eq. 4) [12] uses stochastic variables of c_i and c_j. In each iteration of the recurrence equation for r_i, c_i and c_j will get different values, based on their *pdf*s of ξ_i and ξ_j, respectively. However, such a solution is not applicable in our case, where each job $J_{i,j}$ of a task τ_i has the same *wcet* c_i in each iteration.

The analysis in [3] uses Monte Carlo Simulation (MCS) to determine the probability distribution of r_i. With MCS, a large number of iterations are run, and the following steps are performed. First, for each task τ_i, a value of c_i is generated based on its *wcet pdf*, σ_i. Second, the generated values of c_i are used to determine the *wcrt* r_i of each task τ_i with Eq. 4.

Then, the degree of schedulability r_M is calculated using Eq. 1. Finally, the r_M values are collected over all iterations, and thus the degree of schedulability *pdf* is obtained.

MCS requires a large number of iterations (e.g., 100,000) to get an accurate result, which is time-consuming, and thus we cannot use MCS during design space exploration. In this paper, instead of MCS, we propose using the Kernel Smoothing Density Estimate (KSDE) technique [15] to quickly approximate the degree of schedulability *pdf*.

Similar to MCS, we start by performing a number of iterations to get the degree of schedulability values. However, we need fewer samples for KSDE (e.g., 1,000 instead of 100,000 in MCS), since a smoothing technique is applied to estimate the *pdf* based on the available samples. Given m random samples $X_1, \ldots, X_m$ whose underlying density f is to be estimated, KSDE uses a kernel density estimator,

$$\widehat{f}(x,h) = \frac{1}{mh}\sum_{i=1}^{m} K\left(\frac{x - X_i}{h}\right) \tag{5}$$

where $K(\frac{x-X_i}{h})$ is the kernel and h (> 0) is the bandwidth. The bandwidth h is a smoothing parameter, which controls how wide the probability mass is spread around a sample.

We evaluated several kernels and bandwidths, and compared the results with those obtained by MCS. We decided to use a *normal* kernel,

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{6}$$

and a *normal* optimal smoothing h,

$$h = \frac{\Phi(X_i - \Phi(X_i))}{0.6745} \cdot \left(\frac{4}{3m}\right)^{\frac{1}{5}} \tag{7}$$

where $\frac{\Phi(X_i-\Phi(X_i))}{0.6745}$ is a robust estimate for standard deviation of the distribution, and $\Phi(X_i)$ denotes the median of X_i.

Considering the two mappings, M and M', from Fig. 2, both MCS and KSDE resulted in $\mathcal{R}_M = 93\%$ and $\mathcal{R}_{M'} = 67\%$, i.e., the probability of the tasks in S_0 to be schedulable is 93% (using M) and is 67% (using M'). The difference is that MCS took 25 seconds (using 100,000 samples), whereas KSDE finishes in 0.5 second (using 1,000 samples).

We have used both MCS and KSDE to determine the robustness of 20 task sets mapped on varying number of PEs. The maximum difference between the two techniques is 3%. Thus, we use KSDE as the basis for calculating the two objective functions during the optimization. Note that the analysis presented in this section is only used to guide the search, not to provide schedulability guarantees. We assume that a RTA will be used during the later design and development stages (when maybe more accurate information about *wcet*s and the functionality is available) to check the schedulability of an implementation.

V. Motivational Example

In the following, we show the importance of modeling and taking into account the uncertainties in the early design phases.

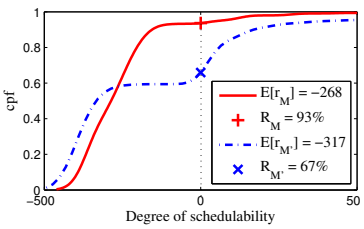

Fig. 4 Results of M and M'

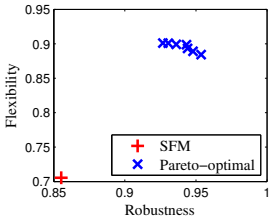

Fig. 5 *SFM and Pareto-optimal*

For comparison purposes, let us introduce a "straightforward mapping" (SFM) approach, which does not take into account the uncertainties. Thus, with SFM, we consider that each *wcet* is characterized by a single value (the expected value of the *wcet pdf* σ_i, denoted by $\mathrm{E}(\sigma_i)$), and the future scenarios are ignored. SFM determines that mapping which minimizes the $E(r_M)$, calculated using the expected *wcet* values in Eq. 1.

