Parameterized dataflow scenarios

Citation for published version (APA):

Skelin, M., Geilen, M. C. W., Catthoor, F., & Hendseth, S. (2017). Parameterized dataflow scenarios. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(4), 669-682. [7529122].
<https://doi.org/10.1109/TCAD.2016.2597223>

DOI:

[10.1109/TCAD.2016.2597223](https://doi.org/10.1109/TCAD.2016.2597223)

Document status and date:

Published: 01/04/2017

Document Version:

Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Parameterized Dataflow Scenarios

Mladen Skelin, Marc Geilen, *Member, IEEE*, Francky Catthoor, *Fellow, IEEE*, and Sverre Hendseth

Abstract—A number of modeling approaches combining dataflow and finite-state machines (FSMs) have been proposed to capture applications that combine streaming data with finite control. FSM-based scenario-aware dataflow (FSM-SADF) is such an FSM/dataflow hybrid that occupies a sweet spot in the tradeoff between analyzability and expressiveness. However, the model suffers from compactness issues when the number of scenarios increases. This hampers its use in analysis of applications exposing high levels of data-dependent dynamics.

In this article, we address this problem by combining parameterized dataflow with finite control of FSM-SADF. We refer to the generalization as FSM-based parameterized scenario-aware dataflow (FSM- π SADF). We introduce the formal semantics of the model, in terms of Max-plus algebra and in particular Max-plus automata. Thereafter, by leveraging the existing results of FSM-SADF, we propose a worst-case performance analysis framework for FSM- π SADF. We show that by using FSM- π SADF and its analysis framework, one can, unlike with FSM-SADF, compactly capture streaming applications exhibiting high levels of data-dependent dynamics in presence of finite control. Furthermore, we show that for practical models our analysis typically yields tighter bounds on worst-case performance indicators such as throughput and latency than the existing techniques based on conservative FSM-SADF modeling (if such modeling can be applied at all). We evaluate our approach on a realistic case-study from the multimedia domain.

Index Terms—parameterized dataflow, finite-state machine, parameterized dataflow scenarios, synchronous dataflow, Max-plus algebra, worst-case performance.

I. INTRODUCTION

DATAFLOW models of computation (MoC) are widely used for modeling streaming applications.

A graphical representation of a dataflow MoC is a dataflow graph. In dataflow graphs, nodes are called *actors*, while edges are called *channels*. Actors represent computational kernels, while channels capture the flow of streams of data values between actors. These data values are called *tokens*. Tokens found in the graph before its execution commences form the set of *initial* tokens and their distribution over channels defines the initial state of the graph. Actors communicate by *firing*. Typically, a firing consumes tokens from the input streams and produces tokens on the output streams. The numbers of tokens consumed and produced are called *rates*. In timed dataflow, actor firing takes a finite amount of time called the actor *firing delay*.

Purely dataflow-based MoCs have the ability to capture concurrency contained in the considered application. However, concurrency is not the only source of complexity. In addition, applications may have intricate control requirements [1]. Therefore, heterogeneous MoCs based on combinations of dataflow and finite-state machines (FSMs) have been proposed to capture such applications. A prominent example of an FSM/dataflow hybrid is FSM-based scenario-aware dataflow

(FSM-SADF). With FSM-SADF, the dynamic run-time behavior of an application is viewed as an evolving sequence of static behaviors called *modes* or *scenarios*. Each scenario is modeled in synchronous dataflow (SDF) formalism [2]. FSM-SADF is a timed dataflow MoC primarily used for worst-case performance analysis of streaming application and systems. An FSM-SADF graph (FSM-SADFG) evolves in iterations of its constituent SDF graphs (SDFGs). An iteration of a dataflow graph is a minimal non-empty set of actor firings that restores the initial token distribution of the graph. The numbers of times particular actors need to be fired within an iteration are organized in a vector called the repetition vector of the graph.

By maintaining determinism and predictability of SDF, while introducing nondeterminism only at the scenario level, FSM-SADF occupies a sweet spot in the tradeoff between expressiveness and analyzability [3]. However, as the number of scenarios grows, FSM-SADF will start to suffer from compactness issues. This hampers the use of FSM-SADF in worst-case performance analysis of applications exhibiting high levels of data-dependent dynamics because the analysis run-time will be prohibitive. This is particularly pronounced if scenario SDF models attain large repetition vector entries [4].

We address these problems by combining the finite control of FSM-SADF with parameterized dataflow into a construct we refer to as FSM-based parameterized scenario-aware dataflow (FSM- π SADF). In particular, we model each scenario using a parameterized synchronous dataflow graph (π SDFG). Parameterization, as a syntactic construct, allows us to represent vast sets of scenarios in a compact way (parameters help keep the size of the model manageable) and to explicitly represent the dependencies between parameters.

FSM- π SADF can in *some cases* be analyzed conservatively in a rather simple way. This is done by choosing parameter values that define worst-case behavior per parameterized scenario, i.e. by taking their upper bounds. The result is a conservative FSM-SADF abstraction of the original parameterized specification that is then subjected to analysis. However, in practical models, the values of parameters are often dependent on one another which means they will rarely attain the values of their upper bounds. In particular, in words of Neuendorffer and Lee [5], these dependencies may be specified explicitly in the construction of the model, e.g., one parameter is given as an expression of another or implicitly, e.g., a dataflow scheduler synthesizes some parameter values. Furthermore, even if we exclude parameter dependencies, in many constellations using upper bounds of parameters concerning graph rates may not yield a conservative abstraction at all but on optimistic one. The latter is unacceptable in a real-time setting where we need to guarantee or prove worst-case timing bounds.

Therefore, as the main contribution of this article, we propose a parametric (symbolic) worst-case performance analysis framework for FSM- π SADF based on the Max-plus algebraic semantics of self-timed execution of SDF [4] and the theory of Max-plus automata used to capture transitions between scenarios. The analysis replaces successive analysis of each parameter value combination by a single parametric (symbolic) analysis and so alleviates FSM-SADF compactness issues. The analysis is conservative, but due to the approximations made, less precise than the enumerative analysis. We show that for practical models where the values of parameters are often

Manuscript received November 14, 2015; revised February 24, 2016 and May 11, 2016; accepted June 28, 2016. This work was partly supported by ITEA 3 project 14014 ASSUME.

M. Skelin and M. Geilen are with the Eindhoven University of Technology, Eindhoven, The Netherlands. F. Catthoor is with IMEC vzw, Leuven, Belgium. S. Hendseth is with the Norwegian University of Science and Technology, Trondheim, Norway. Major part of this work was performed while M. Skelin was affiliated with the Norwegian University of Science and Technology.

Copyright (c) 2015 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org

dependent on one another [5], our analysis techniques will typically yield tighter worst-case performance bounds than conservative FSM-SADF abstractions of input FSM- π SADF specifications where worst-case assignment of values to parameters is chosen by taking their upper bounds. Of course, the assumption here is that these conservative FSM-SADF abstractions can be constructed in the first place. This is often not the case due to effect of rates on the temporal behavior of the graphs. In particular, an increase in rate (parameter) value may result in the decrease of the graph repetition vector entries and consequently to a shorter duration of the iteration. Therefore, it is unclear *a priori* which values of rates define the worst-case behavior of the graph. In this (the latter) case, only our technique can guarantee a conservative result. In the former case, both techniques can co-exist and the tighter of the two results should be taken.

We demonstrate our approach on a realistic case-study from the multimedia domain.

The remainder of this article is structured as follows. Section II discusses related work. Section III introduces the preliminary concepts. Section IV presents our parameterized SDF modeling. Section V presents the FSM- π SADF model. Section VI describes the overall structure of our worst-case performance analysis framework for FSM- π SADF discussed in detail in Sections VII and VIII. Section IX evaluates our modeling and analysis framework on a VC-1 decoder case study. Section X concludes.

A preliminary version of this work can be found in [6]. Furthermore, the omitted proofs can be found in the report version of this article [7].

II. RELATED WORK

Parameterized dataflow as a meta-modeling technique was introduced by [8] and developed in the context of SDF yielding parameterized SDF (PSDF).

Schedulable parametric dataflow (SPDF) and boolean parametric dataflow (BPDF) introduced in [9] and [10], respectively are MoCs closely related to PSDF. They explicitly define requirements that a parameterized dataflow specification must satisfy so that questions about deadlock-freedom, boundedness and schedule construction can be answered at design-time.

Variable-rate dataflow (VRDF) of [11] facilitates frequent changes of actor port rates by means of parameterization. However, VRDF defines strong structural constraints that must be satisfied for achieving boundedness. VPDF [12] is a generalization of VRDF based on cyclo-static dataflow (CSDF) [13] where actors operate through sequences of phases and in each phase, the number of actor firings is parameterized along with the rates.

Parameterized and interfaced dataflow meta-model or shortly PiMM [14] is obtained by enriching the meta-modeling techniques of [8] with the notion of interfaces as introduced in interface based synchronous dataflow (IBSDF) [15].

All abovementioned parameterized dataflow MoCs can compactly express high levels of data-dependent dynamics in applications. However, they do not depart from the dataflow framework and thus provide no means to expose application control requirements directly to the programmer (preferably by means of a well-defined state structure). In addition, all of the models described except VPDF and VRDF are untimed and have no known performance analysis techniques.

Next, we list models that do foster provision for expressing intricate control logic by defining precise semantics for integration of FSMs and dataflow.

Article [1] advocates the use of a combination of hierarchical state machines and various concurrent MoCs to decouple control from concurrency. When SDF is used in

conjunction with FSMs, the resulting model is referred to as heterochronous dataflow (HDF). HDF has only been studied in the context of sequential schedules. Therefore, it does not consider pipelined execution and the implications of pipelined scenario transitions that we consider in this article. Furthermore, HDF is untimed and has no known performance analysis techniques.

Scenario-aware dataflow (SADF) MoC introduced in [16] enables modeling and analysis of dynamic systems by allowing actors to operate in different *modes* or *scenarios* across firings. In different scenarios, actors have different execution times and rates. SADF uses a Markov chain to model scenario occurrence patterns.

FSM-SADF [4] is a restriction of SADF in the sense that with FSM-SADF scenarios can change only between complete iterations of SDF models of the respective scenarios, while with SADF scenario changes are allowed even within an iteration. Furthermore, in FSM-SADF, for worst-case analysis, we abstract from the transition probabilities of SADF Markov chain and obtain a nondeterministic FSM. The overall reduction in expressive power compared to SADF is advantageous from the analysis perspective.

The DF* (pronounced “DFstar”) modeling framework of [17] is another dataflow MoC in the family of FSM/dataflow hybrids. A DF* graph is a network of blocks where each block consists of a set of code segments and a block controller. Each code segment specifies an alternative behavior of the block. The block controller is captured by a nondeterministic FSM.

The FunState MoC introduced in [18] defines precise semantics for separating dataflow from control in terms of functions driven by state machines.

Article [19] adds control flow provisions to bounded dynamic dataflow (BDDF) introduced in the same work yielding another FSM/dataflow hybrid.

The modeling and simulation framework called El Greco [20] supports system specifications given as combinations of dataflow graphs and hierarchical FSMs. However, the framework is tailored for rapid simulation-based algorithm exploration. Therefore, it is not clear from [20] how static analysis is performed in the presence of parameters.

Core functional dataflow (CFDF) [21][22] is based on a formalism where actors execute through dynamic transitions between modes specified by a deterministic FSM. Hierarchical CFDF (HCFDF) [23][24] generalizes CFDF by introducing a special kind of actor called a decision actor that enables finite-state control at the application level. However, both CFDF and HCFDF are deterministic Turing complete models [25]. Therefore, they do not allow for sufficient design-time analyzability such as analysis for boundedness and deadlock-freedom, let alone worst-case performance analysis we focus on in this article.

A common characteristic to all of the abovementioned hybrids regardless of their expressiveness is that they suffer from compactness issues when dealing with applications exposing high levels of data-dependent dynamics. Furthermore, all of them except SADF and FSM-SADF are untimed and have no known performance analysis techniques.

The only hybrids we know that can compactly capture applications with both control requirements and high levels of data-dependent dynamics are core functional parameterized synchronous dataflow (CF-PSDF) [26] and parameterized set of modes CFDF (PSM-CFDF) [27] that combine CFDF and parameterized dataflow. However, as CFDF, CF-PSDF and PSM-CFDF are untimed Turing complete models tailored for rapid simulation and synthesis and relevant properties, such as worst-case throughput and latency that we focus on in this article cannot be determined at design-time. Furthermore,

CFDF, CF-PSDF and PSM-CFDF are deterministic (their FSM is deterministic) in contrast to FSM-SADF and FSM- π SADF that allow for nondeterministic variation of modes.

