Design Space Exploration and System Optimization with SymTA/S - Symbolic Timing Analysis for Systems

Arne Hamann, Marek Jersak, Kai Richter, Rolf Ernst
Institute of Computer and Communication Network Engineering
Technical University of Braunschweig
D-38106 Braunschweig / Germany
{**hamann**|**jersak**|**richter**|**ernst**}**@ida.ing.tu-bs.de**

ABSTRACT

The increasing complexity of heterogeneous SoC and distributed systems confronts the system designer with problems how to determine reasonable design alternatives leading to well functioning systems. Ideally, a designer would try all possible system configuration and choose the best one regarding specific system requirements. Unfortunately, such an approach is not possible because the high number of design parameters in complex systems leads to a very large design-space, prohibiting an exhaustive search. Consequently, good search techniques are needed to find optimal, or at least good, design alternatives. In this paper, we present a design space exploration framework for system optimization using SymTA/S, a software tool for formal performance analysis. In contrast to many previous approaches, our approach takes the hierarchical structure of the design space of heterogeneous SoC and distributed systems into account, allowing the designer to control the exploration process. A main technique in our approach is systematic system optimization using traffic shaping.

I. INTRODUCTION

A major problem during the design of complex heterogeneous embedded systems is that it is not obvious which design alternatives make sense and lead to good system behavior. For this reason it is important to evaluate a large number of alternative architectures and implementation alternatives. Ideally, the designer would try all possible alternatives and choose the best regarding specific system requirements. Unfortunately, this is not possible because the high number of design parameters in complex systems leads to a very large design-space, prohibiting an exhaustive search. Consequently, good exploration techniques are needed to find optimal, or at least good, design alternatives.

Manual design space exploration heavily reduces design productivity. It is highly desirable to automate at least part of the process. Of course, even automatic exploration cannot search the whole design space in reasonable time. Therefore, it is important to find an appropriate sub search space containing good solutions. Restriction of the search space to crucial system parameters is necessary to allow an efficient search for good design alternatives.

In this paper we present a framework for design space exploration and system optimization using SymTA/S. An important aspect in our approach is the introduction of traffic shaping as search parameter. Traffic shaping weakens functional and non-functional performance dependencies and allows to find working system configurations which are not possible without traffic modulation.

Our exploration framework provides the designer with the possibility to perform several exploration steps in locally restricted search spaces. This approach allows him to control the exploration process and provides him insight to system-level performance dependencies. Based on this knowledge the designer can identify interesting design sub-spaces, worthy to be searched in-depth or even completely. An a priori global exploration does not permit such a flexibility and neglects the structure of the design space, giving the designer no possibility to modify and select the exploration strategy. In the worst-case, when the composition of the design space is unfavorable, this can lead to nonsatisfying results with no possibility for the designer to intervene. In many approaches the only possibility for the designer in such a case consists in restarting the exploration, hoping for better results.

The remainder of this paper is structured as follows. After an overview of related work we will give a brief introduction into the formal core of SymTA/S [6], a software tool for formal performance analysis. Afterwards, we will review the theoretical background of the traffic shaping mechanism used in SymTA/S. We will then explain how SymTA/S is used to conduct user controlled design space exploration. Finally, we describe a synthetical SoC example and perform several exploration steps in order to optimize its performance.

II. RELATED WORK

There is a large body of work in the area of design space exploration for system optimization. However, nearly all approaches concern specific domains or focus on a small set of system parameters and optimization objectives. Moreover, the underlying analysis techniques often limit the supported system architectures.

The approach described in [21] introduces an analysis technique to estimate end-to-end packet delays and queuing memory in network processor architectures. Based on this information a measure is defined to characterize the performance of such architectures under different usage

scenarios. By means of design space exploration pareto-optimal architectures are searched trading good performance under several usage scenarios versus cost. In [11] the authors treat the reverse problem. Instead of determining worst-case buffer requirements and output stream properties for given input streams and scheduling policies, the authors search for the input stream rates that can be supported by a given stream processing architecture without violating on-chip buffer constraints. The authors propose the integration of this technique into a tool for automated design space exploration for fast performance evaluation of different stream processing architectures.

[4] presents a heuristic algorithm for priority assignments in distributed hard real-time systems to optimize end-to-end deadlines. The algorithm iteratively decomposes the global deadlines into artificial local deadlines and then assigns deadline monotonic priorities on each resource. This approach is thus not applicable to more general priority assignments or other types of scheduling policies.

The approach in [15] focuses on bus access optimization (TDMA and static priority preemptive) in multi-cluster embedded systems interconnected via gateways. Thereby, the application structure is feed-forward. Optimization objectives are end-to-end deadlines. The authors propose a partitioning and mapping heuristic and a heuristic adjusting TDMA slot sizes in time-triggered clusters. For the priority assignments in event-triggered clusters the heuristic presented in [4] is used. The heuristics work well for the considered feed-forward applications. However, it is not yet published how they perform for more complex application structures.

[5] describes the *Platune* framework allowing performance and power tuning of a specific parameterized SoC platform. For a given application to be mapped on the target SoC, *Platune* determines all sets of architectural parameter values representing pareto-optimal solutions regarding power and performance. The detection of all pareto-optimal solutions is achieved effectively by clustering the search space into independent parts, for which pareto-optimal solutions can be determined separately.

The *Spacewalker* [20], part of the *PICO* project from HP Labs, pursuits a similar approach. For given applications, it searches for pareto-optimal embedded computer systems. The search space is explored using a divide-and-conquer approach. In the first step different subsystem are explored independently. From the sets of obtained pareto-optimal subsystems, global systems are constructed and evaluated. This hierarchical exploration approach seems to work well for the architecture presented in the paper. However, for performance dependent subsystems the combination of local pareto-optima rarely leads to global pareto-optima.

III. The SymTA/S approach

SymTA/S [6] is a software tool for formal performance analysis of heterogeneous SoCs and distributed systems. The core of SymTA/S is our recently developed technique to couple scheduling analysis algorithms using event streams [16], [18]. Event streams describe the possible I/O timing of tasks and are characterized by appropriate event models such as periodic events with jitter or bursts and sporadic events. At the system level, event streams are used to connect local analyses according to the systems application and communication structure.

