Model Checking of Scenario-Aware Dataflow with CADP

Bart Theelen[1], Joost-Pieter Katoen[2] and Hao Wu[2]
[1]Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, Netherlands
[2]RWTH Aachen University, Ahornstraße 55, 52056 Aachen, Germany
bart.theelen@esi.nl, katoen@cs.rwth-aachen.de, hao.wu@cs.rwth-aachen.de

Abstract

Various dataflow formalisms have been used for capturing the potential parallelism in streaming applications to realise distributed (multi-core) implementations as well as for analysing key properties like absence of deadlock, throughput and buffer occupancies. The recently introduced formalism of Scenario-Aware Dataflow (SADF) advances these abilities by appropriately capturing the dynamism in modern streaming applications like MPEG-4 video decoding. This paper reports on the application of Interactive Markov Chains (IMC) to capture SADF and to formally verify functional and performance properties. To this end, we propose a compositional IMC semantics for SADF based on which the Construction and Analysis of Distributed Processes (CADP) tool suite enables model checking various properties. Encountered challenges included dealing with probabilistic choice and potentially unbounded buffers, both of which are not natively supported, as well as a fundamental difference in the underlying time models of SADF and IMC. Application of our approach to an MPEG-4 decoder revealed state space reduction factors up to about 21 but also some limitations in terms of scalability and the performance properties that could be analysed.

I. Introduction

Model-based design of distributed (multi-core) realisations of streaming applications like MPEG-4 and MP3 decoding often use dataflow-based analysis and synthesis techniques [3], [16], [22], [23], [26], [32]. Many such approaches (e.g., [3], [22], [23]) rely on the formalism of (Homogeneous) Synchronous Dataflow ((H)SDF) [21], [29], which is also known as (Weighted) Marked Graphs in Petri Net theory. A wide range of design-time analysis techniques have been developed for SDF, covering both functional properties such as absence of deadlock [12] and the key performance properties of throughput [38], [13], latency [14] and buffer occupancy [31]. However, SDF only allows for expressing a fixed behavioural pattern. Using SDF therefore requires abstracting from any dynamism in applications such as decoding different types of video frames in MPEG-4, which imply large variations in resource usage. Neglecting such resource usage variations during design-time analysis may lead to substantially over-dimensioning [32]. Other model-based design approaches (e.g., [16], [26]) use Kahn Process Networks [20] to capture dynamism. However, relevant properties, such as the minimal storage space needed to avoid deadlock, throughput and latency, cannot be determined at design-time [25], [10]. The formalism of Scenario-Aware Dataflow (SADF) [35] has recently been introduced as an alternative that does allow modelling dynamic behavior, while various design-time analysis techniques are available and efficient implementations can be created [33].

Current state-of-the-art techniques for SADF have been implemented in the SDF^3 tool suite [30], [34]. It supports efficient techniques for worst-case performance analysis of a relevant subset of SADF based on a max-plus algebraic semantics [11] as well as specialised exact model-checking based techniques for any SADF model to evaluate predefined performance properties like throughput, expected response time and maximum buffer occupancy [37], [34]. The latter type of analysis does however not use existing (quantitative) model checking tools, thereby lacking flexible support for evaluating user-defined properties (i.e., properties other than certain predefined ones). This paper reports on an investigation to apply the Construction and Analysis of Distributed Processes (CADP) tool suite [9] for quantitative analysis and qualitative model checking of SADF. This enables the usage of powerful (compositional) state-space reduction techniques as well as checking a large range of functional properties expressed in the temporal logic XTL. To enable using CADP, we introduce a compositional semantics for SADF based on Interactive Markov Chains (IMC) [17], [18], thereby formalising previously noticed similarities between two actor types of SADF as well as revealing a fundamental difficulty in capturing the time model of SADF. We propose solutions to handle probabilistic choice and potentially unbounded buffers in CADP. Application of CADP's model checking capabilities to an MPEG-4 decoder [35] shows that considering dynamism easily yields a state-space explosion, while also substantial state-space reductions can be achieved. Although the variety of analysable properties is enlarged, determining throughput and buffer occupancy suffers from

978-3-9810801-8-6/DATE12/

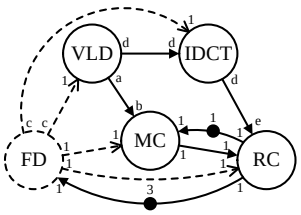
the inability to evaluate reward-based properties in CADP.