III. PRELIMINARIES

A. Max-plus algebra

We briefly introduce basic Max-plus algebra notation. Define $\mathbb{R}_{\max} = \mathbb{R} \cup \{-\infty\}$, where $\mathbb{R}$ is the set of real numbers. Let $a \oplus b = \max(a, b)$ and $a \otimes b = a + b$ for $a, b \in \mathbb{R}_{\max}$. For $a \in \mathbb{R}_{\max}$, $-\infty \oplus a = a \oplus -\infty = a$ and $a \otimes -\infty = -\infty \otimes a = -\infty$, i.e. $-\infty$ is the zero element of the $\oplus$ operation. By Max-plus algebra we understand the analogue of linear algebra developed for the pair of operations $(\oplus, \otimes)$ extended to matrices and vectors and denoted by $\mathcal{R}_{\max} = \{\mathbb{R}_{\max}, \oplus, \otimes\}$. The set of n -dimensional Max-plus vectors is denoted $\mathbb{R}_{\max}^n$, while $\mathbb{R}_{\max}^{n \times n}$ denotes the set of $n \times n$ Max-plus matrices. The (sup-) sum of matrices $A, B \in \mathbb{R}_{\max}^{n \times n}$, denoted by $A \oplus B$ is defined by $[A \oplus B]_{i,j} = [A]_{i,j} \oplus [B]_{i,j}$ where $[A]_{i,j}$ and $[B]_{i,j}$ are entries of matrices A and B with indices i and j . The matrix product $A \otimes B$ is defined by $[A \otimes B]_{i,j} = \bigoplus_{k=1}^n [A]_{i,k} \otimes [B]_{k,j}$. For a vector $\mathbf{a} = [a_1, \dots, a_n]^T \in \mathbb{R}_{\max}^n$, $\|\mathbf{a}\|$ denotes the vector norm, defined as $\|\mathbf{a}\| = \bigoplus_{i=1}^n a_i$. For a vector $\mathbf{a}$ with $\|\mathbf{a}\| > -\infty$, we use $\mathbf{a}^{\text{norm}}$ to denote $\mathbf{a} - \|\mathbf{a}\| = [a_1 - \|\mathbf{a}\|, \dots, a_n - \|\mathbf{a}\|]^T$, i.e. the normalized vector $\mathbf{a}$, so that $\|\mathbf{a}^{\text{norm}}\| = 0$. With $A \in \mathbb{R}_{\max}^{n \times n}$ and $c \in \mathbb{R}$, we use denotations $A \otimes c$ or $c \otimes A$ for matrix where $[A \otimes c]_{i,j} = [c \otimes A]_{i,j} = [A]_{i,j} + c$. The $\otimes$ symbol in the exponent indicates a matrix power in Max-plus algebra. For $A \in \mathbb{R}_{\max}^{n \times n}$, $A^{\otimes k} = \bigotimes_k A$ where $k \in \mathbb{N}_{>0}$.

For scalars $c \in \mathbb{R}$ and $\alpha \in \mathbb{R}$, $c^{\otimes \alpha} = \alpha \cdot c$ where $\cdot$ stands for multiplication in “regular algebra”. Furthermore, it is easy to verify that Max-plus matrix multiplication is linear, i.e. $M \otimes (\mathbf{a} \oplus \mathbf{b}) = M \otimes \mathbf{a} \oplus M \otimes \mathbf{b}$ and $M \otimes (c \otimes \mathbf{a}) = c \otimes M \otimes \mathbf{a}$ for all $M \in \mathbb{R}_{\max}^{n \times n}$, $\mathbf{a}, \mathbf{b} \in \mathbb{R}_{\max}^n$ and $c \in \mathbb{R}_{\max}$. Now, let $M, N \in \mathbb{R}_{\max}^{n \times n}$. We write $M \preceq N$ if $[M]_{i,j} \leq [N]_{i,j}$ for all $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, n\}$. In addition, the matrix multiplication is monotone, which means that if $\mathbf{a} \preceq \mathbf{b}$, then $M \otimes \mathbf{a} \preceq M \otimes \mathbf{b}$. Similarly, if $M \preceq N$, then $M \otimes \mathbf{a} \preceq N \otimes \mathbf{a}$.

B. Sequences in $\mathbb{R}_{\max}$

In the Max-plus theory, inputs, outputs and the state of the system are represented by a vector $\gamma \in \mathbb{R}_{\max}^n$ of dater functions [28]. In particular, the i th entry of the vector is a map $\gamma_i : \mathbb{N}_0 \rightarrow \mathbb{R}_{\max}$ where $\gamma_i(k)$ is usually interpreted as the time of k th occurrence of event labeled i . Consequently, we use the notation $\gamma(k)$ for the vector $[\gamma_1(k), \dots, \gamma_n(k)]^T$. In this article, we will refer to dater functions as *sequences*. Below we define types of sequences relevant for this article.

Definition 1 (Eventually periodic sequence). *Let $c \in \mathbb{N}_{>0}$ and let $\pi \in \mathbb{R}$. A sequence $\gamma : \mathbb{N}_{>0} \rightarrow \mathbb{R}_{\max}$ in $\mathbb{R}_{\max}$ is eventually periodic with period c and ratio π if there exists $T \in \mathbb{N}_{>0}$ such that:*

$$\forall n \geq T : \gamma(n+c) = c \cdot \pi + \gamma(n) = \pi^{\otimes c} \otimes \gamma(n). \quad (1)$$

A special case of an eventually periodic sequence is a linear sequence we define next.

Definition 2 (Linear sequence). *A linear sequence is an eventually periodic sequence $\gamma : \mathbb{N}_{>0} \rightarrow \mathbb{R}_{\max}$ with $c = 1$ such that*

$$\forall n \in \mathbb{N}_{>0} : \gamma(n) = \delta + \pi \cdot n = \delta \otimes \pi^{\otimes n}, \quad (2)$$

where $\delta \in \mathbb{R}$. We refer to the tuple (δ, π) as the *delay-ratio pair*.

In the definition to follow we discuss how to conservatively bound general eventually periodic sequences with linear sequences.

Definition 3 (Conservative linear upper bound). *Let $\gamma : \mathbb{N}_{>0} \rightarrow \mathbb{R}_{\max}$ be an eventually periodic sequence in $\mathbb{R}_{\max}$ with ratio π . A linear sequence $\hat{\gamma} : \mathbb{N}_{>0} \rightarrow \mathbb{R}_{\max}$ defined $\hat{\gamma}(n) = \delta \otimes \pi^{\otimes n}$ is a conservative linear upper bound (CLUB) to γ if*

$$\forall n \in \mathbb{N}_{>0} : \gamma(n) \leq \hat{\gamma}(n). \quad (3)$$

We now show how to define a CLUB of a sup-sum of a set of linear sequences. In particular, the following proposition holds.

Proposition 1. *Let $\Sigma = \{\gamma_i(n) : i \in \mathbb{N}_{>0}\}$ be a set of linear sequences in $\mathbb{R}_{\max}$, i.e. $\gamma_i(n) = \delta_i \otimes \pi_i^{\otimes n}$. Let $\delta = \max_i \delta_i$ and $\pi = \max_i \pi_i$ and let $\gamma(n) = \bigoplus_i \gamma_i(n)$. Then, γ attains the following CLUB*

$$\hat{\gamma}(n) = \delta \otimes \pi^{\otimes n}. \quad (4)$$

Another crucial concept to be defined is the one of Max-plus- or sup- convolution.

Definition 4. *Let $\gamma_{1,2} : \mathbb{N}_{>0} \rightarrow \mathbb{R}_{\max}$ be two sequences in $\mathbb{R}_{\max}$. In analogy to classical convolution, one defines the convolution of the two, denoted $\text{conv}(\sigma_1, \sigma_2)$ as*

$$\text{conv}(\gamma_1, \gamma_2)(n) = \bigoplus_{i=1}^n \gamma_1(n-i+1) \otimes \gamma_2(i), \quad (5)$$

for all $n \in \mathbb{N}_{>0}$.

Note that the convolution of (5) deviates in terms of accounted indices from the usual definitions of sup-convolution that can be found in the literature and is therefore somewhat proprietary to this article for the purpose of studying the properties of actor responses in Section VII.

We now show how to derive an explicit function for the CLUB of the convolution of two linear sequences of a particular type. We formulate the result in Proposition 2.

Proposition 2. *Let $\gamma_{1,2} : \mathbb{N}_{>0} \rightarrow \mathbb{R}_{\max}$ be two linear sequences in $\mathbb{R}_{\max}$ such that $\gamma_1(n) = \delta_1 \otimes \pi_1^{\otimes \lceil r \cdot n \rceil}$ and $\gamma_2(n) = \pi_2^{\otimes n}$, where $n \in \mathbb{N}_{>0}$, $r \in \mathbb{Q}_{\geq 0}$ and $\delta_1, \pi_1, \pi_2 \in \mathbb{R}_{\max}$. Then, sequence $\hat{\gamma}$ is a CLUB to $\text{conv}(\gamma_1, \gamma_2)(n)$ and is defined as follows:*

$$\hat{\gamma}(n) = \begin{cases} \delta_1 \otimes \pi_1^{\otimes (1+r)} \otimes \pi_2^{\otimes n} & \text{if } \pi_2 \geq r \cdot \pi_1 \\ \delta_1 \otimes \pi_2 \otimes \pi_1^{\otimes (1+r \cdot n)} & \text{if } \pi_2 \leq r \cdot \pi_1 \end{cases}. \quad (6)$$

C. Max-plus linear system theory

In Baccelli et al. [29] it has been shown that the state-vector of a stationary autonomous Max-plus-linear systems evolves as follows

$$\gamma(k+1) = M \otimes \gamma(k) \quad (7)$$

where $M \in \mathbb{R}_{\max}^{n \times n}$, $k \in \mathbb{N}_0$ and $\gamma(0) \in \mathbb{R}_{\max}^n$ is a special vector called the initial condition.

The asymptotic growth rate of $\|\gamma(k)\|$ will attain the value that is equal to the maximum cycle mean (MCM) of the communication graph of M . The communication graph of $M \in \mathbb{R}_{\max}^{n \times n}$, denoted $\mathcal{G}(M) = (\mathcal{N}, \mathcal{E})$, is a graph with

the set of nodes given by $\mathcal{N} = \{1, \dots, n\}$ where a pair $(i, j) \in \mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is an edge of the graph if $[M]_{j,i} \neq -\infty$ and $[M]_{j,i}$ is the weight of that edge.

As mentioned, systems described in (7) are Max-plus-linear and therefore satisfy the superposition principle. This means that in the computation of their response like in conventional linear system theory, if convenient, we can consider one input at a time while assuming that all others are available at $t = -\infty$.

Equation (7) considers stationary autonomous Max-plus linear systems. However not all systems are stationary. In particular, in this article we will be especially interested in non-stationary autonomous Max-plus linear systems with finitely valued dynamics, i.e. systems the state-vector of which evolves as follows

$$\gamma(k+1) = M(k) \otimes \gamma(k), \quad (8)$$

where $M(k)$ takes its values in a finite set $\{M_{s_i} : i \in \mathbb{N}_{>0}\}$. In the context and terminology of this article we think of these systems as systems evolving through modes or scenarios of the set $S = \{s_i : i \in \mathbb{N}_{>0}\}$ each of which is captured in matrix M_{s_i} where $i \in \mathbb{N}_{>0}$. Gaubert [28] shows how these systems can be effectively captured using the concept of Max-plus automata. A Max-plus automaton is a triple $\mathcal{A} = (\alpha, \mu, \beta)$ used to compute the completion time of scenario schedule $\bar{s} = s_1, \dots, s_k \in S^*$ as follows

$$\mathcal{A}(\bar{s}) = \alpha^T \otimes \mu(\bar{s}) \otimes \beta, \quad (9)$$

where $\alpha \in \mathbb{R}_{\max}^n$ is the final delay, $\mu : S^* \rightarrow \mathbb{R}_{\max}^{n \times n}$ is the morphism that associates sequences of scenarios with Max-plus matrices as follows

$$\mu(\bar{s}) = M_{s_k} \otimes \dots \otimes M_{s_1} \quad (10)$$

and $\beta \in \mathbb{R}_{\max}^n$ is the initial delay. If we define the throughput of a scenario sequence as the long-run average of completed scenarios per time unit, then the worst-case throughput of the system is equal to the throughput of the slowest sequence. This in turn corresponds to the worst-case increase of $\mathcal{A}(\bar{s})$ in (9). This worst-case increase can be effectively computed as the MCM of the equivalent communication graph of the matrix

$$M^{\text{w-c}} = \bigoplus_{s_i \in S} M_{s_i} \quad (11)$$

that we call the worst-case evaluation matrix of the set $\{M_{s_i} : i \in \mathbb{N}_{>0}\}$ (cf. Theorem 2 of [28]). Article [28] also shows how, given a regular sublanguage of S^* , the set of all finite scenario sequences of arbitrary length, the maximum growth rate can be determined by a product of automata. For more details we refer to [28] and [30].