Let us assume that SFM has to decide between two mappings M and M' from Fig. 2, during design space exploration. Using the expected *wcet* values, $r_M = -260$, while, $r_{M'} = -317$, which means that M' would be preferred. However, we reach a different conclusion if we take into account the uncertainties in *wcet*s and compare the two mappings in terms of robustness. Fig. 4 presents the degree of schedulability *cpf*s for the two mappings. The probability of the mapping being schedulable $P(r_M \leq 0)$ is determined by the intersection of the *cpf* with the vertical line at point "0". As we can see, M has a better chance of being schedulable (93%) than M' (67%), so actually chosing M instead of M' is more "robust", i.e., it has a higher chance of being schedulable.

Let us consider the baseline functionality from Fig. 1, and the future scenarios from Fig. 3. We are interested to determine a mapping which maximizes robustness and flexibility. The Pareto-optimal solutions found after an exhaustive search, are depicted by a (blue) '×' in Fig 5. The rightmost '×' is the most robust mapping, with 95.4% robustness and 88.4% flexibility. The leftmost '×' is the most flexible mapping, with a robustness of 92.7% and a flexibility of 90.1%.

We have also plotted in Fig. 5 the optimal mapping obtained by SFM, using a red '+' symbol. The robustness and flexibility of this mapping have been calculated using Eq. 2 and Eq. 3, respectively, taking into account the uncertainty model from Fig. 1 and Fig. 3. As we can see, SFM produces a poor quality solution, with only 85.5% robustness and 70.6% flexibility.

VI. Mapping Optimization

We propose a Genetic Algorithm (GA)-based approach, called Mapping for Robustness and Flexibility (MRF), to solve the optimization problem presented in Section III. There are several off-the-shelf multiobjective GA implementations, such as NSGA-II [16] and search frameworks for multiobjective optimization such as PISA [17]. In this paper, we focus on determining the importance of modeling the uncertainties in the early design stages, and thus we decided to use the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [16], due to its good performance and its simple implementation.

GA is a metaheuristic optimization approach, which belongs to the class of Evolutionary Algorithms, inspired from the process of natural evolution. The set of candidate solutions is called a "population", and each solution is (i) *encoded* using a string called a "chromosome". The population is (ii) *initialized* to n candidate solutions, where n is the *population size*. The population is evolved by (iii) *selecting* a set of solutions and performing (iv) *recombination* and (v) *mutation* to generate offsprings. Finally, the parent population is (vi) *replaced* with an offspring population with better "fitness". The fitness of a solution is evaluated using our multiobjective function. Steps (iii) to (vi) are repeated until a *termination condition* is reached.

Steps (i) to (vi) are explained in the reminder of this section. There are several choices for their implementation. Through experiments, we decided to choose the following approaches, which can find good solutions in a reasonable time. The parameters were also determined experimentally. One example of parameters is given in Section VII.

(i) *Encoding*: We use direct-value encoding, where each chromosome represents a mapping alternative, and each allele (the value of a gene, which is a component of a chromosome) represents a PE. For example, the mapping of τ_1 and τ_2 to N_1, τ_3 and τ_4 to N_2 is described by the string 1|1|2|2, where i^{th} position in the string is the index j of the PE N_j of task τ_i.

(ii) *Initialization*: The initial population is randomly generated and has a population size n.

(iii) *Selection*: We use "tournament selection"' to select parents for performing recombination and mutation. In a tournament, four chromosomes are chosen at random, and the fittest one wins. In total, $2(p_c \times n)$ parents are chosen for performing recombination, while $n - (p_c \times n)$ parents are chosen for performing mutation, where p_c is the probability of recombination.