This paper is organised as follows. After giving a brief introduction to SADF and IMC, Section IV informally presents our IMC semantics for SADF. Section V discusses the use of CADP, focussing on model checking the MPEG-4 decoder example. Section VI lists the lessons learned.

II. Scenario-Aware Dataflow

Fig. 1 depicts the SADF model of an MPEG-4 decoder for the Simple Profile taken from [35]. The nodes, called *actors* here, represent individual tasks performed in MPEG-4 decoding like Variable Length Decoding (VLD) and Reconstruction (RC) of the final video picture. The edges are called *channels* and denote (potential) dependencies between the actors. Such dependencies arise from communicating *tokens*, which are abstract representations of e.g. frames or macro blocks. The number of tokens that actors consume or produce from/to a channel each time they *fire* (execute) is indicated by the *rate* labels at the head and tail respectively. Fig. 1 shows that such rates may be parameterised as opposed to SDF, where all rates are fixed. SADF separates the data processing behaviour from control behaviour causing (course-grain) dynamism by distinguishing two types of actors and channels. *Kernels* (solid nodes) model the data processing part, while *detectors* (dashed nodes) reflect control behaviour. The Frame Detector (FD), for instance, captures that part of the MPEG-4 decoder responsible for determining which type of frame is to be processed by the other actors. The different frame types imply different operation modes or *scenarios*. The FD notifies other actors about the frame type by communicating scenario-valued tokens over *control* channels (dashed edges). Tokens communicated over *data* channels (solid edges) are unvalued. Any channel represents a FIFO buffer which may include initial tokens at the start of executing the application. Such cases are indicated in Fig. 1 as a black dot on the edge labeled with the number of initial tokens.

The MPEG-4 decoder model supports I-frames and P-frames. When an I-frame is detected, a total of 99 macro blocks must always be processed. This scenario is called I_{99}. A P-frame requires processing between 0 and 99 macro blocks. The workload varies considerably depending on the number of macro blocks that are processed. This is because the VLD and IDCT actors are performed for every individual macro block, while the FD, MC and RC actors fire once per frame. We model this by defining a number of different scenarios P_x based on the number of macro blocks that must be processed. Hence, the model contains different scenarios in which (up to) 0, 30, 40, 50, 60, 70, 80, or 99 macro blocks are processed for a single P-frame. Next to expressing course-grain workload variations via scenarios, SADF can capture fine-grain variations in



Rate	Scenario		
	I_{99}	P_0	P_x
a	0	0	1
b	0	0	x
c	99	1	x
d	1	0	1
e	99	0	x

$x \in \{30, 40, 50, 60, 70, 80, 99\}$

Fig. 1. SADF model of an MPEG-4 decoder

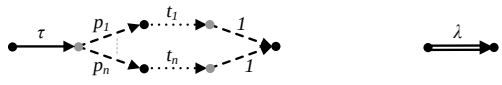
the workload within scenarios by associating a *discrete execution time distribution* to each scenario of any actor.

As indicated, the FD is responsible for determining the scenario in which the other actors operate. SADF abstracts away from the actual data-dependent control conditions on which the scenario is determined in real-life by associating Markov chains to detectors. In Fig. 1, FD contains a fully connected 9-state Markov chain (one state per scenario), where the transition probabilities are chosen such that the probability of visiting each of the 9 states matches with the occurrence probability of the represented scenario.