D. Synchronous dataflow

SDF [2] is the most widely used dataflow MoC. In SDF, actor rates and firing delays are fixed and known at compile-time. Consider the scenario s_1 SDF graph (SDFG) of Fig. 1. It has six actors. Rates are indicated next to the channel ends, while actor firing delays are annotated alongside the actor's names. In SDF, an actor can fire only if sufficient tokens are available on the channels it consumes from. Black dots depict initial tokens that are understood to be initial conditions for graph's execution [31]. We formally define an SDFG in Definition 5.

Definition 5 (SDFG). An SDFG $G = (\mathcal{A}, C, d, r, i)$ is a tuple where $\mathcal{A}$ is the set of actors, $C \subseteq \mathcal{A} \times \mathcal{A}$ is the multiset of

Fig. 1. FSM-SADF.

channels, $d : \mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$ returns for each actor its associated firing delay, $r : C \rightarrow \mathbb{N}_{>0} \times \mathbb{N}_{>0}$ returns for each channel its source and destination port rates and $i : C \rightarrow \mathbb{N}_0$ returns for each channel its number of initial tokens.

Thanks to their static nature, SDFGs can be scheduled at compile-time. Schedule for an SDFG and all dataflow graphs we consider in this article is a loop over a series of actor firings completing an *iteration*. For the running example graph, the term $A_0^1 A_1^2 A_2^3 A_3^3 A_4^2 A_5^1$ denotes its schedule. Exponents represent actor repetition counts. We define the repetition vector of the graph as a map $\Gamma : \mathcal{A} \rightarrow \mathbb{N}_{>0}$. With the abuse of notation, for the running example, $\Gamma(A_0, A_1, A_2, A_3, A_4, A_5) = (1, 2, 3, 3, 2, 1)$.

E. Max-plus algebra for SDF

Article [32] shows how Max-plus algebra [29] can be used to capture the semantics of self-timed execution of SDF. In particular, the production times of initial tokens after the k th graph iteration are recorded in the *timestamp vector* $\gamma(k) \in \mathbb{R}_{\max}^{|I|}$ of the k th graph iteration, where I is the set of graph's initial tokens (notation used throughout the article) and $|I|$ denotes its cardinality. Note that each initial token has one entry in $\gamma(k)$. Then, the evolution the graph across its iterations can be captured with (7) where $M = M_G$. We call M_G the SDFG Max-plus matrix. Each matrix entry defines the minimum timing distance between two initial tokens in consecutive iterations. If the entry equals $-\infty$, no dependency exists. From the monotonicity of Max-plus it follows that SDF is monotone too [32].

Matrix M_G can be derived by symbolically executing one iteration of the SDFG. The procedure is specified in Algorithm 1 of [32].

F. Synchronous dataflow scenarios

The concept of synchronous dataflow scenarios [32] extends the expressive power of SDF by combining streaming data and finite control into a MoC called FSM-SADF [4]. More precisely, different application behaviors are clustered into a group of static modes of operation called *scenarios* each modeled by an SDFG that we call the scenario SDFG. Scenarios dynamically interact to produce the original application behavior where the interaction patterns are constrained by a nondeterministic FSM. Consider the example FSM-SADF of Fig. 1a. The graph has two scenarios: s_1 and s_2 modeled by two scenario SDFGs. The scenario FSM has two states where each of the states corresponds to one scenario. In the figure, state ϕ_1 corresponds to s_1 (notation $\Psi(\phi_1) = s_1$), while ϕ_2 corresponds to s_2 (notation $\Psi(\phi_2) = s_2$). The scenario FSM defines admissible scenario sequences. The operational semantics of the model is as follows: every transition in the scenario FSM schedules the execution of one iteration of the SDFG that models the scenario corresponding to the transition's destination state.

G. Max-plus algebra for synchronous dataflow scenarios

Paper [4] shows that FSM-SADF is a non-stationary Max-plus linear system the temporal behavior of which can be captured using Max-plus automata. In particular, the completion time of a sequence of scenarios $\bar{s} = s_1, \dots, s_k \in S^* \cap L$ where L defines a restriction of S^* defined by the scenario FSM can be given by (9) and (10) where M_{s_i} is the Max-plus matrix of scenario s_i SDFG. Typically, $\beta = \gamma(0) = \mathbf{0}$ and $\alpha = \mathbf{0}$. The structure is used to derive the worst-case performance metrics of FSM-SADF. For more details we refer to [4].

IV. PARAMETERIZING SYNCHRONOUS DATAFLOW

In this section we elaborate our concept of SDF parameterization that is the crucial milestone towards the definition of parameterized synchronous dataflow scenarios in Section V.

We draw inspiration from the approach of [8]. In particular, that work discusses parameterization as a meta-modeling technique that enables integration of parameters and run-time adaptation of parameters into a wide range of dataflow models referred to as *base models* that have a well-defined notion of graph iteration. The concept is formally developed in the context of SDF, which resulted in PSDF.

The PSDF concept describes a hierarchical discipline where each hierarchical actor is composed of 3 subgraphs, namely the *init* ϕ_i , the *subinit* ϕ_s and the *body* ϕ_b subgraph.

The *body* subgraph models the main functional behavior of the specification, while the *init* and *subinit* subgraphs control the behavior of the body graph by performing reconfiguration activity.

Thus, PSDF is an implementation-oriented, untimed and architecture independent model.

However, FSM- π SADF we propose in this article is a formalism tailored for studying the performance of streaming applications. Therefore, it would be rather impractical to build FSM- π SADF using the concept of PSDF because the introduction of auxiliary graphs (init and subinit), from the (worst-case performance) analysis perspective unnecessarily increases the model complexity and reduces its intuitive appeal. Furthermore, PSDF is an untimed dataflow model.

Instead, in the context of performance analysis we advocate for a more appropriate SDF parameterization that focuses on the temporal aspect by adding time to the semantics of the model and abstracting away from synthesis related aspects like the parameter delivery mechanisms of [8].

These considerations bring us to the definition of our flavor of parameterized synchronous dataflow or abbreviated π SDF¹, where π SDFG is an abbreviation for π SDF graph.

Definition 6 (π SDFG). *An π SDFG is a tuple $G = (\mathcal{A}, C, \mathcal{P}_i, \mathcal{P}_d, r, d, i, X_G)$, where $\mathcal{A}$ is the set of actors, $C \subseteq \mathcal{A} \times \mathcal{A}$ the multiset of channels, $\mathcal{P}_i$ set of symbolic variables that can take values from to $\mathbb{N}_{>0}$, $\mathcal{P}_d$ set of symbolic variables that can take values from $\mathbb{R}_{\geq 0}$, $r : C \rightarrow \mathcal{R} \times \mathcal{R}$ returns for each channel its (possibly symbolic) source and destination port rates ($\mathcal{R}$ is a context-free grammar that uses elements of $\mathcal{P}_i$ as its terminal symbols), $d : \mathcal{A} \rightarrow \mathcal{D}$ returns for each actor its (possibly symbolic) firing delay ($\mathcal{D}$ is a context-free grammar that uses elements of $\mathcal{P}_d$ as its terminal symbols), $i : C \rightarrow \mathbb{N}_0$ returns for each channel its associated number of initial tokens while X_G is the domain of the graph.*

The structure of Definition 6 defines a parameterization of the structure of Definition 5. In particular, constant rates and actor firing delays of SDFG are in π SDF replaced by parametric expressions generated by grammars $\mathcal{R}$ and $\mathcal{D}$. The parameters appearing in these expressions change at run-time.

Values the parameters can attain are grouped into configurations. More precisely, a complete configuration of a π SDFG is determined by assigning concrete values to all parameters of $\mathcal{P}_i$ and $\mathcal{P}_d$ in contrast to a partial configuration where some parameters are left unspecified. The domain X_G of a π SDFG G is the set of all complete configurations of G and is typically given as a set of constraints (equality and inequality constraints). We denote a configuration of π SDFG G with $x_i^G \in X_G$ where $i \in \{1, \dots, |X_G|\}$. Once a configuration is applied to the π SDFG, an instance of that graph emerges, denoted $\iota_G(x_i^G)$. An instance of a π SDFG is nothing but an SDFG. However, it is important to notice that parameters in configurations cannot be arbitrarily reconfigured without invalidating the static schedule [8]. Furthermore, the running configuration cannot be changed at any time because some rates would not be well-defined [9][5].

To ensure that the parameters attain values that do not invalidate the static scheduling of the instance SDFG, we must at design-time verify that every instance is consistent, deadlock-free and has a valid schedule [8]. This, of course, calls for an enumeration of X_G which incurs compactness issues we wanted to avoid in the first place. To avoid this, we resort to the parametric (symbolic) analysis of [9]. However, for the analysis to be able to decide on these properties we must limit the type of parametric expressions that appear in port rate definitions to products of positive integers ($k \in \mathbb{N}_{>0}$) or symbolic variables ($p \in \mathcal{P}_i$) as follows

$$\mathcal{R} := k \mid p \mid \mathcal{R}_1 \cdot \mathcal{R}_2. \quad (12)$$

As firing delays do not influence the dataflow behavior of the graph, in principle in Definition 6, we could allow for an arbitrary $\mathcal{D}$, but for simplicity and some technical constraints to be explained in Section VIII we restrict $\mathcal{D}$ as follows

$$\mathcal{D} := k \mid k \cdot d \mid \mathcal{D}_1 + \mathcal{D}_2 \mid \mathcal{D}_1 \cdot \mathcal{D}_2. \quad (13)$$

In (13), $k \in \mathbb{R}_{\geq 0}$ and $d \in \mathcal{P}_d$, which renders actor firing delays to be non-negative polynomials. Polynomials are a sound choice as they can and are often used to approximate more complex functions.

To ensure that all rates are well-defined at all times, reconfiguration is constrained to happen only at iteration boundaries of the model, i.e. parameters are allowed to change between iterations of the π SDFG instances. Neunedorffer and Lee [5] refer to points where parameters are allowed to change as *quiescent points*. Note however, that a particular quiescent point is typically smeared in time because π SDFG iterations will typically be overlapped (pipelined) in time. In practice this means that an actor is eligible for reconfiguration as soon as it has completed all its firings within the current iteration regardless of the completion status of other actors in the graph.

Therefore, we say that π SDFG evolves in full SDF iterations of its instances. The instances are determined by configurations.

π SDF paradigm of Definition 6 offers a high modeling flexibility. First, this is due to the fact that rates and firing delays can be expressed using arbitrary expressions defined in parameters of sets $\mathcal{P}_i$ and $\mathcal{P}_d$. Second, this is due to the fact that the concept of π SDFG domain X_G can be used to explicitly represent the complex parameter interdependencies that may arise from a variety of sources [5].

V. INTEGRATION OF PARAMETERIZED SYNCHRONOUS DATAFLOW AND FINITE-STATE MACHINES

In this section, to alleviate the compactness issues of FSM-SADF, we investigate the integration of π SDF and FSMs in

¹We use the Greek letter π in the abbreviation to differ from PSDF of [8].

Fig. 2. FSM-πSADF.

a concept we refer to as parameterized dataflow scenarios. In particular, we model the behavior of an application as a dynamic interaction of parameterized scenarios each of which is modeled by a π SDFG and an FSM. We refer to the new analysis model as FSM-πSADF. From now on, we use the terms parameterized scenario, scenario and (parameterized) scenario π SDFG interchangeably.

We exemplify using the structure of Fig. 2a. The composite FSM-πSADF graph (FSM-πSADFG) in the figure is defined over two parameterized scenarios s_1^P and s_2^P , each of which is modeled by a scenario π SDFG. Sequencing of scenarios is dictated by the parameterized scenario FSM or shortly scenario FSM, where each state corresponds to one scenario. The example scenario FSM has two states: ϕ_1^P and ϕ_2^P . State ϕ_1^P corresponds to s_1^P and state ϕ_2^P corresponds to s_2^P .