In contrast to previous work, SymTA/S explicitly supports the combination and integration of different kinds of analysis techniques known from real-time research. For this purpose, it is essential to transition between the often incompatible event stream models resulting from the dissimilitude of the local techniques. This kind of incompatibility appears for instance between an analysis technique assuming periodic events with jitter and an analysis technique requiring sporadic events. In SymTA/S we use *event model interfaces (EMIFs)* and *event adaptation functions (EAFs)* to realize these essential transitions [16].

However, integration of heterogeneous systems is not the sole domain of application for EMIFs and EAFs. In SymTA/S so-called shapers can be connected with any event stream. Shapers are basically EMIF-EAF combinations which manipulate an event stream, and thus the interaction between two components. They provide control about the timing of exchanged events and data and consequently also about performance dependencies. We have shown in [17] that this is especially important to break up non-functional dependency cycles and to reduce transient load peaks in dynamic systems. In other words, due to the event model transformation provided by EMIFs and EAFs, SymTA/S is able to analyze many real world examples that holistic approaches [24], [14] cannot handle.

In order to perform a system level analysis, SymTA/S locally performs existing scheduling analyses (e.g. RMS, TDMA, Round Robin, etc.) and propagates their results to the neighbouring components. This analysis-propagate mechanism is repeated iteratively until all components are analyzed, which means that all output streams remained unchanged.

The above described basic SymTA/S approach has been recently extended to support multi-rate systems, tasks with multiple activating inputs (OR- or AND-concatenated), conditional communication and functional cycles [7], [9]. These major extensions enable SymTA/S to cope with complex applications.

Furthermore, SymTA/S is able to consider system context information to tighten analysis bounds. We define as a system context all kinds of correlations between activating events that go beyond the possible timing of consecutive events in one event stream. *Inter* event stream contexts, initially introduced by Tindell [25] and generalized by Palencia and Harbour [13], consider possible phases between

events in different event streams, thus allowing to calculate a tighter number of interrupts of a task by other tasks sharing the same component. *Intra* event stream contexts, initially introduced by Mok and Chen [12], consider correlations between successive computation or communication requests, thus allowing to calculate a tighter load for a number of successive activations of a task. Both types of contexts lead to the calculation of shorter worst-case, and longer best-case response times. In [8] we presented the generalization of intra event stream contexts, the combination of both types of contexts during analysis, and explicit distinction between different types of events on one hand, and different task behaviors on the other. The latter is crucial for subsystem integration and compositional performance analysis, since different types of events are a property of the sender, while different behaviors are a property of the receiver.

IV. Traffic shaping

Scheduling and data dependent behavior induce jitter to the input-output timing of processes and communication [17]. Such jitters accumulate in the system and can lead to event bursts. Both effects increase timing uncertainty and worst-case peak load.

Such peak loads caused by bursty streams can be controlled by modulating the maximum number of events per time, called traffic shaping. Traffic shaping reduces the impact of an event stream on other streams at the cost of a potentially increased latency of the controlled stream. The shaping effects are rather complex and require special modeling considerations that will be explained in the following.

A bursty event stream is defined by three parameters, an average period T, a maximum allowed jitter J, and a minimum event distance d^- during bursts. As a popular measure of system load in scheduling analysis, the $\eta^+(\Delta t)$ function determines the maximum number of events η^+ for a given interval of time Δt. Small time intervals are dominated by bursty behavior, where the system load is only limited by the minimum event distance d^-. Larger observation intervals reveal the generally periodic nature of the event stream. The *arrival curve* [23] in figure 1 illustrates the two different regions. The η^+ function of the stream is the minimum of both regions:

$$\eta^+_{\text{in}}(\Delta t) = \min\left(\left\lceil \frac{\Delta t}{d^-}\right\rceil, \left\lceil \frac{\Delta t + J}{T}\right\rceil\right). \qquad (1)$$

Using time-out buffers, designers can deliberately enforce an additional bound on the minimum event distances. Such time-out buffers represent *traffic shapers* that are inserted in the design between two application components. The time-out mechanism buffers incoming events such that no two successive events are released earlier in time than $d^-_{\text{time out}}$.

According to the extended real-time calculus approach of Thiele et. al. [22], the shaper defines a sporadic upper-bound *service curve* [23] with $\eta^+_{\text{time out}}(\Delta t) = \left\lceil \frac{\Delta t}{d^-_{\text{time out}}}\right\rceil$. The shapers output arrival curve can be calculated from both, input arrival curve $\eta^+_{\text{in}}(\Delta t)$ and shaper service curve $\eta^+_{\text{time out}}(\Delta t)$. The calculations are usually very complex and their general application to arbitrary arrival and service curves, as proposed in [22], is extremely doubtful. In case of traffic shapers, however, the complex real-time calculus equations can be easily reduced to

$$\begin{aligned}\eta^+_{\text{shaped}}(\Delta t) &= \min\left(\eta^+_{\text{timeout}}(\Delta t), \eta^+_{\text{in}}(\Delta t)\right)\\ &= \min\left(\left\lceil \frac{\Delta t}{d^-_{\text{timeout}}}\right\rceil, \left\lceil \frac{\Delta t}{d^-_{\text{in}}}\right\rceil, \left\lceil \frac{\Delta t + J}{T}\right\rceil\right).\end{aligned}$$

The larger value of d^-_{in} and $d^-_{\text{time out}}$ will dominate the other, and we can further reduce the $\eta^+_{\text{shaped}}(\Delta t)$ function to the η^+ function of an event stream with burst as introduced by equation 1. In case of $d^-_{\text{time out}} \leq d^-_{\text{in}}$, the shaper does not actually represent an additional constraint. In other words, the shaper is "inactive", no events are buffered and the output arrival curve equals the input arrival curve.