Designing distributed streaming systems is usually subject to satisfying/optimising various functional and performance properties. A prominent performance criterium is often *throughput*, which for the MPEG-4 decoder is the number of frames that are completely processed per second (i.e., the number of firing completions of actor RC per time unit). Throughput for SDF is known to be maximised in case of so-called *self-timed execution* [29], which refers to a class of scheduling policies where actors fire as soon as they are enabled (i.e., when sufficient tokens have become available on their inputs). Self-timed execution has been adopted for the SADF analysis techniques implemented in SDF^3 [11], [37]. A prerequisite for achieving a positive throughput is *absence of deadlock*, which requires sufficient storage space to be available for the buffers incorporated in channels (amongst others). Analysis of *buffer occupancies* to identify some (minimal) storage space distribution over the buffers (various such distributions can exist [31]) is therefore another important design aspect. Finally, various *latencies* or variations in such latencies must often be minimised. Example latency metrics for the MPEG-4 decoder are the minimum, expected and maximum time until processing the first frame (first firing of actor RC) is completed.

III. Interactive Markov Chains

Contemporary model checkers like CADP allow user-defined (quantitative and qualitative) properties to be evaluated. Such properties include the aforementioned performance criteria for dataflow models as well as functional correctness checks such as absence of deadlocks and livelocks. We enabled the use of CADP by capturing the behaviour of SADF using Interactive Markov



Discrete Time Distribution in SADF Stochastic Time Distribution in IMC

Fig. 2. Difference in time models

Chains (IMC) [17]. IMCs extend traditional continuous-time Markov chains with the ability to interact using *interactive actions*. Hence, an IMC is a state-based model with *action* and *Markovian transitions*. The Markovian transitions model advancing time according to a negative exponential distribution whereas actions can be used to synchronise with other IMCs in a handshake manner like in CSP [19]. By parallel composition of IMCs, a complete system can be described in a compositional way [17], [18]. IMCs have shown their practical relevance in applications of various domains, ranging from dynamic fault trees [4], standardised engineering languages such as AADL (Architectural Analysis and Design Language) [5] and Statemate [1] to Globally Asynchronous Locally Synchronous (GALS) hardware designs [7]. We exploit IMCs to define the semantics of SADF in a *component-based* manner, i.e., each individual actor yields one (or more) IMC(s) which are subsequently put in parallel.

Capturing the behaviour of an SADF model such as the MPEG-4 example in Fig. 1 with IMC is not trivial due to a fundamental difference in their underlying time models. The original semantics of SADF is defined in terms of so-called *Timed Probabilistic Systems* [36]. They contain *action*, *probabilistic* and deterministic *time* transitions. The time transitions denote that time must have advanced with the specified *exact* amount of time units before the next transitions become enabled. A specific combination of transitions allows expressing any discrete execution time distribution associated to the scenario of an actor, see Fig. 2. The left-hand side shows that after performing some (internal) action τ (think of drawing a sample from a discrete distribution), time advances exactly t_i time units with probability p_i for $i = 1, \ldots, n$. Notice that such a distribution has clear (lower and upper) bounds. Opposed to the approach of SADF, the stochastic time model of IMC on the right-hand side of Fig. 2 uses a single Markovian transition to indicate a specific exponential distribution where time advances *on average* with $1/\lambda$ time units. In such case, there exist no exact bounds, thereby rendering analysis of properties depending on such bounds infeasible. On the other hand, the stochastic nature of the time delays in IMCs enable to determine measures like maximal expected time objectives and time-bounded reachability.

To apply CADP to SADF and overcome the difference in time models, we choose to use a single exponential distribution per scenario of an actor such that the mean execution time $1/\lambda$ equals the mean of the original discrete distribution $\sum_{i=1}^{n} p_i \cdot t_i$. This conforms to the principle

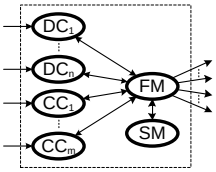


Fig. 3. IMC component model template

that exponential distributions are the optimal statistical approximation (in terms of maximizing the entropy) if only the mean execution times are known. We are aware that this approach may not always imply the same quantitative results compared to using the SADF analysis techniques of SDF[3] but it does allow investigating the scalability of CADP for systems with scenario-based dynamism.