The operational semantics of the FSM-πSADF model is as follows. Each reaction/transition of the scenario FSM incurs the execution of one iteration of an arbitrary instance of the scenario π SDFG that the transition destination state corresponds to. Fig. 2b illustrates the operational semantics of the FSM-πSADF of Fig. 2a. Scenario π SDFG domains are depicted as 2-dimensional planes in the $p-q-u$ space (we omit actor firing delay parameters). E.g., every time a transition $\phi_1^P \rightarrow \phi_2^P$ is taken, one iteration of the scenario s_2^P π SDFG is executed. This corresponds to the execution of one iteration of an arbitrary instance of scenario s_2^P π SDFG defined by the configurations found in the $X_{s_2^P}$ hyperplane. FSM-πSADF (like FSM-SADF) allows for pipelined execution of scenarios.

We use the chance to compare the operational semantics of FSM-πSADF to that of FSM-SADF. Notice that the FSM-SADFG of Fig. 1a can be obtained by applying configuration $x^{s_1^P} = \{q = 2, p = 3, a_1 = 5, a_2 = 4, a_3 = 3, a_4 = 4\}$ to s_1^P π SDFG and by applying the configuration $x^{s_2^P} = \{u = 2\}$ to s_2^P π SDFG of the FSM-πSADFG of Fig. 2a. Thus, the illustration of the operational semantics of the considered FSM-SADF of Fig. 1b can be obtained by collapsing the hyperplanes of Fig. 2b into one point. Thus, FSM-πSADF generalizes FSM-SADF as FSM-SADF can be instantiated from FSM-πSADF. In that case, an FSM-πSADFG with only one configuration per scenario is nothing but an FSM-SADFG.

We formally define the new analysis model. First, we define the parameterized scenario FSM in Definition 7.

Definition 7 (Parameterized scenario FSM). *Given a set S^P of parameterized scenarios, a parameterized scenario FSM F^P over S^P is a tuple $F^P = (\Phi^P, \phi_0^P, \delta^P, \Psi^P)$, where Φ^P is the set of states, ϕ_0^P is the initial state, $\delta^P \subseteq \Phi^P \times \Phi^P$ is the transition relation and $\Psi^P : \Phi^P \rightarrow S^P$ is the scenario labeling.*

Thereafter, we expose the definition of FSM-πSADF.

Definition 8 (FSM-πSADF). *FSM-πSADF F^P is a tuple $F^P = (S^P, F^P)$ where S^P is the set of π SDF scenarios and F^P is*

Fig. 3. FSM-πSADF worst-case performance analysis framework. an FSM on S^P .

VI. OUR ANALYSIS FRAMEWORK

After having defined FSM-πSADF in the previous section, we proceed by discussing the main contribution of this article. In particular, this is the three-stage worst-case performance analysis framework for FSM-πSADF depicted in Fig. 3. The analysis is based on the semantics of FSM-πSADF in terms of Max-plus linear system theory and in particular Max-plus automata. The first two stages are crucial as they enable us to alleviate the compactness issues of FSM-SADF by means of parametric characterization and consequent abstraction. In particular, starting from the input FSM-πSADF specification, in the first stage of the flow, we show how to characterize the temporal behavior of each parameterized scenario, i.e. its π SDFG, in isolation using a collection of parameterized Max-plus matrices. This stage is described in Section VII. In the second stage to which we dedicate Section VIII-A, by using the parameterized matrices made available by the first stage, the worst-case behavior of a parameterized scenario is abstracted into a single concrete (with entries in $\mathbb{R}_{max}$) Max-plus matrix. This matrix is called the worst-case evaluation matrix of a scenario. The third and the last stage described in Section VIII-B corresponds to the original FSM-SADF performance analysis step. In particular, we think of the Max-plus matrix abstractions of parameterized scenarios along with the parameterized scenario FSM as scenario matrices and scenario FSM of some implied FSM-SADFG. From here, using the usual FSM-SADF analysis (Max-plus automata-based technique for throughput and the state-space based technique for latency) we derive conservative bounds on worst-case throughput and latency for the input FSM-πSADF.

VII. ANALYZING A SINGLE PARAMETERIZED SCENARIO

After having defined π SDF as the basic modeling primitive of FSM-πSADF, we now discuss Stage 1 of the analysis framework of Fig. 3. In particular, we show how the evolution of a π SDFG in time can be compactly captured by a collection of parameterized Max-plus matrices. Recall that these matrices will be used in the second stage of the analysis flow of Fig. 3 to abstract the worst-case behavior of a parameterized scenario into a concrete Max-plus matrix.

A. Max-plus algebraic semantics of π SDF

A π SDFG evolves in iterations of its instances that are nothing but SDFGs. Therefore, if we assume that instance $\iota_G(x^G)$ defines the $(k+1)$ st iteration of the π SDFG, it follows from the discussion of Section III-E and [32] that the production times of initial tokens after the $(k+1)$ st iteration are given by the timestamp vector

$$\gamma(k+1) = M_{\iota_G(x^G)} \otimes \gamma(k). \quad (14)$$

In (14), $M_{\iota_G(x^G)}$ is the Max-plus matrix of $\iota_G(x^G)$. Therefore, the temporal behavior of a π SDFG G can be fully described by a set of Max-plus matrices of all its instances. However, the number of instances is typically very large and

the run-time needed to derive them is prohibitive (compactness issues). Therefore, we advocate for a single, parametric (symbolic) analysis. In particular, we consider a parametric version of (14) given by

$$\gamma(k+1) = \underbrace{(\mathcal{M}_G^{\text{par}}(x^G))}_{M_{l_G(x^G)}}(x^G) \otimes \gamma(k). \quad (15)$$

In (15), $\mathcal{M}_G^{\text{par}}(x^G)$ denotes a mapping that for each $x^G \in X_G$ returns the associated parametric Max-plus matrix corresponding to a particular $x^G \in X_G$, that when evaluated for that x^G yields the Max-plus matrix of the instance SDFG, i.e. $M_{l_G(x^G)}$. Consequently, (15) reduces to (14). This way, one needs not to perform the enumeration of X_G . The difficulty is moved, however, to determining the mapping $\mathcal{M}_G^{\text{par}}$ defining the collection of parametric matrices as constituents of its codomain. It is a collection (and not a single parametric matrix) because in a parametric (general) setting, the partitioning of X_G occurs naturally due to the max operator in Max-plus. For an SDFG, its Max-plus matrix is determined by symbolic simulation of one iteration of the graph. In particular, the goal of the symbolic simulation is to express timestamps $\{t'_{il} : l \in \{1, \dots, |I|\}\}$ of initial tokens after the $(k+1)$ st graph iteration (that form the vector $\gamma(k+1)$) as a Max-plus scalar product of a vector populated with suitable constants called the *dependency vector* and vector $\gamma(k)$ of initial token timestamps $\{t_{il} : l \in \{1, \dots, |I|\}\}$ after the k th graph iteration as follows

$$t'_{il} = \bigoplus_{i_j \in I} m_{l,j} \otimes t_{ij} = [m_{l,1}, \dots, m_{l,|I|}] \otimes \gamma(k). \quad (16)$$

The dependency vectors then form the rows of the SDFG Max-plus matrix. For more details we refer to Algorithm 1 of [32].

With π SDF, Algorithm 1 of [32] is not directly applicable due to parameters involved, especially the parametric rates that render the channel quantities parametric.

Therefore, we need a parametric flavor of Algorithm 1 of [32] able to express timestamps $\{t'_{ij} : l \in \{1, \dots, |I|\}\}$ as Max-plus scalar products of parametric dependency vectors and $\gamma(k)$. In the remainder of this section we show how to derive this algorithm for a type of graphs that in addition to being consistent, deadlock-free and (quasi-static) schedulable satisfy the following two requirements. Given a π SDFG, let functions $\text{src} : C \rightarrow \mathcal{A}$ and $\text{dst} : C \rightarrow \mathcal{A}$ return for each channel its source and destination actor, respectively. Furthermore, let π_r and π_l denote right and left projection function¹, respectively.

Requirement 1. *For all π SDFG channels $c \in C$ such that $\text{src}(c) \neq \text{dst}(c)$ and $i(c) > 0$, $i(c) > \pi_r(r(c)) \cdot \Gamma(\text{dst}(c))$ must hold, i.e. if c has initial tokens, there must be enough of them for actor $\text{dst}(c)$ to complete all its firings within the iteration.*

With this requirement, we limit our attention to feed-forward structures where initial tokens in graph channels (other than self-edges) are not reproduced more than once within an iteration. This way, in cyclic graphs, feedback loops can be broken resulting in acyclic specifications for the duration of one iteration. Across more iterations, once all these initial tokens have been consumed, cycles are effectively restored. In the context of our Max-plus analysis we impose

¹Let $\mathbf{x} = (x_1, \dots, x_k)$ be an element of the Cartesian product $(X_1 \times \dots \times X_k)$. In that case, $\pi_l(\mathbf{x}) = x_1$ and $\pi_r(\mathbf{x}) = x_k$.

this requirement as it is not clear how to deal with schedule loops of length greater than one [33] with parametric repetition counts. Fortunately, a large class of practical streaming applications fall under this requirement that is typically enforced in literature to enable effective quasi-static scheduling [34][9][8]. However, there are some negative repercussions of Requirement 1 when it comes to modeling of channel buffer sizes that one would normally encode using back-edge initial tokens. In particular, Requirement 1 implies that in graphs under consideration buffers are over-dimensioned, i.e. that enough buffer space has been allocated so that performance is not affected.

Requirement 2. *For all π SDFG channels $c \in C$ such that $\text{src}(c) = \text{dst}(c)$, $i(c) = 1$ must hold.*

This requirement disables the bounding of auto-concurrency. Auto-concurrency of actors can be bounded by inserting a particular number of tokens on their self-edges. With Requirement 2 we allow either full auto-concurrency for an actor or no auto-concurrency at all. This is because we during the process of determining $\mathcal{M}_G^{\text{par}}$ wish to avoid situations where tokens produced by the actor depend on different self-edge tokens from one actor firing to the next. This requirement is not restrictive in practice as any such actor in the graph can be replaced by its latency-rate abstraction [35] that conservatively captures its temporal behavior. We believe the same principle could be straightforwardly applied to cyclic graph substructures with channels not compliant to Requirement 1 using the notion of local iterations [9]. This is however a subject of future work.

B. Max-plus model of π SDF execution

We now proceed by sketching the parametric version of Algorithm 1 of [32] that we use to derive the mapping $\mathcal{M}_G^{\text{par}}$. For a full presentation, we refer the reader to [7].

To render $\mathcal{M}_G^{\text{par}}$, we need to express timestamps of initial tokens after the $(k+1)$ st graph iteration in terms of timestamps of initial tokens after the k th graph iteration (cf. (16)). This means that we need to compute one iteration of the graph in a parametric fashion. This is done by simulating any (quasi-static) schedule of the graph in a self-timed manner where we think of a graph schedule as series of actor firings completing an iteration. Now, initial tokens as any other tokens in the graph are produced as results of particular actor firings. Therefore, their timestamps can be expressed in terms of corresponding actor firing completion times.

In particular, let G be a π SDFG. Furthermore, let $\tau(A_j, n) \in \mathbb{R}_{\max}^{|I|}$ denote the completion time of n th firing of actor $A_j \in \mathcal{A}$ where $n \in \mathbb{Z}$ (nonpositive indices are used to index some past actor firings that produced initial tokens). Then, in accordance with the semantics of self-timed execution of SDF that π SDF inherits:

$$\begin{aligned} \tau(A_j, n) = & \bigoplus_{A_i | (A_i, A_j) \in C} \tau \left(A_i, \left\lceil \frac{n \cdot \pi_r(r((A_i, A_j))) - i((A_i, A_j))}{\pi_l(r((A_i, A_j)))} \right\rceil \right) \\ & \otimes d(A_j). \end{aligned} \quad (17)$$

In particular, the completion time of the n th firing of actor A_j is determined by the largest among the completion times of corresponding firings of its dependencies that produce tokens required by the actor to perform its n th firing (synchronization). This time is then increased by the firing delay of A_j itself (delay).

Fig. 4. Example π SDFG.

Obviously, as follows from (17), a π SDF actor is Max-plus linear and when computing its firing completion times we can use the superposition principle, i.e. consider one dependence at a time and ultimately superpose the contributions stemming from different dependencies.

We exemplify using a minimal but representative π SDF structure of Fig. 4. The structure consists of four actors A_1, A_2, A_3 and A_4 with associated firing delays $a_1, a_2, 0$ and a_4 , respectively. Some of the rates are parameterized using parameters p and q . We say the structure is representative, because it covers all the relevant cases one may encounter when deriving $\mathcal{M}_G^{\text{par}}$ using (17). These cases include the following situations: 1) an input channel to an actor other than a self-edge contains initial tokens, 2) we need to compute a sup-convolution of two linear sequences and 3) an actor has multiple dependencies and to compute its firing completion times we need to use the superposition principle in Max-plus.