(iv) *Recombination* (also called *crossover*): We employ a standard single point crossover. For each two parents, we compare a randomly generated number with p_c, if this number $\leq p_c$, the two parents are cut at a random point and the sections after the cut point are swapped to generate the offsprings. Otherwise, the offsprings are just copies of their parents. For example, if 1|2|1|2 and 2|1|2|1 are decided to crossover on their third position, the offsprings are 1|2|2|1 and 2|1|1|2.

(v) *Mutation* is used to add diversity to a population obtained from recombination. For each position of a parent's string, we compare a randomly generated number with p_m (probability of mutation) and if this number $\leq p_m$, this position is mutated, i.e., the task is randomly remapped to another PE.

(vi) *Replacement*: Recombination and mutation generate n offsprings out of the n parents in the current population. Replacement decides which n solutions are kept out of the $2n$ solutions available. The key advantage of NSGA-II lies in how it performs selection and replacement, with the goal of preserving diverse non-dominated solutions, in the hope of finding the Pareto-optimal front. See [16] for the details on the selection and replacement procedures used in NSGA-II.

Steps (iii) to (vi) are repeated until there is no improvement for a given number of consecutive generations, e.g., 10. In the end, we obtain a Pareto-front of solutions, which, however, is not guaranteed to contain the Pareto-optimal, since NSGA-II is a search metaheuristic which does not guarantee optimality.

A. Determining the mappings of future scenarios

When measuring the flexibility $\mathcal{F}_{M_0}$ of a mapping M_0, we need to determine the mapping of each future scenario, S_i ($i = 1, 2, \ldots$), and calculate its robustness $\mathcal{R}_{M_i}$ using Eq. 3. To get an accurate flexibility value for $\mathcal{F}_{M_0}$, M_i should be as close as possible to the optimal, i.e., it has a maximum robustness value for $\mathcal{R}_{M_i}$. However, determining such optimal mappings is time-consuming, and the evaluation of flexibility is performed when visiting every M_0 alternative. Hence, we propose a Greedy algorithm to determine the mapping M_i of each future scenario S_i. Note that we only have to map those tasks from S_i which are not present in the baseline functionality S_0, i.e., they are new tasks. All the other tasks will keep the some mapping as in M_0. Thus, the new tasks in S_i are sorted according to their utilization, calculated using the expected *wcets*, i.e., $u_i = E(\sigma_i)/T_i$. Then each task is mapped on the PE with the lowest utilization, and the PE utilizations are updated before moving to the next task.

To determine the quality of the proposed Greedy algorithm, we have also implemented a GA-based mapping. The two algorithms have been evaluated using the synthetic benchmark *"synthetic 1"* (see Section VII). The difference is only 4%, but GA is 35-times slower than the greedy algorithm. Therefore, we have decided to use the Greedy algorithm instead of GA for determining the mapping of future scenarios.

VII. Experimental Results

To evaluate our proposed approach, we used four real-life benchmarks (Table II) from the Embedded System Synthesis Benchmark Suite (E3S), version 0.9 [18], and eight synthetic benchmarks (Table I) generated using Task Graphs For Free (TGFF) [19]. The details of the benchmarks are reported in Columns 1–4 of the tables. For the synthetic benchmarks, *wcet*s were generated in the range 30–70 ms. These values are considered as the *wcet*s with 50^{th} percentile. *wcet*s with 90^{th} percentile are generated to be up to 50% larger than their 50^{th} percentile.

For each benchmark, four future scenarios, S_1 to S_4, are considered. To create S_1, we have randomly selected 10% of tasks in S_0 and increased their 50^{th} percentile with 20% (and correspondingly adjusted their 90^{th} percentile). For S_2, we