IV. IMC Semantics of SADF

Under the assumption discussed in the previous section, we propose an IMC semantics for SADF. A useful observation is that actors (kernels and detectors) of an SADF model behave quite similar (See [36] for a detailed explanation). They first determine the scenario to operate in. Subsequently, they wait for a number of tokens to become available on their data inputs. Then, time advances according to some execution time distribution and their firing ends with consuming and producing tokens. Kernels and detectors mainly[1] differ in the way they determine the scenario in the first step. Given the similarity, we propose a common model for kernels and detectors consisting of multiple interacting IMCs for each SADF actor together with its input (data and control) channels. The latter simplifies testing on the availability of sufficient tokens while also avoiding larger state spaces when channels are modelled as separate components. In case of the MPEG-4 example in Fig. 1, we therefore get five of such IMC-based component models.

Fig. 3 visualises the constituents of the IMC component model template for an SADF actor with n data and m control inputs. Since SADF abstracts from the content of data tokens, a data channel (DC) can be modelled as a counter that is initialised with the number of initial tokens. The counter can be incremented by the producer as well as inspected and decremented by the consumer. Rates exceeding one are realised with a number of consume/produce actions equal to the corresponding rate. Control channels (CC) are more complex in the sense that the scenario values of control tokens must be stored. We choose to store the token sequence of a control channel as a string of which the first character can be inspected and removed by the consumer, while characters can be concatenated (one-by-one) at the tail by the producer. This allows us to not predefine the size of control buffers. Hence, both data and control channel models in IMC are potentially unbounded

[1]Only detectors produce scenario-valued tokens onto control channels.

(conform the definition in SADF). This provides ample means to identify alternative buffer space distributions as analysis results.

To represent the actual SADF actor, the component model template in Fig. 3 contains two more IMCs. The *Functional Module* (FM) captures the main behaviour of an actor. Its first actions express waiting until a token is available on each control input, which requires interacting with the CCs. Upon availability of such control tokens, FM activates the *Scenario Module* (SM) to delegate establishing the scenario to operate in. The next step of FM after obtaining the scenario from SM is to wait until sufficient tokens are available on the data inputs by interacting with the appropriate DCs. Subsequently, FM includes a Markovian transition (at most one for each scenario) to model waiting according to the appropriate exponentially distributed execution time. The final step of FM is to interact with the DCs and CCs of the considered actor to update their status in correspondence with consuming tokens, while it further interacts with DC and CC IMCs of the destination actors to realise the production of tokens.

The SM in Fig. 3 is responsible for establishing the scenario. In SADF, kernels and detectors do this in slightly different ways. Both operate in a default scenario if there are no control inputs, otherwise the scenario is (partly) determined by interpreting the tokens received on control inputs. For kernels, that is all there is, but for detectors, the scenario is further detailed by making a transition in a Markov chain. In general, a Markov chain is associated to a detector for each possible scenario-value combination of received control tokens [36]. In case a detector has no control inputs, as for the MPEG-4 example in Fig. 1, a single Markov chain details the scenario. However, if a detector has control inputs, a Markov chain is selected based on the scenario-value combination of the control tokens. The state of each Markov chain is retained until the next occurrence of the corresponding scenario-value combination [36]. In our IMC semantics, we use the same modelling approach for kernels and detectors. That is, we associate single state Markov chains (with a single transition) to kernels, one for each scenario in which they can operate. Hence, SM first interprets scenario-valued control tokens (if there are control inputs) and subsequently makes an appropriate probabilistic choice (always with a single option for kernels). Since IMCs do not support probabilistic transitions (but CADP does), we use action transitions with a designated label and replace these labels with concrete probabilities at analysis time, see also Section V.