Note that the structure complies to Requirements 1 and 2 and attains the following repetition vector $\Gamma(A_1, A_2, A_3, A_4) = (q, p, 1, 1)$ closely related to its quasi-static schedule that takes the form $A_1^q A_4^1 A_2^p A_3^1$. We define $\gamma(k)$ as follows

$$\gamma(k) = [t_{i_1}, t_{i_2}, t_{i_3}]^T, \quad (18)$$

where every initial token timestamp relates to the corresponding firing completion time of the producing actor. In particular,

$$\begin{aligned} t_{i_1} &= \tau(A_1, 0) = [0, -\infty, -\infty] \otimes \gamma(k), \\ t_{i_2} &= \tau(A_2, 0) = [-\infty, 0, -\infty] \otimes \gamma(k), \\ t_{i_3} &= \tau(A_3, 0) = [-\infty, -\infty, 0] \otimes \gamma(k). \end{aligned} \quad (19)$$

We define $\gamma(k+1)$ as follows

$$\gamma(k) = [t'_{i_1}, t'_{i_2}, t'_{i_3}]^T, \quad (20)$$

where every initial token timestamp relates to the completion time of the firing of the producing actor that completes the iteration. Obviously, in this case, this is the firing indexed by the producing actor's repetition vector entry. In particular,

$$t'_{i_1} = \tau(A_1, q), t'_{i_2} = \tau(A_2, p), t'_{i_3} = \tau(A_3, 1). \quad (21)$$

Therefore, to construct $\mathcal{M}_G^{\text{par}}$, we need to compute (21). We do this by applying (17) to actors as they come in the quasi-static schedule. For actor A_1 , via (19) and backward substitution [7] we obtain

$$\tau(A_1, n) = \tau(A_1, n-1) \otimes a_1 = [a_1^{\otimes n}, -\infty, -\infty] \otimes \gamma(k). \quad (22)$$

We can now derive $\tau(A_4, n)$. One of the channels feeding actor A_4 has initial token(s). This is channel (A_3, A_4) . Due to Requirement 1, its repetition vector entry must be non-parametric and all its consumptions from that channel (within an iteration) will refer to those initial tokens, i.e. within an iteration it will never have to wait for the source actor A_3 to perform a firing. Thus is the feedback loop effectively broken and it is enough to compute $\tau(A_4, n)$ for values of n up to

the value of the nonparametric repetition vector entry which in this case equals 1:

$$\begin{aligned} \tau(A_4, 1) &= (\tau(A_1, q) \oplus \tau(A_3, 0)) \otimes a_4 \\ &= [a_1^{\otimes q} \otimes a_4, -\infty, a_4] \otimes \gamma(k). \end{aligned} \quad (23)$$

The case of actor A_2 is more interesting. This actor has two dependencies. Thanks to the superposition principle, we can consider them in isolation. First we consider the dependency defined by channel $c_1 = (A_1, A_2)$ and the self-edge. Actually, when considering any input dependency we must always take into account the self-edge because it captures the state of the considered actor:

$$\tau_{c_1}(A_2, n) = \left(\tau(A_1, \lceil \frac{q \cdot n}{p} \rceil) \oplus \tau(A_2, n-1) \right) \otimes a_2. \quad (24)$$

We treat the recurrence of (24) with backward substitution (for more details we refer to [7]). After rearranging, we obtain

$$\tau_{c_1}(A_2, n) = [\text{conv}(a_1^{\otimes \lceil \frac{q \cdot n}{p} \rceil}, a_2^{\otimes n}), a_2^{\otimes n}, -\infty] \otimes \gamma(k). \quad (25)$$

To derive $\mathcal{M}_G^{\text{par}}$ that can be used in the later performance analysis of the enclosing FSM- π SADF we need dependency vector entries expressed in regular algebra. For (24) and in general this means that we must get rid of the sup-convolution(s) (cf. Definition 4). Now, our actors and graphs are Max-plus linear systems. Therefore, sup-convolutions involved will always involve eventually periodic sequences (cf. Definition 1) and consequently will their results be eventually periodic sequences too. This periodicity is crucial to simplify the analysis into a linear pattern (cf. Definition 2) using the CLUB concept of Definition 3. In particular, Proposition 2 defines means to achieve this by defining a CLUB to the sup-convolution of two linear sequences. If we apply Proposition 2 to (25), we need to split the domain of the considered π SDFG into two exclusive parts and for each of these parts continue the procedure in a separate branch. One where $q \cdot a_1 \geq p \cdot a_2$ and one where $q \cdot a_1 \leq p \cdot a_2$. In this exercise, we proceed with the latter option. In that case, (24) transforms to

$$\hat{\tau}_{c_1}(A_2, n) = [a_1^{\otimes(1+\frac{q}{p})} \otimes a_2^{\otimes n}, a_2^{\otimes n}, -\infty] \otimes \gamma(k). \quad (26)$$

In (26), $\hat{\tau}_{c_1}(A_2, n)$ now denotes a CLUB to $\tau_{c_1}(A_2, n)$. The intuitive interpretation of the result is that if the ratio of the input sequence is smaller than the intrinsic ratio of the actor defined by its firing delay (we may call it the “eigenratio”) then the ratio of the output sequence is equal to the “eigenratio”. Practically, this means that the actor is fed with tokens at a faster rate than it can process them: this is a kind of “low pass” effect.

We proceed by computing the contribution of the dependency $c_2 = (A_4, A_2)$:

$$\tau_{c_2}(A_2, n) = [a_1^{\otimes q} \otimes a_4 \otimes a_2^{\otimes n}, a_2^{\otimes n}, a_4 \otimes a_2^{\otimes n}] \otimes \gamma(k). \quad (27)$$

To derive $\tau(A_2, n)$, i.e. $\hat{\tau}(A_2, n)$ we superpose $\hat{\tau}_{c_1}(A_2, n)$ and $\tau_{c_2}(A_2, n)$ as follows

$$\begin{aligned} \hat{\tau}(A_2, n) &= \hat{\tau}_{c_1}(A_2, n) \otimes \tau_{c_2}(A_2, n) \\ &= [(a_1^{\otimes(1+\frac{q}{p})} \otimes a_2^{\otimes n}) \oplus (a_1^{\otimes q} \otimes a_4 \otimes a_2^{\otimes n}), \\ &\quad a_2^{\otimes n}, a_4 \otimes a_2^{\otimes n}] \otimes \gamma(k) \end{aligned} \quad (28)$$

To get rid of the $\oplus$ operator in the first entry of the dependency vector, we make use of Proposition 1 that defines a CLUB of linear sequences defined by different delay-ratio pairs. In case of (28) the procedure continues in two separate branches, one where $(1 + \frac{q}{p}) \cdot a_2 \geq q \cdot a_1 + a_4$ and the other where

$(1 + \frac{q}{p}) \cdot a_2 \leq q \cdot a_1 + a_4$. In this exercise we proceed with the second option. In that case

$$\hat{\tau}(A_2, n) = [a_1^{\otimes q} \otimes a_4 \otimes a_2^{\otimes n}, a_2^{\otimes n}, a_4 \otimes a_2^{\otimes n}] \otimes \gamma(k). \quad (29)$$

Finally, the iteration completes by a sole firing of A_3 that completes at

$$\begin{aligned} \hat{\tau}(A_3, 1) &= \hat{\tau}(A_2, p) \otimes 0 \\ &= [a_1^{\otimes q} \otimes a_4 \otimes a_2^{\otimes p}, a_2^{\otimes p}, a_4 \otimes a_2^{\otimes p}] \otimes \gamma(k). \end{aligned} \quad (30)$$

By evaluating dependency vectors of (22), (29) and (30) for corresponding values of n (cf. (21)), and organizing them into a matrix in a row-by-row fashion, we can construct the mapping $\mathcal{M}_G^{\text{par}}$. In particular, in our exercise,

$$\hat{\mathcal{M}}_G^{\text{par}}(x^G) = \begin{bmatrix} a_1^{\otimes n} & -\infty & -\infty \\ a_1^{\otimes q} \otimes a_4 \otimes a_2^{\otimes p} & a_2^{\otimes p} & a_4 \otimes a_2^{\otimes p} \\ a_1^{\otimes q} \otimes a_4 \otimes a_2^{\otimes p} & a_2^{\otimes p} & a_4 \otimes a_2^{\otimes p} \end{bmatrix} \quad (31)$$

for $x^G \in X_G \cap (q \cdot a_1 \leq p \cdot a_2) \cap ((1 + \frac{q}{p}) \cdot a_2 \leq q \cdot a_1 + a_4)$. Note that in (31) we use the CLUB notation, i.e. $\hat{\mathcal{M}}_G^{\text{par}}(x^G)$ is a conservative estimate of $\mathcal{M}_G^{\text{par}}(x^G)$ due to use of the CLUB concept during its construction. This is a weakness of the parametric analysis. In particular, although it overcomes compactness issues by avoiding the need to analyze every configuration separately, the analysis is less precise (but conservative). A crucial thing is that, as the CLUB attains the ratio of the original eventually periodic sequence it bounds, the relative estimation error incurred by $\hat{\mathcal{M}}_G^{\text{par}}(x^G)$ will asymptotically go to zero with growing values of repetition vector entries. This speaks in favor of our CLUB as a sound estimation concept. While studying the construction procedure of $\hat{\mathcal{M}}_G^{\text{par}}$ we noticed that it splits the original domain into subregions that we call here natural π SDFG subdomains. Each subdomain is assigned with a parametric Max-plus matrix that all together form the codomain of mapping $\mathcal{M}_G^{\text{par}}$, denoted $\text{cod}(\mathcal{M}_G^{\text{par}})$. The domain of $\hat{\mathcal{M}}_G^{\text{par}}$, denoted $\text{dom}(\hat{\mathcal{M}}_G^{\text{par}})$ is, of course, X_G . The full specification of the algorithm for construction of $\hat{\mathcal{M}}_G^{\text{par}}$ is given in [7]. The algorithm has exponential time complexity in the worst-case. However, practical applications may be expected to have only a few critical parameters while the graph structures can expose sparsity in the sense that there will be no dependencies between many initial tokens in the graph. Furthermore, the definitions of the domains may be such that many exploration paths will be pruned out due to infeasibility. Checking feasibility is a nonlinear constraint satisfaction problem that can be solved numerically using the tool support of [36].

VIII. WORST-CASE PERFORMANCE ANALYSIS OF PARAMETERIZED SYNCHRONOUS DATAFLOW SCENARIOS

In this section we investigate the performance analysis problem for FSM- π SADF specifications. First we define the formal semantics of FSM- π SADF in terms of Max-plus linear system theory and Max-plus automata. Based on this semantics we show how the parameterized matrices of scenarios derived in Stage 1 of the analysis flow of Fig. 3 (cf. Section VII) are used to obtain a Max-plus algebraic abstraction of the worst-case behavior of a parameterized scenario (Stage 2 of Fig. 3). Finally, we show how to use these abstractions to derive worst-case throughput and latency estimates for FSM- π SADF specifications via MCM and state-space analysis, respectively (Stage 3 of Fig. 3).

A. Formal semantics of FSM- π SADF

We proceed by defining the formal semantics of FSM- π SADF. In particular, we describe the execution of a FSM- π SADF as a progression of executions of constituent parameterized scenarios where scenario transitions are captured in the Max-plus automata formalism.

In particular, let

$$\mathbb{M} = \{\mathcal{M}_{s_1^P}^{\text{par}}, \dots, \mathcal{M}_{s_{|S^P|}^P}^{\text{par}}\} \quad (32)$$

be the set of all mappings of that per particular parameterized scenario, given its π SDFG configuration, return the associated parametric Max-plus matrix (recall the semantics of $\mathcal{M}_G^{\text{par}}$ from (15)).

Now, assume that in its $(k+1)$ st iteration an FSM- π SADF runs as instance $\iota_{s^P}(x^{s^P})$ of parameterized scenario s^P . Then, the timestamp vector $\gamma(k+1)$ of the $(k+1)$ st FSM- π SADF iteration is related to the timestamp vector $\gamma(k)$ of the k th iteration as follows

$$\gamma(k+1) = \underbrace{\left(\mathbb{M}(s^P) \right) (x^{s^P})}_{\mathcal{M}_{\iota_{s^P}(x^{s^P})}} \otimes \gamma(k) \quad (33)$$

In (33), mapping $\mathbb{M} : S^P \rightarrow \mathbb{M}$ returns the mapping $\mathcal{M}_{s_i^P}^{\text{par}}$ for a particular $s_i^P \in S^P$. Finally, the matrix of the underbrace of (33) is nothing but a concrete Max-plus matrix of the SDFG instance running as the $(k+1)$ st iteration of the FSM- π SADF.