Table I Synthetic Benchmarks

Test Set	Number of			SFM		MRF			
	PEs	Tasks				Most robust		Most flexible	
		S_0	S_f	$\mathcal{R}_{M_0}$	$\mathcal{F}_{M_0}$	$\mathcal{R}_{M_0}$	$\mathcal{F}_{M_0}$	$\mathcal{R}_{M_0}$	$\mathcal{F}_{M_0}$
synthetic 1	3	22	30	34%	17%	84%	56%	69%	64%
synthetic 2	3	22	30	75%	30%	96%	52%	87%	82%
synthetic 3	6	42	58	26%	22%	70%	23%	61%	63%
synthetic 4	6	42	58	77%	44%	95%	45%	89%	81%
synthetic 5	8	62	86	38%	13%	70%	23%	60%	33%
synthetic 6	8	62	86	81%	23%	97%	64%	93%	73%
synthetic 7	10	84	116	22%	6%	88%	18%	69%	27%
synthetic 8	10	84	116	74%	9%	88%	11%	81%	28%

Table II REAL-LIFE BENCHMARKS

Test Set	Number of			SFM		MRF			
	PEs	Tasks				Most robust		Most flexible	
		S_0	S_f	$\mathcal{R}_{M_0}$	$\mathcal{F}_{M_0}$	$\mathcal{R}_{M_0}$	$\mathcal{F}_{M_0}$	$\mathcal{R}_{M_0}$	$\mathcal{F}_{M_0}$
consumer-cords	2	12	16	67%	51%	96%	66%	93%	68%
networking-cords	2	13	17	75%	69%	86%	73%	80%	75%
auto-indust-cords	4	24	32	42%	12%	59%	38%	56%	41%
telecom-cords	4	30	42	46%	44%	97%	57%	91%	73%

have randomly introduced new functionality, which is about 10% of the size of S_0. S_3 is similar to S_2, but larger, 20% of S_0. Finally, S_4 is a combination of S_1 and S_2. The weights of four scenarios, S_1 to S_4, are 0.8, 0.4, 0.6 and 0.2, respectively.

In the first set of experiments (Table I), we were interested to determine the importance of capturing the uncertainties during the early design phases. We have varied the size of the system from 22 tasks (S_0) and 3 PEs to 84 tasks (S_0) and 10 PEs and applied two optimization approaches: MRF, presented in the previous section, which takes into account both uncertainties, and SFM, introduced in Section V, which uses the expected values of *wcet*s and ignores the future scenarios.

The robustness and flexibility of SFM are reported in columns 5 and 6 of Table I. MRF produces a Pareto-front of solutions. Due to a lack of space, we show only the Pareto-front of *"synthetic 1"* (Fig. 6), and report only the extremes in the Pareto-front of all synthetic benchmarks, the most robust solution (columns 7 and 8) and the most flexible solution (columns 9 and 10).

We have tuned the NSGA-II parameters such that the results are as close as possible to the optimal (i.e., no improvements were seen after a very long runtime). Taking *"synthetic 7"* as an example, we set $n = 100$, $p_c = 0.4$, $p_m = 0.2$ and the search terminates if no improvement is seen after 10 generations. Both SFM and MRF are implemented in Matlab 2010 and run on an Intel Core i7 CPU 920 (2.67 GHz) computer. This resulted in runtime in between 0.5 and 3.5 hours.

As we can see from Table I, SFM is not able to find robust and flexible solutions, whereas our MRF approach is able to find good quality solutions, where the robustness and flexibility is significantly increased compared to SFM.

In the second set of experiments, we evaluated the MRF approach using four real-life benchmarks from E3S. The experimental setup details and the results obtained are presented in Table II. Due to a lack of space, we show only the Pareto-front of *"consumer-cords"* (Fig. 7), and report only the extremes in the Pareto-front of all real-life benchmarks in columns 7–10 of Table II. The evaluation confirms the results obtained from the synthetic benchmarks.