Parallel composition of the different IMCs constituting the template in Fig. 3 gives an IMC semantics for one SADF actor. Further parallel composition yields an IMC semantics for a complete SADF model. Such parallel composition leads to non-deterministic choices between

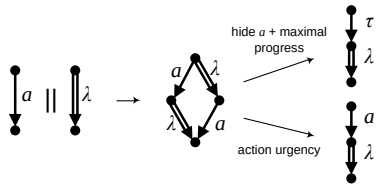


Fig. 4. Maximal progress & action urgency

the transitions that parallel components can perform independently. Many forms of such non-determinism can exist in IMCs [17]. Commonly, the assumption of *maximal progress* is used to resolve non-determinism between internal (τ) actions and Markovian transitions. Internal actions are those actions that do not participate in the interaction between IMCs. Therefore, such internal actions are often hidden for the environment. The upper part of Fig. 4 illustrates the effect of applying maximal progress after parallel composition. In Section II, we discussed the concept of self-timed execution for dataflow models as a means to partially resolve non-determinism: an actor is fired without delays in case *all* its enabled actions are performed prior to advancing time, i.e., not only internal actions that can be hidden for other IMCs. Prioritising any action over progress in time is sometimes called *action urgency* [24], [37]. Its effect is illustrated in the lower part of Fig. 4. The next section discusses constructing the IMC-based state space of the MPEG-4 example with CADP.

V. Formal Verification with CADP

The CADP tool suite provides ample means for the three phases of traditional model checking approaches: state-space generation, state-space reduction and (qualitative and quantitative) analysis. The main benefit of CADP is its ability to exploit compositionality. State spaces can be obtained and be reduced in a component-based manner. All this relies on expressing behaviour in an extended version of LOTOS [2]. CADP supports *action* transitions, *probabilistic* transitions and *Markovian* transitions (but not the time transitions as defined in SADF). Given the natural match with two types of these transitions, IMCs are easily expressible without additional semantic modifications [6].

a) State-Space Generation.: The main steps of mapping SADF models onto the analysis capabilities of CADP are provided in Fig. 5. First, the IMC models of each individual SADF actor are glued together by means of parallel composition. The action transitions allow a flexible way of combining IMCs such that channels are for instance shared between a sending and receiving SADF actor. As now the actions to compose the individual IMCs are not of any relevance, they are abstracted away (hiding). This entails that all action transitions are labeled with the internal action τ. Applying the maximal progress principle of IMCs yields the removal of all Markovian transitions emanating from states that also have outgoing

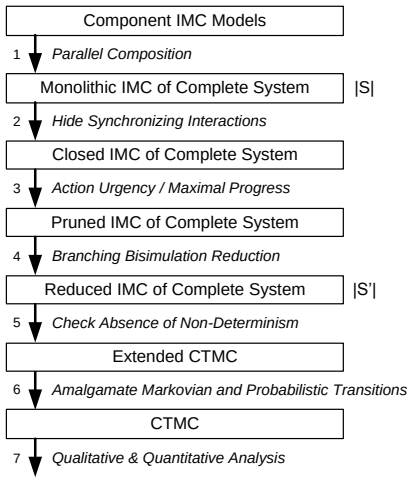


Fig. 5. From SADF models to analysis

τ-transitions. Thus, as a result either a state has only outgoing τ-transitions or outgoing Markovian transitions. Parts that become unreachable by this operation are removed from the state space. Subsequently, the state space of the IMC is reduced using branching bisimulation [15]. In case no non-determinism is left, i.e., no state has two (or more) outgoing τ-transitions —this holds for any SADF model thanks to action-determinism [35]— the resulting IMC is a stochastic process that after amalgamating probabilistic with subsequent Markovian transitions [28] (illustrated in Fig. 6) yields a continuous-time Markov chain (CTMC). CADP allows qualitative model checking on any IMC, including the non-deterministic ones; quantitative analysis however is restricted to CTMCs. All above steps are realised by the scripting language SVL [8] of CADP.