Obviously more interesting is the case of $d^-_{\text{time out}} > d^-_{\text{in}}$. Input events are buffered and the shaper "flattens" the burst slope of the output arrival curve according to $d^-_{\text{time out}}$: $\eta^+_{\text{shaped}}(\Delta t) = \min\left(\left\lceil \frac{\Delta t}{d^-_{\text{time out}}}\right\rceil, \left\lceil \frac{\Delta t + J}{T}\right\rceil\right)$. Figure 2 illustrates this behavior. The arrival curve with a minimum distance of d^-_{in} is above the service curve with a minimum event distance of $d^-_{\text{time out}}$. The block arrows indicate buffering. Thiele et. al. already recognized that the vertical distance between the arrival and the service curve captures the so

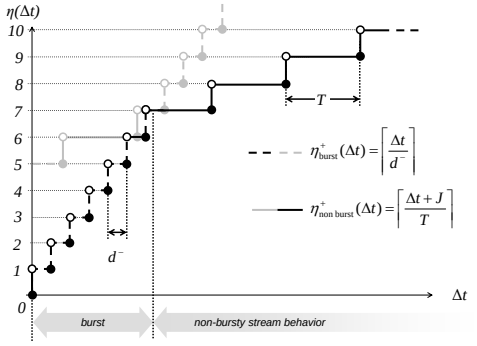


Fig. 1 - Event Arrival Curve of Incoming Event Stream

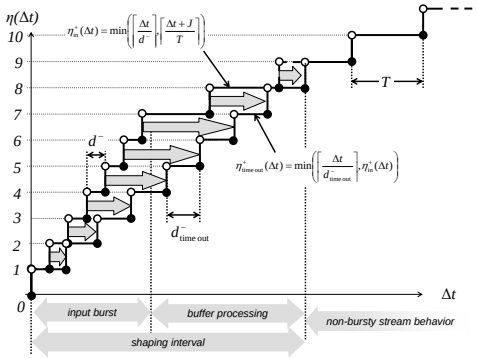


Fig. 2 - Event Arrival Curve of Incoming Event Stream

called backlog [23], i. e. the number of buffered events at a given point in time: $backlog(\Delta t) = \eta^+_{\text{in}}(\Delta t) - \eta^+_{\text{time out}}(\Delta t)$.

The vertical distance between the curves, i. e. the length of the arrows in the figure, represents the delay of the corresponding event. The calculations are slightly more sophisticated than the *backlog*, although the specialties of traffic shaping reduce the complexity of the general real-time calculus theory [23]. We recently introduced another function $\delta^-(n)$ that determines the minimum distance between *n* successive events [19]. Roughly speaking, $\delta^-(n)$ is the inverse of $\eta^+(\Delta t)$ since it returns the earliest time Δt at which the *n*th event ($n \geq 2$) can arrive after the first one. For the bursty arrival curve and the sporadic service curve, these are given by $\delta^-_{\text{in}}(n) = \max\left((n-1)d^-_{\text{in}}, (n-1)T - J\right)$ and $\delta^-_{\text{time out}}(n) = (n-1)d^-_{\text{time out}}$. Hence, the delay is given by: $delay(n) = \delta^-_{\text{time out}}(n) - \delta^-_{\text{in}}(n)$.

The sought-after maxima of $backlog_{\max} = \max_{\Delta t>0} backlog(\Delta t)$ and $delay_{\max} = \max_{n\geq 2} delay(n)$ can be calculated through linearization of the discrete η^+ and δ^- functions. Details can be found in [19]. For this paper, the following qualitative explanation shall be sufficient. It should not surprise that the worst-case buffering and delay situation appears at the end of the input burst. At that time, *the most events* are stored "waiting" for being processed until the buffer is empty and the behavior returns to "non-bursty". And clearly the last event of the input burst has to *wait longest*.

Compared to full synchronization, traffic shapers provide promising peak load reduction and load balancing capabilities with smaller buffers and delays. And shapers allow to trade-off different design objectives. Larger $d^-_{\text{time out}}$ values result in more balanced system load and better schedulability, while they increase delays and buffering requirements along task chains (or paths). In this paper, the shapers time-out value is subject to system-level optimization.

V. Design space exploration in SymTA/S

In this section we will give an overview of the compositional design space exploration framework used in SymTA/S which is based on evolutionary optimization techniques. We will first describe system parameters which can be subject to optimization and how they can be composed to define the search space. Then we will give some examples of metrics expressing desired or undesired system properties, i.e. the optimization objectives. Finally, we will explain the iterative design space exploration performed in SymTA/S.

A. Search Space

We see the entire system as a set of independent *chromosomes*, each representing a distinct subset of system parameters. A chromosome carries the variation operators necessary for combination with other chromosomes of its type. In SymTA/S we currently use the standard operators mutation and crossover which are independently applied to the chromosomes. The scope of a chromosome is arbitrary, it reaches from one single system parameter to the whole system. Examples for reasonable independent chromosomes are:

- priority assignments of tasks on one or several priority-scheduled resources
- time slot sizes of tasks on one or several TDMA or round robin scheduled resources
- speed / throughput of one or several resources
- traffic shaping

Traffic shaping is included because it increases the design space and allows to find solutions which are not possible without traffic modulation. This shall be shown with a small example.

We consider the task set in table I scheduled according to the static priority preemptive policy. All tasks are activated periodically except $T0$ which has a very large jitter leading to the simultaneous arrival of 3 activations in the worst case.

Name	Activating Event Model	CET	Deadline
T0	$\mathcal{P}(100)$ + $\mathcal{J}(200)$ + $d(10)$	4	8
T1	$\mathcal{P}(100)$ + $\mathcal{J}(0)$ + $d(0)$	8	12
T2	$\mathcal{P}(100)$ + $\mathcal{J}(0)$ + $d(0)$	5	21
T3	$\mathcal{P}(100)$ + $\mathcal{J}(0)$ + $d(0)$	3	24

TABLE I - Simple Task Set

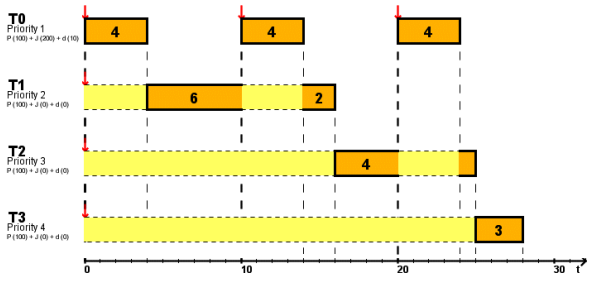
We conduct two experiments. The first one with the original activating event models and the second one using a shaper at the input of $T0$ extending the minimum distance to 12. In the first experiment we do not find a priority assignment leading to a system fulfilling all constraints. However, in the second experiment we find the priority assignment $T0 > T1 > T2 > T3$ leading to a working system. The reason for this is that extending the minimum distance of successive activations of $T0$ relaxes the impact of the burst and leads to more freedom for the lower priority tasks to execute. This results in less preemption and thus earlier completion for $T1$, $T2$ and $T3$. Figures 3(a) and 3(b) visualize this effect by showing the worst-case scheduling scenarios for the priority assignment $T0 > T1 > T2 > T3$ with minimum distances 10 and 12.