The recursion of (33) can be used to define the completion time of a sequence $\bar{s}^P = s_1^P, \dots, s_k^P$ of parameterized scenarios that we consider a part of the scenario FSM trace. As the activation of the parameterized scenario entails the activation of its arbitrary instance defined by a configuration (nondeterministic choice), we expand $\bar{s}^P$ as follows

$$\bar{s}^P = (s_1^P, x_1^{s_1^P} \mid \dots \mid x_{|X_{s_1^P}^P|}^{s_1^P}), \dots, (s_k^P, x_1^{s_k^P} \mid \dots \mid x_{|X_{s_k^P}^P|}^{s_k^P}), \quad (34)$$

where bar $\mid$ denotes a nondeterministic choice. Then, the completion time of (34) can be defined as follows

$$\mathcal{A}^P(\bar{s}^P) = \alpha^P \otimes \mu^P(\bar{s}^P) \otimes \beta^P \quad (35)$$

where α^P is the final delay, β^P is the initial delay and $\mu^P : S^{P*} \rightarrow \mathbb{R}_{\max}^{I \times I}$ is a morphism that associates sequences of scenarios $\bar{s}^P$ with Max-plus matrices as follows

$$\begin{aligned} \mu^P(\bar{s}^P) &= \mu^P((s_1^P, x_1^{s_1^P} \mid \dots \mid x_{|X_{s_1^P}^P|}^{s_1^P}), \dots, (s_k^P, x_1^{s_k^P} \mid \dots \mid x_{|X_{s_k^P}^P|}^{s_k^P})) \\ &= \left(\mathbb{M}(s_k^P) \right) (x_1^{s_k^P} \mid \dots \mid x_{|X_{s_k^P}^P|}^{s_k^P}) \otimes \dots \\ &\quad \otimes \left(\mathbb{M}(s_1^P) \right) (x_1^{s_1^P} \mid \dots \mid x_{|X_{s_1^P}^P|}^{s_1^P}). \end{aligned} \quad (36)$$

In the triple $\mathcal{A}^P = (\alpha^P, \mu^P, \beta^P)$ we recognize the Max-plus automaton structure discussed in III-C. The meaning of the initial and final delays is left unchanged. Note that in (36), for the right-hand side matrix product to be well-defined, scenarios may or may not share the same set of initial tokens. If they do not, scenario Max-plus matrices need to be expanded. For more details on how to perform these simple matrix transformations we refer to [37].

The Max-plus automaton structure of (35) with (36) can be used to study the performance of FSM- π SADF in a similar fashion it had been used to study the performance of FSM-SADF because FSM- π SADF is, in essence, a compact representation of FSM-SADF. In particular, in FSM-SADF, the Max-plus automata morphism μ of (10) returns for a scenario the corresponding scenario SDFG matrix and for each scenario there is only one such matrix. With FSM- π SADF, the mapping μ^P for a parameterized scenario returns the Max-plus matrix of an arbitrarily chosen parameterized scenario SDFG instance. Further comparison of the morphisms reveals that (36) can be unfolded into (10) in a way that every parameterized scenario instance would become a scenario in an equivalent FSM-SADF. This follows straightforwardly from the discussion on operational semantics of FSM- π SADF compared to that of FSM-SADF in Section V. Then the equivalent structure could be used to analyze the original parameterized specification for worst-case performance. However, in practice, for systems with high levels of data-dependent dynamics this is typically not possible due to several reasons. First, the run-time needed to characterize all instances of parameterized scenarios using Algorithm 1 of [32] is prohibitive especially if there are many instances the repetition vector entries of which attain large values. In particular, paper [4] reports that Algorithm 1 of [32] scales “more than linear with the repetition vector entries”. Second, Max-plus automata-based techniques for throughput analysis of FSM-SADF rely on the automata product structure which would explode in size due to unfolding. The same effect would incapacitate the use of state-space-based latency analysis techniques. Therefore, we need a more compact analysis. This does not mean we actually give up on the FSM-SADF techniques. Instead, we need to find a way to compact the representation of (36), i.e. remove the need for explicit characterization of every parameterized scenario instance and explicit consideration of all nondeterministic choices implied by a scenario transition. Being interested in worst-case performance, Theorem 2 of [28] will provide a solution to the problem. Recast in the context and terminology of this article, the theorem defines the worst-case abstraction of a parameterized scenario through the concept of worst-case evaluation matrix we define next. Let $s_j^P \in S^P$ be a parameterized scenario. Then, we call the matrix

$$M_{s_j^P}^{w-c} = \bigoplus_{x_i^{s_j^P} \in X_{s_j^P}} \left((\mathbb{M}(s_j^P)) (x_i^{s_j^P}) \right) (x_i^{s_j^P}), \quad (37)$$

the worst-case evaluation matrix of s_j^P . By definition, this matrix is a sup-sum of all scenario instance Max-plus matrices. The intuition behind the result is as follows. Running a particular instance of a parameterized scenario will advance the timestamp vector of FSM- π SADF as given by (33). Now, an entry of the Max-plus instance matrix corresponds to the minimal distance between the production times of two initial tokens between two consecutive FSM- π SADF iterations. Furthermore, an entry of the worst-case evaluation matrix of a parameterized scenario is defined as the supremum over all corresponding entries of all instance matrices. Therefore, as matrix multiplication in Max-plus is monotone (cf. Section III-A), if we in (33) replace particular scenario instance matrices with the worst-case evaluation scenario matrix, the resulting timestmap vector is always a conservative estimate of the actual timestamp vector. Consequently, if worst-case evaluation matrices are used in (36) instead of particular parameterized scenario matrices, $\mathcal{A}^P(\bar{s}^P)$ of (35) will define a conservative bound on the completion time of $\bar{s}^P$. This is

ALGORITHM 1: Worst-case evaluation of a parameterized scenario.

Data: Mapping $\hat{\mathcal{M}}_{s_j^P}^{\text{par}}$

Result: Worst-case evaluation matrix of a parameterized scenario $\hat{M}_{s_j^P}^{w-c}$

```

1 for  $m = 1$  to  $|I|$  do
2   for  $n = 1$  to  $|I|$  do
3      $[\hat{M}_{s_j^P}^{w-c}]_{m,n} = -\infty$ ;
4     foreach  $M^{\text{par}} \in \text{cod}(\hat{\mathcal{M}}_{s_j^P}^{\text{par}})$  do
5       if  $[M^{\text{par}}]_{m,n} \neq -\infty$  then
6          $\text{tmp} = \text{maximize}_{x^{s_j^P}} [M^{\text{par}}(x^{s_j^P})]_{m,n}$ 
7         subject to  $x^{s_j^P} \in \text{dom}(\mathbb{M}(s_j^P))$  s.t. (39)
8          $\hat{\mathcal{M}}_{s_j^P}^{\text{par}}(x^{s_j^P}) = M^{\text{par}}$ 
9          $[\hat{M}_{s_j^P}^{w-c}]_{m,n} = [\hat{M}_{s_j^P}^{w-c}]_{m,n} \oplus \text{tmp}$ 
10      end if
11    end foreach
12  end for
13 end for
```

because

$$\mu^P(\bar{s}^P) \preceq M_{s_k^P}^{w-c} \otimes \dots \otimes M_{s_1^P}^{w-c}. \quad (38)$$

However, this conservative bound is not just any type of conservative bound. In particular, Theorem 2 of [28] shows that the worst-case increases of right-hand sides of both (38) and (36) for growing length of $\bar{s}^P$ are equal. As the worst-case throughput of the system is equal to the inverse of this worst-case increase, the worst-case evaluation matrix abstraction of parameterized scenarios can be used to compute the worst-case throughput of an FSM- π SADFG without the loss of accuracy. For worst-case latency on the other hand, the abstraction will provide a conservative estimate of the exact value. We will discuss this further.

Now we discuss how to obtain the worst-case parameterized scenario evaluation matrices. The definition of (37) calls for an enumeration of the scenario domain where for each configuration we would have to successively apply Algorithm 1 of [32]. However, as mentioned, the run-time will typically be prohibitive due to the domain sizes and possibly large repetition vector entries of SDFGs involved. Instead, we make use of the mappings $\hat{\mathcal{M}}_G^{\text{par}}$ obtained in Stage 1 of the proposed framework of Fig. 3. Recalling that given a parameterized scenario s_j^P , the codomain of $\hat{\mathcal{M}}_{s_j^P}^{\text{par}}$ is a collection of parametric Max-plus matrices, i.e. matrices the entries of which are functions of parameters, computing the sup-sum of (37) equals solving a sequence of optimization problems embedded in Algorithm 1.

The input to the algorithm is the mapping $\hat{\mathcal{M}}_{s_j^P}^{\text{par}}$, while the output is the conservative estimate $\hat{M}_{s_j^P}^{w-c}$ of the actual $M_{s_j^P}^{w-c}$ of (37). Thanks to the properties of the delay-ratio abstraction used in the construction of $\hat{\mathcal{M}}_{s_j^P}^{\text{par}}$ itself, the relative estimation error goes to zero with growing repetition vector entries of the parameterized scenario. Each entry of $\hat{M}_{s_j^P}^{w-c}$ corresponds to the maximal entry among all corresponding maximal entries of parametric matrices defining the codomain of $\hat{\mathcal{M}}_{s_j^P}^{\text{par}}$ (cf. Line 4). These entries correspond to the maximum value an entry of the parametric matrix attains when evaluated for all configurations within the subdomain the matrix is defined in. It is obtained by solving the optimization problem of (39) enclosed in Algorithm 1 where the objective function is the

entry of the considered parametric matrix, i.e. a parametric expression.

The type of optimization problems encountered in (39) depends on the formulations of $\mathcal{R}$ and $\mathcal{D}$ in the definition of the parameterized scenario (cf. Definition 6) as well as on the specification of the scenario domain. With the formulations of (12) and (13) and with regard to Propositions 1 and 2 and the discussion of Section VII, the optimization problem of (39) will fall into the class of polynomial programming problems [38].

B. Performance metrics for FSM- π SADF

We now finalize by presenting Stage 3 of our worst-case performance analysis framework of Fig. 3. In particular, we show how to derive conservative worst-case throughput ($\hat{T}h_{FP}$) and latency ($\hat{L}_{FP}$) bounds for FSM- π SADF specifications via MCM and state-space analysis, respectively. We adopt the worst-case throughput and latency definitions of FSM-SADF from [4]. Worst-case throughput is defined in terms of completed scenarios per time unit and is a scalar. Worst-case latency is defined as a bound on completion time of iterations (scenarios) in terms of production times of initial tokens relative to a periodic reference that is equal to the throughput of the graph [3]. The completion times of iterations (scenarios) are given by timestamp vectors. Therefore, the worst-case latency is itself a vector. This vector shows when at the latest a particular FSM- π SADFG iteration (scenario) with index k can possibly finish with respect to the reference point of time $\frac{1}{\hat{T}h_{FP}} \cdot k$, where, of course, $\frac{1}{\hat{T}h_{FP}}$ is the reference period. A similar definition of latency is used in the work of [39] where it is referred to as *lateness*.

These are conservative worst-case throughput and latency bounds because for determining them, we will be using conservative estimates of worst-case evaluation matrices obtained in Stage 2 of the framework of Fig. 3. The relative estimation error will move towards zero for growing scenario repetition vector entries.

With parameterized scenarios abstracted into worst-case evaluation matrices, to obtain performance indicators for FSM- π SADF we can freely reuse the techniques of FSM-SADF where the worst-case matrices can be thought of as scenario matrices of some implied FSM-SADFG and the parameterized scenario FSM as the FSM of the implied FSM-SADFG. Then, the conservative bound on worst-case throughput of the input FSM- π SADFG will equal the MCM of the so-called throughput graph of the implied FSM-SADFG.

The throughput graph is constructed as follows. Traverse over all scenario FSM states $\phi_u^P \in \Phi^P$ and add a node to the throughput graph for each initial token $i_v \in I$ of the FSM- π SADFG and label the node with (ϕ_u^P, i_v) . Then for every transition (ϕ_r^P, ϕ_s^P) add an edge from node (ϕ_r^P, i_m) to node (ϕ_s^P, i_n) if $[\hat{M}_{\Psi^P(\phi_s^P)}^{w-c}]_{n,m} \neq -\infty$ and set the weight of the edge to $[\hat{M}_{\Psi^P(\phi_s^P)}^{w-c}]_{n,m}$.