b) State-Space Reductions.: As the SADF buffers are modeled by data types and are in principle unbounded, they do not have a fixed a priori capacity. This means that state-space generation may not terminate. Specific characteristics of the MPEG-4 example (which include consistency between production and consumption rates and the cyclic dependencies) imply a bounded system, but indications about the size of the individual buffers are not available by a static analysis of the SADF model. In case of bounded SADF systems, CADP can generate the state space, possibly apply some reductions, and perform various analyses. The state-space reductions obtained by CADP using branching bisimulation are substantial, cf. the two tables in Table I. The tables differ in the set of scenarios of the MPEG-4 decoder that are supported in the

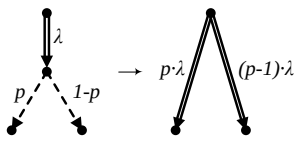


Fig. 6. Eliminating probabilistic transitions

Scenarios $I_{99}, P_0, P_{30}, P_{40}, P_{50}, P_{60}, P_{70}, P_{80}, P_{99}$

PD	$\|S\|$	Memory	Time	$\|S'\|$	Factor	Time
1	121430	148MB	4.34s	20664	*5.88*	14.24s
2	11843682	19.2GB	282s	748813	*15.82*	604s
3	?	>192GB	?	?	?	?

Scenarios $I_{99}, P_0, P_{30}, P_{60}, P_{99}$

PD	$\|S\|$	Memory	Time	$\|S'\|$	Factor	Time
1	89166	121MB	3.98s	12580	*7.09*	12.15s
2	3853042	6.9GB	88s	259145	*14.87*	175s
3	56867106	135GB	2241s	2661313	*21.37*	3279s

TABLE I. State space reductions

model. PD denotes the *pipelining degree* of the MPEG-4 example which coincides with the number of initial tokens on the channel between RC and FD. The parameter PD varies the state space: the more initial tokens are available, the more pipelining (i.e., concurrency) is possible. The first three columns provide the size of the state space $|S|$ of the original, i.e., non-reduced model, the storage requirements for this model, and its generation time. The last three columns provide the size of the minimised state space $|S'|$, the reduction factor and the generation plus minimisation time. The experiments were ran on a 2.6 GHz machine with 192GB main memory. The entries ? in Table I indicate that more memory is needed to cover these models.

c) Qualitative and Quantitative Analysis.: Several functional properties of the MPEG-4 model were analysed using the built-in model checker of CADP. This includes simple reachability properties ("does RC eventually fire?") as well as properties such as "between two consecutive MC firings, RC must fire at least once" and leadsto properties ("after VLD fires, eventually IDCT fires"). To analyse the CTMC obtained from the state space generation and reduction process, CADP supports standard transient analysis ("what is the probability to be in a certain state at time t?") as well as steady-state analysis ("what is the state probability on the long term?"). Such analysis can be readily applied to the minimised models. A more intricate performance measure that has been considered is the response time distribution of the RC component. That is to say, "what is the probability that RC finishes its first firing within a certain time?". We accomplished this by modeling an observer to the IMC of the SADF component which synchronises on the first firing completion of the RC component. The time to reach the final state of the observer now yields the requested measure. The results are depicted in Fig. 7. This form of analysis is not supported by SDF^3, which only allows computing some statistics of the response time distribution being the minimum, maximum and expected response time. This shows the added value of using a model checking tool like CADP.