Note that in the general case concerning distributed systems with complex performance dependencies, optimization through traffic shaping is not applicable in such a straight forward manner. Nevertheless, traffic shaping can broaden considerably the solution-space by restricting event streams, leading to increased freedom on cross-related event streams. The example in section VII will underline this by means of a small but realistic example.

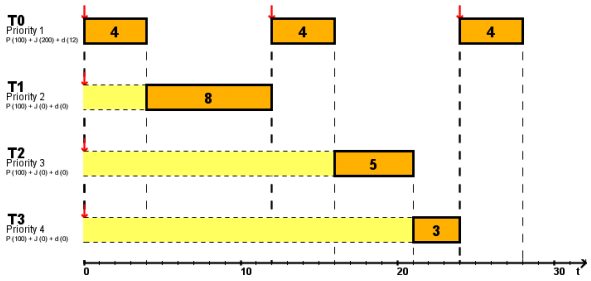
The designer defines the current search space, by selecting and configuring the set of chromosomes representing the desired search space. System parameters not included inside the selected chromosomes remain immutable.

Figure 4 shows this principle.

The set of chromosomes representing the search space serves as blueprint for specific *individuals* (phenotypes)



(a) Minimum Distance 10



(b) Minimum Distance 12

Fig. 3 - WC SCHEDULING SCENARIOS $T0 > T1 > T2 > T3$

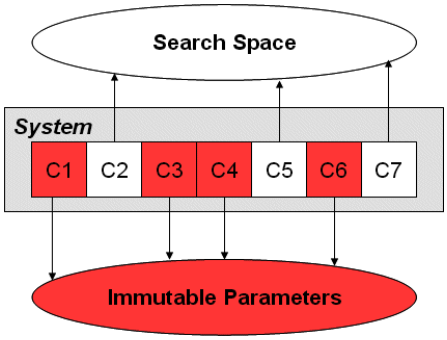


Fig. 4 - SEARCH SPACE DEFINITION

used during exploration. The variation operators (i.e. crossover and mutation) for these individuals are applied chromosome-wise.

There are two reasons why we have chosen independent encoding and variation. First, it is easier to establish a constructively correct encoding on a small subset of design decisions. Such an encoding scheme ensures that all chromosome values correspond to valid decisions such that any chromosome variation is constructively valid. This improves the optimization process as it greatly reduces the effort of checking a generated design for validity. It allows to use the analysis engine of SymTA/S which requires correct design parameters to apply analysis (e.g. sum of time slots no longer than the period, legal priority setting). Secondly, it is easy to add and remove design parameters to the optimization process, even dynamically, which we exploit in our approach.

Chromosomes can be defined arbitrarily fine or coarse grain. This enables the designer to define the search space very precisely. She can limit certain parameters locally while giving others a more global scope. This way of defining the search space represents a compositional approach to optimization and allows to scale the search process. The designer can conduct several well directed exploration steps providing her insight into the system's performance dependencies. Based on this knowledge she can then identify interesting design sub-spaces, worthy to be searched in-depth or even completely. An a priori global exploration does not permit such a flexibility and neglects the structure of the design space, giving the designer no possibility to modify and select the exploration strategy. In the worst-case, when the composition of the design space is unfavorable, this can lead to nonsatisfying results with no possibility for the designer to intervene. In many approaches the only possibility for the designer in such a case consists in restarting the exploration, hoping for better results.

One important precondition for this approach to design space exploration is the on-line configurability of the search space. Our framework allows the designer to redirect the exploration in a new direction without discarding already obtained results. She can for example downsize the search space by fixing parameters having the same values in (nearly) all obtained pareto-optimal solutions, or expand it with parameters not yet considered. Note that this methodology is more flexible than separate local parameter optimization and subsequent recombination.

B. Optimization Objectives

Optimization objectives can be any kind of metric defined on desired or undesired properties of the considered system. Note that some metrics only make sense in combination with constraints. Each individual is associated with a fitness vector containing one entry for every concurrent optimization objective. We use the following notation:

R - maximum response time of a task or
maximum end-to-end latency along a path
D - deadline (task or end-to-end)
ω - constant weight > 0
k - number of tasks or
number of constrained tasks/paths in the system

and define following example optimization objectives available in our framework:

1) minimization of the (weighted) sum of completion times
$$\sum_{i=1}^{k} \omega_i * R_i$$
2) minimization of the maximum lateness
$$\max(R_1 - D_1, \ldots, R_k - D_k)$$
3) maximization of the minimum earliness
$$\min(D_1 - R_1, \ldots, D_k - R_k)$$

4) minimization of the (weighted) average lateness

$$\sum_{i=1}^{k} \omega_i * (R_i - D_i)$$

5) maximization of the (weighted) average earliness

$$\sum_{i=1}^{k} \omega_i * (D_i - R_i)$$

6) minimization of end-to-end latencies
7) minimization of jitters
8) minimization of the sum of communication buffer sizes

The choice of the metric for optimization of a specific system is very important to obtain satisfying results. Example metrics 4 and 5, for instance, express the average timing behavior of a system with regard to its timing constraints. They might mislead an evolutionary algorithm and prevent him from finding system configurations fulfilling all timing constraints, since met deadlines compensate linearly for missed deadlines. For systems with hard real-time constraints, metrics with higher penalties for missed deadline and less rewards for met deadlines can be more appropriate, since they lead to a more likely rejection of system configurations violating hard deadline constraints. Following example metric penalizes violated deadlines in an exponential way and can be used to optimize the timing properties of a system with hard real-time constraints:

$$\sum_{i=0}^{k} c_i^{R_i - D_i},\ c_i > 1\ constant$$

Performing a multi-objective optimization in SymTA/S usually leads to the discovery of several *pareto-optima*. Pareto-optima are best solutions with respect to a particular parameter. More precisely, given a set V of k-dimensional vectors $v \in \mathbb{R}^k$. A vector $v \in V$ dominates a vector $w \in V$ if for all elements $0 \leq i < k$ we have $v_i \leq w_i$ and for at least one element l we have $v_l < w_l$. A vector is called pareto-optimal if it is not dominated by any other vector in V.

Pareto-optimal solutions represent a certain trade-off between two or more objectives, leaving it to the designer to decide which solution to adopt. In our case, individuals with pareto optimal fitness vectors represent the different system design trade-offs.