We exemplify using the running example FSM- π SADFG of Fig. 2. Assume that the respective scenario π SDFG domains are given as follows

$$\begin{aligned} X_{s_1^P} = & \{p \cdot a_1 \geq q \cdot a_1, p \cdot a_3 \leq q \cdot a_1, q \cdot a_4 \leq p \cdot a_2, \\ & a_4 \leq a_4, q \cdot a_4 \leq p \cdot a_3, \\ & p = w_1 \cdot w_2, w_1 + w_2 = 2 \cdot x_1 - x_2, \\ & p \in [1, 10], q \in [1, 10], w_1 \in [1, 3], w_2 \in [1, 4], \\ & x_1 \in [1, 3], x_2 \in [1, 5], a_1 \in [1, 7], a_2 = 4, \\ & a_3 \in [1, 5], a_4 = 4\}, \\ X_{s_2^P} = & \{u = 30\}. \end{aligned} \quad (40)$$

(a) Throughput graph.

(b) State-space.

Fig. 5. Performance analysis structures.

Domain $X_{s_1^P}$ in (40) entails a number of arbitrary nonlinear constraints. E.g., parametric rate p nonlinearly depends on design environment parameters w_1 and w_2 which in turn depend on parameters x_1 and x_2 . In addition, a default validity interval is given for each parameter, e.g. $p \in [1, 10]$. After running Algorithm 1, we obtain the worst-case evaluation matrices of the respective parameterized scenarios using which we construct the throughput graph which is partially displayed in Fig. 5a. The critical cycle, i.e. the cycle attaining the MCM of the throughput graph is depicted using bold arrows. In particular, the MCM attains the value $(30 + 24)/2 = 27$. The conservative worst-case throughput estimate equals the inverse of the MCM, i.e. $\hat{T}h_{FP} = 1/27$ iterations per time-unit.

At this point one might argue, that given the relatively small cardinality of the scenario domains of (40) one could enumerate the domains and use techniques of FSM-SADF to obtain the worst-case throughput value. This claim holds for the running example, but in practice the respective domains can be vast and enumeration becomes infeasible. Another might argue that instead of computing the worst-case evaluation matrices of parameterized scenarios via Algorithm 1, one could simply construct worst-case SDFG abstractions of scenario π SDFGs by taking parameter upper bounds and subject them to FSM-SADF analysis. In response to this, we argue that such an approach (assuming the case where conservativeness can be guaranteed in presence of parameterized rates) that disregards parameter dependencies typically present in practical designs [5] might incur too much pessimism. For the running example, by merely taking $p = 10$, $q = 10$ and $u = 30$ with $a_1 = 7$, $a_2 = 4$, $a_3 = 5$ and $a_4 = 4$ we can construct a conservative FSM-SADFG abstraction of the input FSM- π SADFG that attains the worst-case throughput value of $1/70$ iterations per time unit which clearly is an over-approximation of the bound derived using our worst-case evaluation matrix-based abstraction.

The remaining performance metric to be discussed is worst-case latency. Using the worst-case evaluation matrices of the parameterized scenarios and the scenario FSM we obtain worst-case latency conservative estimates via the analysis of the reachable part of the state-space of all timestamp vectors $\gamma(k)$. State-space is constructed in a breadth-first search manner from the parameterized scenario FSM. The state itself is a tuple $(\Psi^P(\phi^P), \gamma, w)$ where $\phi^P \in \Phi^P$, γ is a Max-plus timestamp vector which is used to initialize the next scenario execution and w is the state weight. Let tuple $(\Psi^P(\phi^{P'}), \gamma', w')$ define a state that is directly reachable from $(\Psi^P(\phi^P), \gamma, w)$. In that case, $\gamma' = (\hat{M}_{\Psi^P(\phi^{P'})}^{w-c} \otimes \gamma)^{\text{norm}}$ and $w' = \|\hat{M}_{\Psi^P(\phi^{P'})}^{w-c} \otimes \gamma\|$. Continuation of the state-space

construction will eventually result in revisiting an already existing state if the reachable part of the state-space is finite. Sufficient conditions for the finiteness of the state-space are discussed in detail in [7]. The exploration terminates, when there are no more new states. For any path of length k leading to state $(\Psi^P(\phi^P), \gamma, w)$, the actual $\gamma(k)$ of the associated parameterized scenario sequence is given by $T \otimes \gamma$ where T equals the sum of the weights of the path states. Assuming we know the throughput of the graph with a prerequisite that the reachable part of the state-space is finite, we can bound the worst-case latency in a single traversal of state space by finding the smallest vector $\hat{\mathbf{L}}_{FP}$ such that $\gamma(k) \leq \hat{\mathbf{L}}_{FP} + \frac{k}{Th_{FP}}$. This is equivalent to determining the maximum observed value of $\gamma(k) - \frac{k}{Th_{FP}}$ observed. The exploration needs to consider only acyclic paths in the state-space because any cycle in the state-space will not be faster than determined by the throughput and hence will not lead to a larger $\hat{\mathbf{L}}_{FP}$ [3]. We demonstrate this for the running example along the state-space path of Fig. 5b indicated with bold arrows. In particular,

$$\begin{aligned} \hat{\mathbf{L}}_{FP} = & \bigoplus \{ [0, 0, 0, 0, 0, 0]^T, [0, 30, 0, 0, 30, 0]^T - 27, \\ & [24, 54, 35, 62, 62, 30]^T - 54, [24, 84, 35, 62, 84, 62]^T - 81, \\ & [86, 108, 97, 116, 116, 84]^T - 108, \\ & [86, 138, 97, 116, 138, 116]^T - 135 \} = [0, 3, 0, 8, 8, 0]^T. \end{aligned} \quad (41)$$

IX. CASE STUDY

In this section, we demonstrate the application of our parameterized scenario modeling and analysis techniques to a realistic case study from the multimedia domain. In particular, we consider the case of a VC-1 video decoder used in a region of interest (ROI) coding scheme.

A. Experimental setup

ROI coding [40] is a feature of modern video codecs that allows to independently store and transmit a video in a variety of regions of interest. This feature is useful for achieving higher error resilience or for saving bandwidth [40]. Typical way of representing ROIs in a video picture is by the use of a rectangular region that corresponds to a picture slice where slice is a group of macroblocks. We exemplify using the picture from the *Foreman* sequence shown in Fig. 6b. In the sequence, the region of interest is the foreman's face represented by the rectangular "ROI slice", while the background is represented by the "Background slice". The face and the background regions are embodied into separate slices, encoded and stored/transmitted.

In VC-1 coding, three different types of slices are supported: I , i and Ii slices. In an I -slice all macroblocks are encoded in the *Intra* mode. In an i -slice all macroblocks are encoded in the *Inter* mode. In an Ii -slice all macroblocks are both *Intra* and *Inter* coded. The types of slices naturally represent three modes of operation of the decoder shown in Fig. 6a adopted from [41]. Each mode is represented by a different π SDFG according to our parameterized scenario modeling technique. Each π SDFG iteration corresponds to decoding of one slice. Parametric rates p and q denote the number of macroblocks in a slice and the number of blocks within a macroblock, respectively. Actor firing delays (not displayed) are adopted from the profiling results of [41].

We proceed in the context of face detection of Fig. 6b where the decoder's task is to decode a sequence of images where each image is split into two slices, one being the ROI (foreman's face) and the other being non-ROI (background). With

two slice types amenable to processing (ROI and background) and in consideration of the nature of the slices (I , i and Ii), we obtain six decoder scenarios: I_{ROI}^P , i_{ROI}^P , Ii_{ROI}^P , $I_{background}^P$, $i_{background}^P$ and $Ii_{background}^P$. E.g., scenario I_{ROI}^P models the decoding of an I -slice capturing the ROI, i.e. the foreman's face. Relative to the given input frame resolution, the slice sizes expressed in macroblocks (parametric rate p) will differ at run-time depending on the distance of the foreman's face from the capturing device, i.e. the camera.

We assume the ROI can be abstracted into an ellipse of known characteristics, i.e. of known circumference o and eccentricity ϵ where ξ_M and ξ_m are the major and minor axes of the ellipse, respectively. The ellipse abstraction is a natural representation for a face where eccentricity can be thought of as a characteristic of a particular face (some faces are more oval than the others) while the circumference models the distance of the face from the capturing device. The bounding rectangle of the ellipse defines the actual slice to be decoded. These consideration lead to the definitions of respective scenario domains. We exemplify with scenario I_{ROI}^P domain definition (within a picture of $w \times h$ pixels) shown in (42).

$$X_{I_{ROI}^P} = \{ p = (2 \cdot \xi_M \cdot 2 \cdot \xi_m) / (16 \cdot 16), p \in [1, P], \quad (42a)$$

$$q \in [1, 16], \quad (42b)$$

$$p' \geq \mu \cdot P, p + p' \leq P, \quad (42c)$$

$$o^2 = 4 \cdot \pi^2 (\xi_M^2 + \xi_m^2), o \geq O, \quad (42d)$$

$$\epsilon^2 \cdot \xi_M^2 = \xi_M^2 - \xi_m^2, \epsilon = E, \quad (42e)$$

$$2 \cdot \xi_M \leq w, 2 \cdot \xi_m \leq h, \quad (42f)$$

$$a = a_{ref}, b = b_{ref}, c = c_{ref}, d = d_{ref}, \quad (42g)$$

$$e = e_{ref}, f = f_{ref}, g = q_{ref}, h = h_{ref} \} \quad (42h)$$

The number of macroblocks p within the slice is given by the area of the ellipse's bounding rectangle (cf. (42a)). Note that the size of a macroblock is 16×16 pixels. Depending on resolution, the picture/frame consists of maximally P macroblocks (cf. (42a)). The number of blocks within a macroblock q is constrained by (42b). It is known that o is always greater than a certain predefined constant O (cf. (42d)), i.e. O defines the maximal distance from the face to the camera. Furthermore, ϵ is equal to a constant E and the ellipse is entirely contained inside the picture/frame (cf. (42e) and (42f)). Within a picture, it is assumed that the background always occupies the portion μ of the picture/frame comprising p' macroblocks (cf. (42c)). Referent actor firing delays (cf. (42g) and (42h)) were taken from [41] and are expressed in cycles of the STMicroelectronics STxP70 processor.

Slices are sequenced as follows. First, I slices of both ROIs are decoded. This corresponds to the decoding of a complete I picture/frame. Thereafter, a number of i and Ii slices forming i and Ii picture/frames are decoded, respectively. This is first done for the ROI and thereafter for the background. In reality, the number of i frames following I and Ii frames is bounded by the Group of Pictures length. For simplicity, we approximate this conservatively by allowing an arbitrary long sequence of i slices that is always followed by one Ii slice for both ROIs. Finally, the FSM revisits the initial state.

From the case study we see the two-level modeling flexibility our parameterized dataflow scenario concept offers. At the bottom level, within parameterized scenarios, it allows to express data dependent behavior using parameters, i.e. the behavior of a scenario is defined by the values these parameters attain at run-time. These values depend on the characteristics of the input data (the input signal). In the case study, this

Fig. 6. Case study.

is the relative displacement of the tracked object (face) and the camera and the ovality of the face. At the top level, the enclosing FSM is used to specify intricate control logic. In the case study, the control concerns the ordering of different types of slices.

B. Results and discussion

In the exercise, we assume SDTV input format with signal type 480i 16:9 and resolution 720x480 pixels. Thus, $w = 720$, $h = 240$ and $P = 1620$. Furthermore, $O = 700$, $E = 0.6$ and $\mu = 30$. For these values using our performance analysis technique presented in Section VIII we obtain a conservative throughput estimate of $1.78516 \cdot 10^{-7}$ slices per cycle. If we were to use the FSM-SADF techniques using the parameter upper bounds we obtain a throughput estimate of $1.44252 \cdot 10^{-7}$ slices per cycle. The comparison shows that our result tightens the FSM-SADF result by 19.19%. The worst-case latency is bounded by the vector $[0, 0, 0, 0, 0, 0, 9347, 9357, 9357, 0]^T$. For the case study we were indeed able to construct a conservative FSM-SADF abstraction where, per scenario, parameters attained the values of their own upper bounds specified in the scenario domain definitions. The construction was possible because actor firing delays were bounded and because scenario π SDFs attained parametric repetition vectors the entries of which were products of rates (meaning that if a rate increases in value so do the corresponding repetition vector entries). In general, due to reasons explained in Section I, this is not always possible. In particular, we mean the case when repetition vectors have entries expressed as fractions of rates. Then it is not possible to know what are the rate values that result in the worst-case behavior prior to deriving the parameterized matrices and solving the optimization problem(s) of (39). Note that due to consistency requirement, these parameterized fractions at run-time evaluate to integers.