VI. Lessons Learned

Applying CADP for model checking SADF revealed several new insights compared to the SDF^3 approach. The

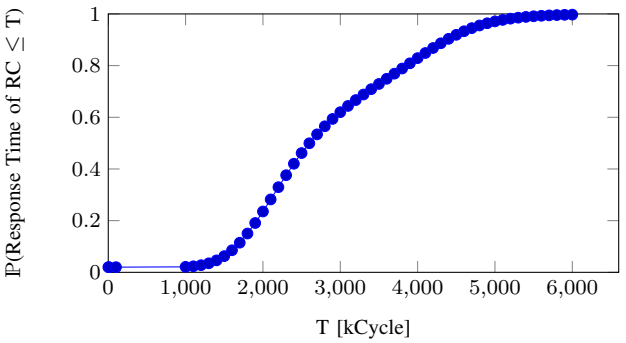


Fig. 7. Response time of RC

latter uses shared data structures for the channels, which are not an integrated part of the processes as opposed to the IMC actor model template in Fig. 3. Moreover, instead of using explicit interactive actions, SDF^3 currently solely uses guards on the shared data structures modelling the channels (i.e., interactions are implicit). The compositional approach shown in this paper is however perfectly transferable to an alternative implementation in SDF^3. The same holds for exploiting the similarity between kernels and detectors discussed in Section III, which could unify separate functionality in SDF^3 for kernels and detectors, thereby potentially improving efficiency.

Contemporary model checkers like CADP provide more flexibility over dedicated ones like SDF^3, especially in terms of the range of analysable properties. On the other hand, SDF^3 mostly supports reward-based properties that could not be evaluated with CADP. Support for evaluating such properties could boost applicability of CADP. Due to the different time models of CADP and SDF^3, their effectiveness (in terms of quantitative analysis results) and efficiency (in terms of state space sizes) seem however fairly incomparable. Future work includes identifying what properties are invariant to the difference in time models.

References

[1] E. Böde, et al. Compositional Dependability Evaluation for STATEMATE. *IEEE Tr. on Softw. Eng.*, vol. 35, 274–292, 2009

[2] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Comp. Netw.*, vol. 14, 25–59, 1987

[3] A. Bonfietti, et al. An Efficient and Complete Approach for Throughput Maximal SDF Allocation and Scheduling on Multi-Core Platforms. *DATE*, ACM, pp. 897–902, 2010

[4] H. Boudali, et al. Rigorous, Compositional, and Extensible Framework for Dynamic Fault Tree Analysis. *IEEE Tr. on Secure and Dependable Comp.*, vol. 7, 128–143, 2009

[5] M. Bozzano, et al. Safety, Dependability and Performance Analysis of Extended AADL Models. *Comp. J.*, vol. 54(5), 754–775, 2011

[6] N. Coste, et al. Ten Years of Performance Evaluation for Concurrent Systems using CADP. *ISOLA*, LNCS 6416, pp. 128–142, 2010

[7] N. Coste, et al. Quantitative Evaluation in Embedded System Design: Validation of Multiprocessor Multithreaded Architectures. *DATE*, ACM, pp. 88–90, 2008

[8] H. Garavel and F. Lang. SVL: A Scripting Language for Compositional Verification. *FORTE*, Kluwer, pp. 377–394, 2001

[9] H. Garavel, et al. CADP 2010: A Toolbox for the Construction and Analysis of Distributed Processes. *TACAS*, LNCS 6605, pp. 372–387, 2011

[10] M. Geilen and T. Basten. Requirements on the Execution of Kahn Process Networks. *ESOP*, LNCS 2618, pp. 319–334, 2003

[11] M. Geilen and S. Stuijk. Worst-case Performance Analysis of Synchronous Dataflow Scenarios. *CODES+ISSS*, pp. 125–134, 2010

[12] A. Ghamarian, et al. Liveness and Boundedness of Synchronous Data Flow Graphs. *FMCAD*, IEEE, pp. 68–75, 2006

[13] A. Ghamarian, et al. Throughput Analysis of Synchronous Data Flow Graphs. *ACSD*, IEEE, pp. 25–36, 2006

[14] A. Ghamarian, et al. Latency Minimization for Synchronous Data Flow Graphs. *DSD*, IEEE, pp. 189–196, 2007