C. Design Space Exploration Loop

Figure 5 shows the design space exploration loop performed in SymTA/S. The *Optimization Controller* is the central element. It is connected to SymTA/S, which performs the analysis of the individuals, and to an evolutionary multi-objective optimizer. The latter is responsible for the problem-independent part of the optimization problem, i.e. elimination of individuals and selection of interesting individuals for variation. Currently, we use FEMO (Fair Evolutionary Multiobjective Optimizer) [10] and SPEA2 (Strength Pareto Evolutionary Algorithm 2) [26] for this part. Both are coupled via PISA (Platform and Programming Language Independent Interface for Search Algorithms) [1]. Note that the problem-specific part of the optimization problem is coded in the chromosomes and their variation operators. An example for a variation operator is *order crossover* [2]. It is applicable for priority assignments coded as lists, in which each entry corresponds to the priority of a specific task. The offspring inherits the priority assignments of the tasks between two randomly chosen positions in the priority list from the first parent. The remaining priorities are inherited from the second parent, beginning at the first position of its priority list, starting from the second chosen position and skipping over all priorities already assigned in the offspring.

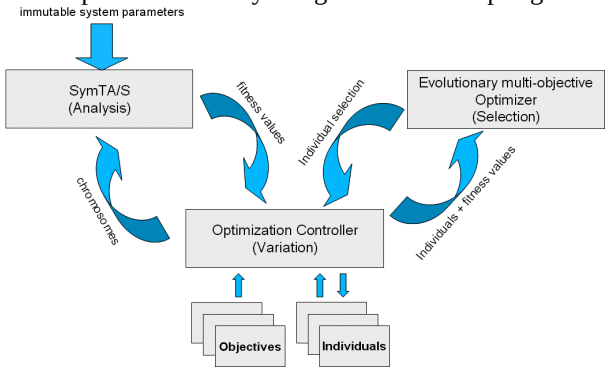


Fig. 5 - DESIGN SPACE EXPLORATION LOOP

Before the exploration loop is started, SymTA/S is initialized with the immutable part of the system architecture. In order to analyze a design alternative represented by an individual, its chromosomes are transformed into commands and applied to SymTA/S. This completes the system design which can then be analyzed by SymTA/S. After analysis the optimization controller requests the system parameters necessary to determine the fitness values according to the optimization objectives. This procedure is performed for every individual currently considered. The individuals and their fitness vectors are then sent to the evolutionary multi-objective optimizer. On the basis of the fitness values the optimizer creates two sets. One set contains individuals selected for elimination, the other contains individuals selected for variation (mutation and crossover). These sets are communicated to the optimization controller, which deletes eliminated individuals and performs the requested mutation and crossover operations. The next iteration is then started with the surviving and newly created individuals.

After each iteration the designer can choose to modify the search space. This consists, like explained in section V-A, in adding/removing chromosomes to/from the individuals. The reevaluation of the fitness values is performed automatically and the next iteration is then started.

Note that the selection of individuals for elimination and variation depends on the used multi-objective optimizer. For instance FEMO [10], eliminates all dominated

individuals in every iteration and pursuits a fair sampling strategy, i.e. each parent participates in the creation of the same number of offsprings. This leads to a uniform search in the neighborhood of elitist individuals.

VI. System on chip example

The system in Fig. 6 represents a SoC consisting of a micro-controller (*uC*), a digital signal processor (*DSP*) and dedicated hardware (*HW*), all connected via an on-chip bus (*BUS*). The *HW* acts as an interface to a physical system. It runs one task (*sys_if*) which issues actuator commands to the physical system and collects routine sensor readings. *sys_if* is controlled by controller task *ctrl*, which evaluates the sensor data and calculates the necessary actuator commands. *ctrl* is activated by a periodic timer (*tmr*) and by the arrival of new sensor data (AND-activation in a cycle).

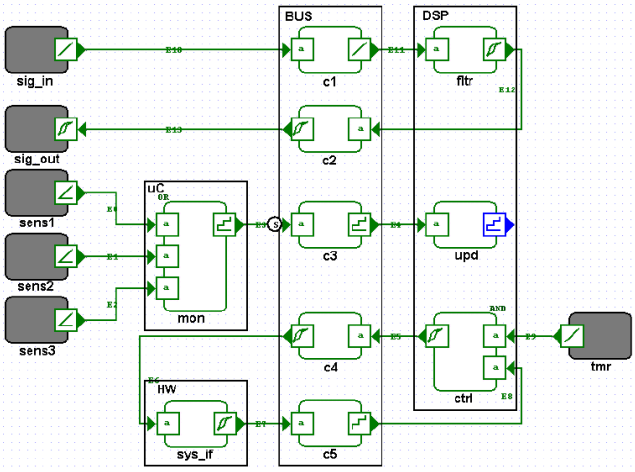


Fig. 6 - System on chip example

The physical system is additionally monitored by 3 smart sensors ($sens_1$ - $sens_3$), which produce data sporadically as a reaction to irregular system events. This data is registered by an OR-activated monitor task (*mon*) on the *uC*, which decides how to update the control algorithm. This information is sent to task *upd* on the *DSP*, which writes the updated controller parameters into shared memory.

The *DSP* additionally executes a signal-processing task (*fltr*), which filters a stream of data arriving at input *sig_in*, and sends the processed data via output *sig_out*. All communication (with the exception of shared-memory on the *DSP*) is carried out by communication tasks *c1* - *c5* over the on-chip *BUS*.

Computation and communication tasks shall have the core execution times listed in table II (i.e. assuming no interrupts). We assume the event models at system inputs specified in table III. In order to function correctly, the system has to satisfy the path latency constraints and the maximum jitter constraint at *sig_out* listed in tables IV(a) and IV(b). In the following we assume that the *DSP* as well as the *BUS* are scheduled according to a static priority preemptive policy.

computation task	core execution time
mon	[10,12]
sys_if	[15,15]
fltr	[12,15]
upd	[5,5]
ctrl	[20,23]
communication task	**core communication time**
c1	[8,8]
c2	[4,4]
c5	[4,4]
c3	[4,4]
c4	[4,4]

TABLE II - Core execution times

input	event model
$sens_1$	sporadic, $\mathcal{P}_{s1} = 1000$
$sens_2$	sporadic, $\mathcal{P}_{s2} = 750$
$sens_3$	sporadic, $\mathcal{P}_{s3} = 600$
sig_in	periodic, $\mathcal{P}_{in} = 60$
tmr	periodic, $\mathcal{P}_{tmr} = 70$

TABLE III - Input event models

constraint #	path	maximum latency
1	$sens_i \rightarrow upd$	70
2	$sig\_in \rightarrow sig\_out$	60
3	cycle (e.g. $ctrl \rightarrow ctrl$)	140

(a) Path latency constraints

constraint #	output	event model jitter
4	*sig_out*	$\mathcal{J}_{out,max} = 22$

(b) Maximum jitter constraint at sig_out

TABLE IV - Constraints

VII. Design space exploration

In this section, we explore the given SoC example. We will do this in several steps, extending the search space gradually. First we will perform a local optimization on the *BUS* altering only the priorities of the communication channels. Afterwards, we will extend the search space by allowing shapers at reasonable positions. During these two steps we will assume the following priority assignment on the *DSP*: $upd > fltr > ctrl$. Finally, we will optimize the system globally, i.e. the priority assignment on the *BUS* and the *DSP* as well as traffic shaping. Optimization objectives are the minimization of the path latencies (constraint 1-3) and the jitter at output *sig_out* (constraint 4).