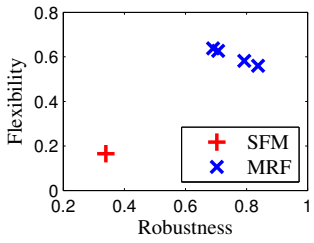

Fig. 6 *synthetic 1*

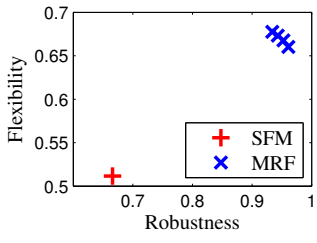

Fig. 7 *consumer-cords*

VIII. CONCLUSION

In this paper, we have addressed the mapping of hard real-time applications on distributed heterogeneous architectures, during the early design phases. We have considered a fixed-priority preemptive scheduling policy, where the system schedulability is determined using a response time analysis. We have modeled the uncertainties in *wcet*s and functionality requirements, and we have used the Kernel Smoothing Density Estimate as the basis for determining the schedulability probability of a mapping alternative. We have proposed a GA-based mapping optimization targeting both robustness and flexibility, related to the uncertainties in *wcet*s and functionality requirements, respectively. The results obtained on the synthetic and real-life benchmarks show the importance of modeling the uncertainties during the early design phases, and taking them into account during design space exploration.

REFERENCES

[1] L. Lavagno and C. Passerone, "Design of embedded systems," *Embedded Systems Handbook*, 2005.

[2] J. Axelsson, "Cost models with explicit uncertainties for electronic architecture trade-off and risk analysis," *Intl. Council on Systems Engineering (INCOSE)*, 2006.

[3] ——, "A method for evaluating uncertainties in the early development phases of embedded real-time systems," in *Proc. RTCSA*, 2005.

[4] I. Bate and P. Emberson, "Incorporating scenarios and heuristics to improve flexibility in real-time embedded systems," in *Proc. RTAS*, 2006, pp. 221–230.

[5] P. Pop, V. Izosimov, P. Eles, and Z. Peng, "Design optimization of time- and cost-constrained fault-tolerant embedded systems with checkpointing and replication," *IEEE Trans. on VLSI Systems*, vol. 17, no. 3, pp. 389–402, 2009.

[6] W. Zheng, J. Chong, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli, "Extensible and scalable time triggered scheduling," in *Application of Concurrency to System Design*, 2005, pp. 132–141.

[7] A. Ghosal, H. Zeng, M. Di Natale, and Y. Ben-Haim, "Computing robustness of flexray schedules to uncertainties in design parameters," in *Proc. DATE*, 2010, pp. 550–555.

[8] R. Racu, A. Hamann, and R. Ernst, "A formal approach to multi-dimensional sensitivity analysis of embedded real-time systems," 2006.

[9] M. Lukasiewycz, M. Glaß, and J. Teich, "Robust design of embedded systems," in *Proc. DATE*, 2010, pp. 1578–1583.

[10] P. Pop, P. Eles, and Z. Peng, "Flexibility driven scheduling and mapping for distributed real-time systems," in *RTCSA*, 2002.

[11] C. Haubelt, J. Teich, K. Richter, and R. Ernst, "System design for flexibility," in *Proc. DATE*, 2002, pp. 854–861.

[12] K. Kim, J. Diaz, L. Bello, J. Lopez, C. Lee, and S. Min, "An exact stochastic analysis of priority-driven periodic real-time systems and its approximations," *IEEE Trans. on Computers*, vol. 54, no. 11, pp. 1460–1466, 2005.

[13] S. Kotz and S. Nadarajah, *Extreme value distributions: theory and applications*. World Scientific Publishing Company, 2000.

[14] C. Fidge, "Real-time schedulability tests for preemptive multitasking," *Real-Time Systems*, vol. 14, no. 1, pp. 61–93, 1998.

[15] A. Bowman and A. Azzalini, *Applied smoothing techniques for data analysis: the kernel approach with S-Plus illustrations*. Oxford University Press, USA, 1997, vol. 18.

[16] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *Parallel Problem Solving from Nature PPSN VI*. Springer, 2000, pp. 849–858.

[17] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "Pisa – a platform and programming language independent interface for search algorithms," in *Evolutionary multi-criterion optimization*. Springer, 2003, pp. 1–1.

[18] R. Dick, "Embedded system synthesis benchmarks suite," 2002.

[19] R. Dick, D. Rhodes, and W. Wolf, "Tgff: task graphs for free," in *Proc. workshop on Hardware/software codesign*. IEEE Computer Society, 1998, pp. 97–101.