SymTA/S - Symbolic Timing Analysis for Systems

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

ABSTRACT

SymTA/S is a performance and timing analysis tool based on formal scheduling analysis techniques and symbolic simulation. The tool supports heterogeneous architectures, complex task dependencies, context aware analysis, and combines optimization algorithms with system sensitivity analysis for rapid design space exploration. This paper gives an overview of the current and future research interests in the SymTA/S project.

I. INTRODUCTION

Although there are countless approaches for formal performance and timing analysis known from real-time system research, only very few have been adopted in design of heterogeneous SoCs and distributed systems. The SymTA/S approach enables a completely new view on system level analysis and supports explicitly the combination and integration of heterogeneous subsystems.

In the first part of the paper we will give a brief overview about the formal core of SymTA/S. Afterwards, we will shortly introduce current research interests in the SymTA/S project. These are context aware analysis, optimization, and sensitivity analysis. We conclude the paper with an example demonstrating the relevance of these aspects.

II. THE SYMTA/S APPROACH

SymTA/S [1] 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 [9], [11]. 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 all known 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 [9].

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. More precisely, they provide control about the timing of exchanged events and data. Consequently, they enable the user to model buffering and perform traffic shaping. This is important because buffering and traffic shaping break up non-functional dependency cycles and can tremendously reduce transient load peaks in dynamic systems [10]. 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 [12], [8] cannot handle.

In order to perform a system level analysis, SymTA/S locally performs existing scheduling analysis (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 discribed basic SymTA/S approach has been recently extended to support multi-rate systems, multiple inputs and functional cycles [2], [4]. These major extensions enable SymTA/S to cope with complex applications. Furthermore, SymTA/S is able to consider inter and intra context information to tighten analysis bounds [3].

III. SYSTEM CONTEXTS

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 [13] and generalized by Palencia and Harbour [7], 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 [6], 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. Our contribution lies in 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

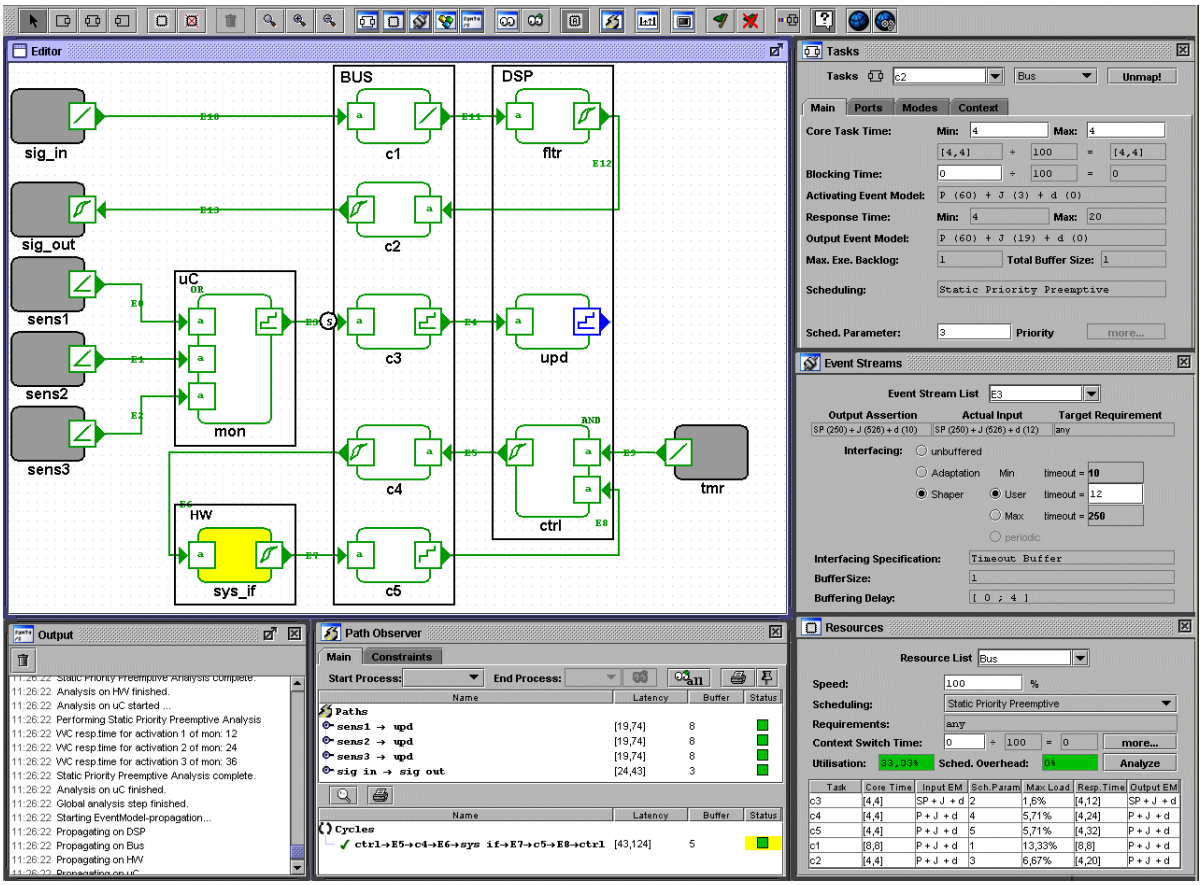

Fig. 1 - SymTA/S with system on chip example

on the other [3]. 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. Optimization