[15] R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *J. of the ACM*, vol. 43, 555–600, 1996

[16] W. Haid, et al. Multiprocessor SoC Software Design Flows. *Signal Processing Magazine*, vol. 26, no. 6, pp. 64–71, 2009

[17] H. Hermanns. Interactive Markov Chains and the Quest for Quantified Quality. LNCS 2428, 2002

[18] H. Hermanns and J.P. Katoen. The How and Why of Interactive Markov Chains. *FMCO'09*, LNCS 6286, pp. 311–337, 2009

[19] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985

[20] G. Kahn. The Semantics of a Simple Language for Parallel Programming. *Proc. of IFIP'74*, pp. 471–475, 1974

[21] E. Lee and D. Messerschmitt. Static Scheduling of Synchronous Data Flow Programs for Digital Signal Processing. *IEEE Tr. on Comp.*, vol. 36, no. 1, pp. 24–35, 1987

[22] W. Liu, et al. Efficient SAT-based Mapping and Scheduling of Homogeneous Synchronous Dataflow Graphs for Throughput Optimization. *RTSS*, pp. 492-504, 2008

[23] O. Moreira, et al. Multiprocessor Resource Allocation for Hard-Real Time Streaming with a Dynamic Job-Mix. *RTAS*, pp. 332-341, 2005

[24] X. Nicollin and J. Sifakis. An Overview of Synthesis on Timed Process Algebras. *CAV*, pp. 376–398, 1991

[25] T. Parks. *Bounded Scheduling of Process Networks*. PhD Thesis, University of California, Berkeley, 1995

[26] A. Pimentel. The Artemis Workbench for System-level Performance Evaluation of Embedded Systems. *J. of Embedded Systems*, vol. 3, no. 3, pp. 181-196, 2008

[27] P. Poplavko, et al. Execution-Time Prediction for Dynamic Streaming Applications with Task-Level Parallelism. *DSD*, IEEE, pp. 228–235, 2007

[28] M. Rettelbach. Probabilistic Branching in Markovian Process Algebras. *Comp. J.*, vol. 38, no. 7, 1995.

[29] S. Sriram and S. Bhattacharyya. *Embedded Multiprocessors: Scheduling and Synchronization*, Second Edition. CRC Press, 2009

[30] S. Stuijk, et al. SDF^3: SDF For Free. *ACSD*, IEEE, pp. 276–278, 2006

[31] S. Stuijk, et al. Throughput-Buffering Trade-Off Exploration for Cyclo-Static and Synchronous Dataflow Graphs. *IEEE Tr. on Comp.*, vol. 57, no. 10, pp. 1331–1345, 2008

[32] S. Stuijk, et al. A Predictable Multiprocessor Design Flow for Streaming Applications with Dynamic Behaviour. *DSD*, IEEE, pp. 548–555, 2010

[33] S. Stuijk, et al. Scenario-Aware Dataflow: Modeling, Analysis and Implementation of Dynamic Applications. *SAMOS*, pp. 404–411, 2011

[34] B.D. Theelen. A Performance Analysis Tool for Scenario-Aware Streaming Applications. *QEST'07*, IEEE, pp. 269–270, 2007

[35] B.D. Theelen, et al. A Scenario-Aware Data Flow Model for Combined Long-Run Average and Worst-Case Performance Analysis. *MEMOCODE*, pp. 185–194, 2006

[36] B.D. Theelen, et al. *Scenario-Aware Dataflow*. Technical Report ESR-2008-08, Eindhoven University of Technology, July 2008

[37] B.D. Theelen, et al. Performance Model Checking Scenario-Aware Dataflow. *FORMATS*, LNCS 6919, pp. 43–59, 2011

[38] M. Wiggers, et al. Efficient Computation of Buffer Capacities for Multirate Real-Time Systems with Back-Pressure. *CODES+ISSS*, pp. 10–15, 2006