For this relatively simple architecture an exploration loop of 15 iterations with a population size of 50 individuals found all pareto-optimal solutions in almost every experiment. This exploration takes approximately 20 seconds on a Pentium 4 at 2400 MHz.

A. Optimizing the BUS

The first step in our design space exploration is local optimization of the *BUS*. Although there are only five communication channels on the *BUS*, it is not intuitive for the designer which priority assignments lead to systems that meet all constraints. Local exploration of the *BUS* will give us a first feeling about the systems behaviour, and thus a deeper understanding of its performance dependencies. Table V shows the obtained solutions.

#	*BUS* tasks	*DSP* tasks	con. 1	con. 2	con. 3	con. 4
1	*c2,c1,c3,c4,c5*	*upd,fltr,ctrl*	55	42	120	18
2	*c2,c1,c4,c3,c5*	*upd,fltr,ctrl*	59	42	112	18
3	*c2,c4,c1,c3,c5*	*upd,fltr,ctrl*	59	46	108	22
4	*c1,c2,c4,c5,c3*	*upd,fltr,ctrl*	63	42	96	18
5	*c2,c4,c1,c5,c3*	*upd,fltr,ctrl*	63	46	92	22

TABLE V - PARETO-OPTIMAL SOLUTIONS: LOCAL OPTIMIZATION ON THE BUS

As we can see there are five priority assignments for the communication channels on the *BUS* leading to functioning systems. These solutions are pareto-optimal, which means that they represent a certain trade-off between multiple objectives, leaving it to the designer to decide which solution to adopt.

We observe that channels $c1$ and $c2$ have high priorities in all obtained solutions, whereas channel $c5$ has throughout the lowest or second lowest priority.

B. Traffic shaping

In the second step we want evaluate the optimization potential of selective traffic shaping (see section IV) for the given architecture. We extend our search-space by using shapers at the output of task *mon*. It is making sense to perform traffic shaping at this location, because the OR-activation of *mon* can lead in the worst-case scenario to bursts at its output. That is, if all three sensors trigger at the same time, *mon* will send three packets over the *BUS* with a distance of 10 time units, which is its minimum core execution time. This transient load peak affects the overall system performance in a negative way. A shaper is able to increase this minimum distance in order to weaken the global impact of the worst-case burst. Exploration over the minimum distance of successive packets enforced by the inserted shaper is subject of this exploration step.

We observed in the previous experiment that communication channel $c5$ was always assigned the lowest or second lowest priority. Even in the lowest case, the cycle constraint (constraint 3) was easily met. Therefore, we will fix channel $c5$ to the lowest priority on the bus. This narrows the search space considerably, the number of possible priority assignments on the bus is reduced from $5! = 120$ to $4! = 24$.

Table VI shows the additional pareto-optimal solutions found using shapers at the output of *mon* extending the minimum distance of successive events to between 11 and 20 time units. Solutions which are dominated by the results obtained in the previous section are not listed.

#	*BUS* tasks	*DSP* tasks	δ^-	con. 1	con. 2	con. 3	con. 4
6	*c3,c2,c1,c4,c5*	*upd,fltr,ctrl*	13	51	45	120	21
7	*c3,c2,c1,c4,c5*	*upd,fltr,ctrl*	14	53	45	116	21
8	*c3,c2,c1,c4,c5*	*upd,fltr,ctrl*	16	57	45	112	21
9	*c2,c3,c1,c4,c5*	*upd,fltr,ctrl*	17	63	41	112	17
10	*c3,c2,c1,c4,c5*	*upd,fltr,ctrl*	20	65	40	104	16

TABLE VI - ADDITIONAL PARETO-OPTIMAL SOLUTIONS: OPTIMIZING BUS AND TRAFFIC SHAPING AT MON OUTPUT

We observe that performing traffic shaping at the output of *mon* leads to several new interesting solutions. We found new priority assignments on the *BUS* which, combined with a certain shaper, result in better values for constraints 1,2 and 4. Solely the previously obtained values for constraint 3 are not reached, but the constraint remains fulfilled by a large margin.

Only two different priority assignments, $c2 > c3 > c1 > c4 > c5$ and $c3 > c2 > c1 > c4 > c5$, occur in the solutions listed in table VI. Let us take a closer look on the global impact of traffic shaping at the output of *mon*. Tables VII(a) and VII(b) show the performance of the system with growing minimum distance of events at the output of *mon* for these two priority assignments. Rows containing pareto optimal solutions are emphasized. Note that in this example a shaper extending the minimum distance to a value between 11 and 20 time units needs to store at most one packet at a time. To achieve larger minimum distances, two packets need to be stored in the worst-case.

We see that the value for constraint 1 is climbing with growing minimum distance. This is not surprising because the inserted shaper is creating additional latency in the worst-case, depending on the desired minimum distance. The longest minimum distance that does not lead to violation of constraint 1 for the priority assignments $c2 > c3 > c1 > c4 > c5$ and $c3 > c2 > c1 > c4 > c5$ are 20 and 22 respectively.

While the shaper leads to increased values for constraint 1, the rest of the system is profiting from the weakend burst. Figures 7(a) and 7(b) give a concise graphical overview of the system behaviour with growing minimum distance.

C. Including the DSP

So far, we obtained ten solutions representing different trade-offs for our example SoC by using local exploration techniques. Now, we extend our search space by the priority assignment on the *DSP*. Since we already observed that the cycle constraint is uncritical, we will fix the priorities of communication channels $c4$ and $c5$ to the second lowest and lowest on the *BUS* respectively. Additionally, we fix the priority of task *ctrl* to the lowest on the *DSP*.