One strong point of performance analysis is that even complex systems can be analyzed in very short time. This fact provides the possibility to perform architecture exploration. Exploration is needed, since manual optimization is very time consuming for distributed systems due to the multitude of complex hard to track performance dependencies between tasks.

In SymTA/S we experiment with evolutionary algorithms to optimize distributed systems. Thereby, the search space and the optimization objectives can be multi-dimensional. Search parameters include mapping of tasks onto different resources, changing priorities on priority-scheduled resources, time slot sizes on TDMA or round robin scheduled resources, and modifying resource speed. Since shapers in SymTA/S allow to control the timing of events and data between connected components (see section II), additional optimization is possible due to systematic traffic shaping.

Performing a multi-objective optimization inevitably leads to the discovery of several pareto optima. More precisely, each solution represents a certain trade-off between two or more objectives, leaving it to the designer to decide which solution to adopt.

V. Sensitivity Analysis

The analysis techniques known from literature calculate the timing behavior of a specific system considering a predefined set of input parameters (core execution times of tasks, activation periods, input jitters, etc). Such solutions are sufficient for the verification of the performance of a given system.

However, in a realistic system design process it is important to understand the effects of small parameter variations on system performance, as such variations are inevitable during implementation and integration. Capturing the bounds within which a parameter can be varied without violating constraints offers more flexibility for the system designer and supports future changes.

Different system parameters can be used as basis for the sensitivity analysis [5], [14]. An example is an exact characterization of the variation bounds of core execution times or activation jitters, such that timing constraints are always satisfied. Another example is the dependency between input

event model parameters and the buffers required to capture activation back-logs.

VI. System on chip example

The system in Fig. 1 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).

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 I (i.e. assuming no interrupts). We assume the event models at system inputs specified in table II. 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 III and IV.

comp. task	core exe. time	comm. task	core exe. time
mon	[10,12]	*c1*	[8,8]
sys_if	[15,15]	*c2*	[4,4]
fltr	[12,15]	*c5*	[4,4]
upd	[5,5]	*c3*	[4,4]
ctrl	[20,23]	*c4*	[4,4]

TABLE I - 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 II - Input event models

A. Analysis

We will use static priority scheduling both on the *DSP* and the *Bus*. The priorities on the *Bus* respectively *DSP* are assigned as follows: $c1 > c2 > c3 > c4 > c5$ and $fltr > upd > ctrl$.

constraint #	path	maximum latency
1	$sens_i \rightarrow upd$	70
2	$sig\_in \rightarrow sig\_out$	120
3	cycle (e.g. $ctrl \rightarrow ctrl$)	140

TABLE III - Path latency constraints

constraint #	output	event model jitter
4	*sig_out*	$\mathcal{J}_{out,max} = 18$

TABLE IV - Maximum jitter constraint

Table V shows the calculated response times of the computation and communication tasks with and without taking into account inter event stream contexts. We observe that the exploitation of context information leads to much tighter response time intervals in the given example. This in turn reduces the calculated worst-case values for the constrained parameters. Table VI shows that, in contrast to the inter context blind analysis, all system constraints are satisfied when performance analysis takes inter event stream context information into account. In other words, a context blind analysis would have discarded a solution which is in reality valid.

comp task	$Resp_{blind}$	$Resp_{sens}$	comm. tasks	$Resp_{blind}$	$Resp_{sens}$
mon	[10,36]	[10,36]	*c1*	[8,8]	[8,8]
sys_if	[15,17]	[15,15]	*c2*	[4,12]	[4,4]
fltr	[12,15]	[12,15]	*c3*	[4,16]	[8,12]
upd	[5,22]	[5,22]	*c4*	[4,28]	[8,20]
ctrl	[20,53]	[20,53]	*c5*	[4,32]	[8,32]

TABLE V - context blind and sensitive analysis

constraint #	inter context-blind	inter context-sensitiv
1	74	70
2	35	27
3	130	120
4	11	3

TABLE VI - constraints context blind and sensitive

B. Optimizations

Let us now try to optimize our example architecture. Optimization objectives are the four defined constraints. We try to minimize the latencies on paths 1-3 and the jitter at output *sig_out*.

In the first experiment our search space consists of the priority assignments on the *BUS* and the *DSP*. Table VII shows the existing pareto optimal solutions. In the first two columns, tasks are ordered by priority, highest priority on the left. In the last four columns, we give the actual value for all four constrained values. The best reached values for each constraint are emphasized.

As we can observe there are several possible solutions, each with its own advantages and disadvantages. We also observe

#	*Bus* tasks	*DSP* tasks	con. 1	con. 2	con. 3	con. 4
1	*c1, c2, c3, c4, c5*	*upd, fltr, ctrl*	**55**	42	120	18
2	*c1, c2, c4, c3, c5*	*upd, fltr, ctrl*	59	42	112	18
3	*c2, c1, c4, c5, c3*	*upd, fltr, ctrl*	63	42	**96**	18
4	*c1, c2, c3, c4, c5*	*fltr, upd, ctrl*	70	**27**	120	**3**

TABLE VII - PARETO OPTIMAL SOLUTIONS

#	*Bus* tasks	*DSP* tasks	con. 1	con. 2	con. 3	con. 4
1	*c2, c1, c3, c4, c5*	*upd, fltr, crtl*	**59**	42	120	18
2	*c1, c2, c4, c3, c5*	*upd, fltr, ctrl*	63	42	112	18
3	*c3, c2, c1, c4, c5*	*fltr, upd, ctrl*	64	35	120	11
4	*c2, c1, c5, c4, c3*	*upd, fltr, ctrl*	67	42	**96**	18
5	*c2, c3, c1, c5, c4*	*fltr, upd, ctrl*	68	**31**	134	**7**

TABLE VIII - PARETO OPTIMAL SOLUTIONS: SHAPER AT MON OUTPUT

that in each solution one constraint is only barely satisfied. A designer might want to find some alternative solutions where all constraints are fulfilled with a larger margin to the respective maximum values.

We extend our search space by using a shaper at the output of task *mon*. This is a good place to perform traffic shaping, because the OR-activation of *mon* can lead in the worst-case scenario to bursts at its output. More precisely, 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.

Table VIII shows pareto optimal solutions using a shaper at the output of *mon* extending the minimum distance of events at the output of *mon* to 12 time units, and thus weakening the global impact of the worst-case burst. The required buffer for this shaper is minimal, because at most one packet needs to be buffered at any time.

We observe that several new solutions are found. Not all best values for each constraint from the first attempt are reached, yet configurations 3 and 5 are interesting since they are more balanced regarding the constraints.

C. Sensitivity analysis

We applied sensitivity analysis to the pareto optimal system configurations obtained in Section VI-B. The Δ values show the variation limits of core execution times when varying only one task at a time. Tables IX and X present the values obtained for the system configurations described in tables VII and VIII, respectively. The emphasized values indicate the maximum variation limits obtained.

#	$\Delta c1$	$\Delta c2$	$\Delta c3$	$\Delta c4$	$\Delta c5$	Δupd	$\Delta fltr$	$\Delta ctrl$	$\Delta sysif$	Δmon
1	0	0	1.11	3.33	10	0	0	**7**	13	**5**
2	0	0	**3.66**	**6**	**18**	0	0	**7**	**21**	3.66
3	0	0	2.33	2.5	2.5	0	0	**7**	9	2.33
4	0	0	0	3.33	13.5	0	0	**7**	13	0

TABLE IX - SENSITIVITY ANALYIS: CORE TASK TIMES

#	$\Delta c1$	$\Delta c2$	$\Delta c3$	$\Delta c4$	$\Delta c5$	Δupd	$\Delta fltr$	$\Delta ctrl$	$\Delta sysif$	Δmon
1	0	0	1.11	3.33	10	0	0	**7**	13	**3.66**
2	0	0	**3.66**	4	**18**	0	0	**7**	**21**	2.33
3	0	**4**	0	3.33	13.5	**1**	**3**	3	13	2
4	0	0	0	0	0	0	0	5	5	1
5	0	2	0	**5**	0	**1**	2	3	3	0.66

TABLE X - SENSITIVITY ANALYIS: CORE TASK TIMES (WITH SHAPER)

VII. CONCLUSION

In this paper we gave a brief overview about the SymTA/S approach. We shortly reviewed the underlying coupling technique using event streams, enabling SymTA/S to combine different kinds of analysis techniques known from real-time research. Afterwards we introduced a SoC example containing multiple inputs with AND- as well as OR-activation and functional cycles. We analyzed the system and saw that the consideration of system contexts can considerably tighten analysis bounds. We then explored the optimization potential. We found several pareto optimal solutions for different optimization objectives. In the last part we analyzed sensitivity of the pareto optimal solutions. We characterized their robustness, and thus their flexibility for later system changes.

REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

[5] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proc. Real-Time Systems Symposiom*, pages 166–171, IEEE Computer Society Press, 1989.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

[14] Steve Vestal. Fixed-priority sensitivity analysis for linear compute time models. *IEEE Transactions on Software Engineering*, 20(4), April 1994.