In table VIII we see the new system configurations found. Solutions which are dominated by the results obtained in the previous sections are not listed.

The obtained solutions represent new interesting trade-offs because they lead to a low jitter at *sig_out* (con. 4).

δ^-	con. 1	con. 2	con. 3	con. 4
10	49	50	162	26
11	51	50	162	26
12	53	46	120	22
13	55	46	120	22
14	57	46	116	22
15	59	46	116	22
16	61	46	112	22
17	**63**	**41**	**112**	**17**
18	65	41	112	17
19	67	41	112	17
20	69	41	104	17

(a) $c2 > c3 > c1 > c4 > c5$

δ^-	con. 1	con. 2	con. 3	con. 4
10	45	54	162	30
11	47	54	162	30
12	49	50	120	26
13	**51**	**45**	**120**	**21**
14	**53**	**45**	**116**	**21**
15	55	45	116	21
16	**57**	**45**	**112**	**21**
17	59	45	112	21
18	61	45	112	21
19	63	45	112	21
20	**65**	**40**	**104**	**16**
21	67	40	104	16
22	69	40	104	16

(b) $c3 > c2 > c1 > c4 > c5$

TABLE VII - SYSTEM PERFORMANCE WITH TRAFFIC SHAPING AT MON OUTPUT.

#	*BUS* tasks	*DSP* tasks	δ^-	con. 1	con. 2	con. 3	con. 4
11	*c2,c1,c3,c4,c5*	*fltr,upd,ctrl*	10	70	27	120	3
12	*c3,c2,c1,c4,c5*	*fltr,upd,ctrl*	12	64	35	120	11
13	*c2,c3,c1,c4,c5*	*fltr,upd,ctrl*	12	68	31	120	7
14	*c3,c2,c1,c4,c5*	*fltr,upd,ctrl*	14	68	35	116	11

TABLE VIII - ADDITIONAL PARETO-OPTIMAL SOLUTIONS: GLOBAL OPTIMIZATION

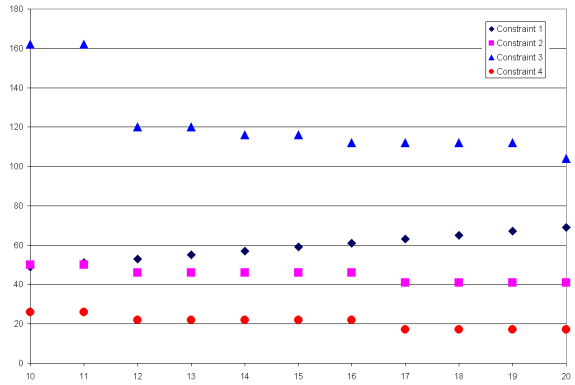
This quality did not exist in any of the previously obtained system configurations. However, the low jitter at *sig_out* is bought with high values for constraint 1.

D. Summary of results

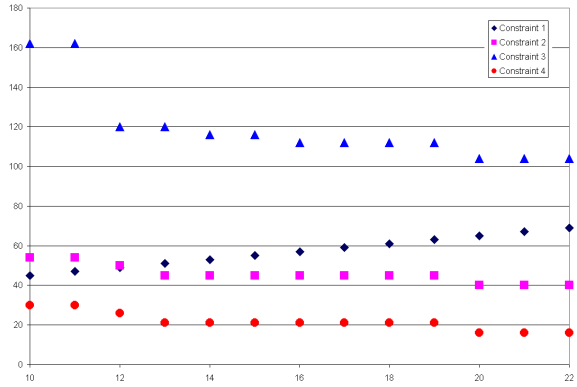
Table IX gives an overview about all pareto-optimal system configurations from which a designer can choose. For example an attractive solution might be one where all constraints are fulfilled with a healthy margin to the respective maximum values. This is the case for solutions 4, 10, and 12 for instance. Figure 8 gives a graphical overview of all solutions.

VIII. CONCLUSION

In this paper we presented a framework for flexible design space exploration and system optimization for heterogeneous SoC and distributed systems using SymTA/S and evolutionary optimization techniques. The ambition



(a) $c2 > c3 > c1 > c4 > c5$



(b) $c3 > c2 > c1 > c4 > c5$

Fig. 7 - SYSTEM PERFORMANCE WITH TRAFFIC SHAPING AT MON OUTPUT: GRAPHICAL OVERVIEW

#	*BUS* tasks	*DSP* tasks	δ^-	con. 1	con. 2	con. 3	con. 4
1	*c2,c1,c3,c4,c5*	*upd,fltr,ctrl*	10	55	42	120	18
2	*c2,c1,c4,c3,c5*	*upd,fltr,ctrl*	10	59	42	112	18
3	*c2,c4,c1,c3,c5*	*upd,fltr,ctrl*	10	59	46	108	22
4	*c1,c2,c4,c5,c3*	*upd,fltr,ctrl*	10	63	42	96	18
5	*c2,c4,c1,c5,c3*	*upd,fltr,ctrl*	10	63	46	92	22
6	*c3,c2,c1,c4,c5*	*upd,fltr,ctrl*	13	51	45	120	21
7	*c3,c2,c1,c4,c5*	*upd,fltr,ctrl*	14	53	45	116	21
8	*c3,c2,c1,c4,c5*	*upd,fltr,ctrl*	16	57	45	112	21
9	*c2,c3,c1,c4,c5*	*upd,fltr,ctrl*	17	63	41	112	17
10	*c3,c2,c1,c4,c5*	*upd,fltr,ctrl*	20	65	40	104	16
11	*c2,c1,c3,c4,c5*	*fltr,upd,ctrl*	10	70	27	120	3
12	*c3,c2,c1,c4,c5*	*fltr,upd,ctrl*	12	64	35	120	11
13	*c2,c3,c1,c4,c5*	*fltr,upd,ctrl*	12	68	31	120	7
14	*c3,c2,c1,c4,c5*	*fltr,upd,ctrl*	14	68	35	116	11

TABLE IX - ALL PARETO-OPTIMA

of our framework is not to perform a global black-box optimization, but to give the designer control about the exploration process. To ensure that, our framework allows a very precise definition of the search space. This enables the designer to perform multiple exploration steps, adjusting the search space as his understanding of the systems performance dependencies grows, in order to

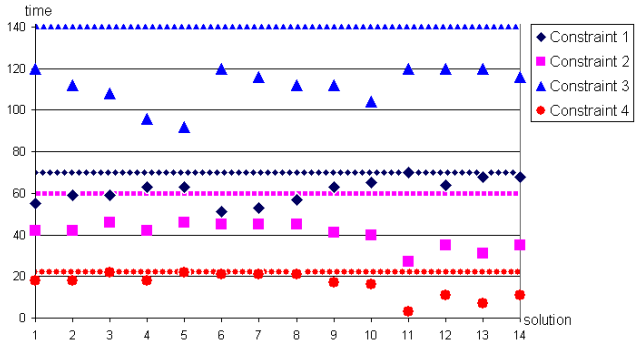


Fig. 8 - ALL PARETO-OPTIMA: GRAPHICAL OVERVIEW

identify interesting design sub-spaces containing good solutions. Thereby, arbitrary system properties can be subject to optimization. We demonstrated a possible exploration process in the given SoC example, extending the search space with new system parameters while restricting others in every iteration.

A central aspect in our approach is traffic shaping. The optimization potential through traffic shaping in complex heterogeneous SoC and distributed systems is very high. We saw in the described SoC example that inserting a shaper in order to weaken a transient load peak considerably improves system behavior, and consequently leads to the discovery of interesting design alternatives.

ACKNOWLEDGEMENT

We would like to thank Lothar Thiele and his group from the ETH Zürich for providing us with the PISA interface and the corresponding algorithms for multi-objective optimization.

REFERENCES

[1] Stefan Bleuler, Marco Laumanns, Lothar Thiele, and Eckart Zitzler. PISA — a platform and programming language independent interface for search algorithms. *http://www.tik.ee.ethz.ch/pisa/*.

[2] L. Davis. Applying adaptive algorithms to epistatic domains. In *Proc. of the 9th IJCAI*, pages 162–164, Los Angeles, CA, 1985.

[3] K. Deb. *Multi-objective optimization using evolutionary algorithms.* John Wiley, Chichester, 2001.

[4] J.J.G. Garcia and M.G. Harbour. Optimized priority assignment for tasks and messages in distributed real-time systems. In *Proc. Workshop on Parallel and Distributed Real-Time Systems*, 1995.

[5] T. Givargis and F. Vahid. Platune: A tuning framework for system-on-a-chip platforms. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, v21, n11, pp 1317-1327*, 2002.

[6] Arne Hamann, Rafik Henia, Marek Jersak, Razvan Racu, Kai Richter, and Rolf Ernst. SymTA/S - Symbolic Timing Analysis for Systems. *http://www.symta.org/*.

[7] M. Jersak and R. Ernst. Enabling scheduling analysis of heterogeneous systems with multi-rate data dependencies and rate intervals. In *Proc. 40th Design Automation Conference*, Annaheim, USA, June 2003.

[8] M. Jersak, R. Henia, and R. Ernst. Context-aware performance analysis for efficient embedded system design. In *Proc. of Design, Automation and Test in Europe (DATE'04)*, Paris, France, March 2004.

[9] M. Jersak, K. Richter, and R. Ernst. Performance analysis for complex embedded applications. *International Journal of Embedded Systems, Special Issue on Codesign for SoC*, 2004.

[10] M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, and K. Deb. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. *In Parallel Problem Solving From Nature — PPSN VII*, 2002.

[11] A. Maxiaguine, S. Künzli, S. Chakraborty, and L. Thiele. Rate analysis for streaming applications with on-chip buffer constraints. In *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 131–136, Yokohama, Japan, January 2004.

[12] A.K. Mok and D. Chen. A multiframe model for real-time tasks. *IEEE Transactions on Software Engineering*, 23(10):635–645, 1997.

[13] J. C. Palencia and M. G. Harbour. Schedulablilty analysis for tasks with static and dynamic offsets. In *Proc. 19th IEEE Real-Time Systems Symposium (RTSS'98)*, Madrid, Spain, 1998.

[14] P. Pop, P. Eles, and Z. Peng. Holistic scheduling and analysis of mixed time/event-triggered distributed embedded systems. In *Tenth International Symposium on Hardware/Software Codesign (CODES'02)*, Estes Park, Colorado, USA, May 2002.

[15] Paul Pop, Petru Eles, Zebo Peng, Viacheslav Izosimov, Magnus Hellring, and Olof Bridal. Design optimization of multi-cluster embedded systems for real-time applications. In *Proc. of Design, Automation and Test in Europe (DATE'04)*, Paris, France, March 2004.

[16] K. Richter and R. Ernst. Event model interfaces for heterogeneous system analysis. In *Proc. of Design, Automation and Test in Europe (DATE'02)*, Paris, France, March 2002.

[17] K. Richter, R. Racu, and R. Ernst. Scheduling analysis integration for heterogeneous multiprocessor SoC. In *Proc. 24th International Real-Time Systems Symposium (RTSS'03)*, Cancun, Mexico, December 2003.

[18] K. Richter, D. Ziegenbein, M. Jersak, and R. Ernst. Model composition for scheduling analysis in platform design. In *Proc. 39th Design Automation Conference*, New Orleans, USA, June 2002.

[19] Kai Richter. On the characterization of communication traffic and task load models in performance verification and architecture evaluation. Technical Report TR-SPI-04-01, Institut für Datentechnik und Kommunikationsnetze, Technische Universität Braunschweig, 2004.

[20] G. Snider. Automated design space exploration for embedded computer systems. Technical Report HPL-2001-220, Hewlett-Packard Laboratories, 2001.

[21] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli. A framework for evaluating design tradeoffs in packet processing architectures. In *Proc. 39th Design Automation Conference (DAC)*, pages 880–885, New Orleans, USA, 2002. ACM Press.

[22] L. Thiele, S. Chakraborty, M. Gries, A. Maxiaguine, and J. Greutert. Embedded software in network processors - models and algorithms. In *Proc. 1st Workshop on Embedded Software (EMSOFT)*, Lake Tahoe (CA), USA, October 2001.

[23] Lothar Thiele, Samarjit Chakraborty, and Martin Naedele. Real-time calculus for scheduling hard real-time systems. In *Proceedings International Symposium on Circuits and Systems (ISCAS)*, Geneva, Switzerland, 2000.

[24] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing & Microprogramming*, 50(2-3):117–134, April 1994.

[25] K. W. Tindell. Adding time-offsets to schedulability analysis. Technical Report YCS 221, Univ. of York, 1994.

[26] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.