In occurrences when there are no fractional repetition vector entries, we need to differ between two cases. First is the case when parameters are not interdependent. Then the FSM-SADF analysis will yield a tighter estimate. However, in practical models, the values of parameters are often dependent on one another. This is the second case. Here, the parametric analysis will most likely be tighter because the FSM-SADF analysis cannot account for parameter dependencies. However, there may exist corner cases when parameter dependencies permit some (maybe critical) parameters to attain their upper bounds. In this case, the analysis of FSM-SADF due to approximations made in the presented parametric analysis may be tighter. This means that in this case, both techniques can co-exist and the tighter of the two estimates should be taken.

Currently, parametric matrices of parameterized scenarios are manually derived and run-time analyses are not available.

Still, results are encouraging. In particular, if (manual) analysis is practicable for the case study, we believe that results obtained are also representative for other applications with similar characteristics. On the other hand, we verified that enumerative analysis is not practicable at all. This is due to the fact that analysis of instances of the VC-1 decoder parameterized scenarios with large repetition vector entries using Algorithm 1 of [32] implemented in the state-of-the-art SDF³ tool can take up to 80s on an Intel Core i5-750 CPU running at 2.67 GHz with 8GB main memory. The fact that the number of instances is proportional to the cardinality of the product set of parameter ranges and the fact that we need to analyze them all, renders the enumerative analysis infeasible in practice. For the case study, the enumerative analysis would take hours to complete.

X. CONCLUSION

In this article, we have presented a dataflow formalism called FSM- π SADF that overcomes compactness issues of FSM-SADF while retaining its expressiveness. FSM- π SADF combines finite control with parameterized synchronous dataflow as the underlying concurrency model.

In particular, FSM- π SADF adopts the scenario-based modeling abstraction where the execution of an application is interpreted as a sequence of scenarios each modeled by a π SDF graph with the attached domain that captures complex relationships between graph parameters, design environment parameters and (possibly) input signal parameters. The scenario occurrence patterns are given by the scenario FSM. For the model we developed novel techniques for worst-case performance analysis that by working directly with graph parameters avoid the need for enumeration of the respective domains. Furthermore, our analysis guarantees conservativeness, and is in many cases able to produce tighter performance bounds than ones that can be obtained by using conservative FSM-SADF models of input FSM- π SADF specifications if such can be constructed at all (w.r.t. conservativeness).

REFERENCES

- [1] A. Girault, B. Lee, and E. Lee, "Hierarchical finite state machines with multiple concurrency models," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, no. 6, pp. 742–760, Jun 1999.
- [2] E. Lee and D. Messerschmitt, "Synchronous data flow," *Proc. IEEE*, vol. 75, no. 9, pp. 1235–1245, Sept 1987.
- [3] S. Stuijk, M. Geilen, B. Theelen, and T. Basten, "Scenario-aware dataflow: Modeling, analysis and implementation of dynamic applications," in *Proceedings SAMOS*, July 2011, pp. 404–411.
- [4] M. Geilen and S. Stuijk, "Worst-case performance analysis of synchronous dataflow scenarios," in *Proceedings CODES/ISSS*. New York, NY, USA: ACM, 2010, pp. 125–134.
- [5] S. Neuendorffer and E. Lee, "Hierarchical reconfiguration of dataflow models," in *Proceedings MEMOCODE*, June 2004, pp. 179–188.

- [6] M. Skelin, M. Geilen, F. Catthoor, and S. Hendseth, "Parametrized dataflow scenarios," in *Proceedings EMSOFT*. Piscataway, NJ, USA: IEEE Press, 2015, pp. 95–104.
- [7] M. Skelin, M. Geilen, F. Catthoor, and S. Hendseth, "Parameterized dataflow scenarios," Eindhoven University of Technology, Tech. Rep. ESR-2015-01, 2015.
- [8] B. Bhattacharya and S. Bhattacharyya, "Parameterized dataflow modeling for DSP systems," *IEEE Trans. Signal Process.*, vol. 49, no. 10, pp. 2408–2421, Oct 2001.
- [9] P. Fradet, A. Girault, and P. Poplavko, "SPDF: A schedulable parametric data-flow MoC," in *Proceedings DATE*. San Jose, CA, USA: EDA Consortium, 2012, pp. 769–774.
- [10] V. Bebelis, P. Fradet, A. Girault, and B. Lavigueur, "BPDF: A statically analyzable dataflow model with integer and boolean parameters," in *Proceedings EMSOFT*, Sept 2013, pp. 1–10.
- [11] M. Wiggers, M. Bekooij, and G. Smit, "Buffer capacity computation for throughput constrained streaming applications with data-dependent inter-task communication," in *Proceedings RTAS*, April 2008, pp. 183–194.
- [12] M. H. Wiggers, M. J. G. Bekooij, and G. J. M. Smit, "Buffer capacity computation for throughput-constrained modal task graphs," *ACM Trans. Embed. Comput. Syst.*, vol. 10, no. 2, pp. 17:1–17:59, Jan. 2011.
- [13] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete, "Cycle-static dataflow," *IEEE Trans. Signal Process.*, vol. 44, no. 2, pp. 397–408, Feb 1996.
- [14] K. Desnos, M. Pelcat, J.-F. Nezan, S. Bhattacharyya, and S. Aridhi, "PiMM: Parameterized and interfaced dataflow meta-model for mpsoos runtime reconfiguration," in *Proceedings SAMOS*, July 2013, pp. 41–48.
- [15] J. Piat, S. Bhattacharyya, and M. Raulet, "Interface-based hierarchy for synchronous data-flow graphs," in *Proceedings SiPS*, Oct 2009, pp. 145–150.
- [16] B. D. Theelen, M. C. W. Geilen, T. Basten, J. P. M. Voeten, S. V. Gheorghita, and S. Stuijk, "A scenario-aware data flow model for combined long-run average and worst-case performance analysis," in *Proceedings MEMOCODE*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 185–194.
- [17] N. Cossement, R. Lauwereins, and F. Catthoor, "DF*: An extension of synchronous dataflow with data dependency and non-determinism," in *Proceedings FDL*, 2000.
- [18] L. Thiele, K. Strehl, D. Ziegenhein, R. Ernst, and J. Teich, "Funstate: an internal design representation for codesign," in *Proceedings ICCAD*, Nov 1999, pp. 558–565.
- [19] M. Pankert, O. Mauss, S. Ritz, and H. Meyr, "Dynamic data flow and control flow in high level DSP code synthesis," in *Proceedings ICASSP*, vol. ii, Apr 1994, pp. II/449–II/452 vol.2.
- [20] J. Buck and R. Vaidyanathan, "Heterogeneous modeling and simulation of embedded systems in El Greco," in *Proceedings CODES*, May 2000, pp. 142–146.
- [21] W. Plishker, N. Sane, and S. Bhattacharyya, "A generalized scheduling approach for dynamic dataflow applications," in *Proceedings DATE*, April 2009, pp. 111–116.
- [22] W. Plishker, N. Sane, and S. Bhattacharyya, "Mode grouping for more effective generalized scheduling of dynamic dataflow applications," in *Proceedings DAC*, July 2009, pp. 923–926.
- [23] K. Sudusinghe, I. Cho, M. van der Schaar, and S. S. Bhattacharyya, "Model based design environment for data-driven embedded signal processing systems," *Procedia Computer Science*, vol. 29, pp. 1193 – 1202, 2014.
- [24] K. Sudusinghe, Y. Jiao, H. B. Salem, M. van der Schaar, and S. S. Bhattacharyya, "Multiobjective design optimization in the lightweight dataflow for DDDAS environment (LiD4E)," *Procedia Computer Science*, vol. 51, pp. 2563 – 2572, 2015.
- [25] S. S. Bhattacharyya, E. F. Deprettere, and B. D. Theelen, "Dynamic dataflow graphs," in *Handbook of Signal Processing Systems*, S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala, Eds. Springer New York, 2013, pp. 905–944.
- [26] L. H. Wang, C. C. Shen, and S. Bhattacharyya, "Parameterized core functional dataflow graphs and their application to design and implementation of wireless communication systems," in *Proceedings SiPS*, Oct 2013, pp. 1–6.
- [27] S. Lin, L. H. Wang, A. Vosoughi, J. R. Cavallaro, M. Juntti, J. Boutellier, O. Silvén, M. Valkama, and S. S. Bhattacharyya, "Parameterized sets of dataflow modes and their application to implementation of cognitive radio systems," *Journal of Signal Processing Systems*, vol. 80, no. 1, pp. 3–18, 2014.
- [28] S. Gaubert, "Performance evaluation of (max,+) automata," *IEEE Trans. Autom. Control*, vol. 40, no. 12, pp. 2014–2025, Dec 1995.
- [29] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat, "Synchronization and linearity: an algebra for discrete event systems," 2001.
- [30] B. Heidergott, G. J. Olsder, and J. Van Der Woude, *Max Plus at Work: Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. Princeton University Press, 2006.
- [31] C. Ptolemaeus, Ed., *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014.
- [32] M. Geilen, "Synchronous dataflow scenarios," *ACM Trans. Embed. Comput. Syst.*, vol. 10, no. 2, pp. 16:1–16:31, Jan. 2011.
- [33] S. S. Bhattacharyya, E. A. Lee, and P. K. Murthy, *Software Synthesis from Dataflow Graphs*. Norwell, MA, USA: Kluwer Academic Publishers, 1996.
- [34] B. Bhattacharya and S. Bhattacharyya, "Quasi-static scheduling of reconfigurable dataflow graphs for DSP systems," in *Proceedings IWRSP*. IEEE Computer Society, 2000, pp. 84–89.
- [35] M. H. Wiggers, M. J. G. Bekooij, and G. J. M. Smit, "Modelling run-time arbitration by latency-rate servers in dataflow graphs," in *Proceedings SCOPES*. New York, NY, USA: ACM, 2007, pp. 11–22.
- [36] L. Granvilliers and F. Benhamou, "Algorithm 852: Realpaver: An interval solver using constraint satisfaction techniques," *ACM Trans. Math. Softw.*, vol. 32, no. 1, pp. 138–156, Mar. 2006.
- [37] F. Siyoum, M. Geilen, O. Moreira, and H. Corporaal, "Worst-case throughput analysis of real-time dynamic streaming applications," in *Proceedings CODES+ISSS*. New York, NY, USA: ACM, 2012, pp. 463–472.
- [38] H. D. Sherali and W. P. Adams, *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*. Springer Science & Business Media, 2013, vol. 31.
- [39] P. Poplavko, T. Basten, M. Bekooij, J. van Meerbergen, and B. Mesman, "Task-level timing models for guaranteed performance in multiprocessor networks-on-chip," in *Proceedings CASES*. New York, NY, USA: ACM, 2003, pp. 63–72.
- [40] D. Grois and O. Hadar, "Recent advances in region-of-interest coding," *Recent Advances on Video Coding*, pp. 49–76, 2011.
- [41] V. Bebelis, P. Fradet, and A. Girault, "A framework to schedule parametric dataflow applications on many-core platforms," in *Proceedings LCTES*. New York, NY, USA: ACM, 2014, pp. 125–134.

Mladen Skelin is a postdoctoral researcher in the Department of Electrical Engineering at Eindhoven University of Technology. He received his M.Eng. degree in Industrial electronics from the Faculty of Electrical Engineering and Computing, University of Zagreb. He did his PhD studies at the Norwegian University of Science and Technology. His research interests include model-based design and analysis of embedded systems with a special focus on dataflow models of computation.

Marc Geilen is an assistant professor in the Department of Electrical Engineering at Eindhoven University of Technology. He holds an MSc and a PhD from Eindhoven University of Technology. His research interests include formal models of computation, multiprocessor systems-on-chip, networked embedded systems and cyber-physical systems, and multi-objective optimization and trade-off analysis. He is a member of IEEE. He has served on various TPCs and on organizing committees for several conferences including DATE as a topic chair. He co-supervised 6 PhD students.

Francky Catthoor received the engineering degree and a PhD in electrical engineering from the Katholieke Universiteit Leuven, Belgium in 1982 and 1987 respectively. Between 1987 and 2000, he has headed several research domains in the area of high-level and system synthesis techniques and architectural methodologies, including related application and deep submicron technology aspects, and smart photo-voltaic modules, all at IMEC, Heverlee, Belgium. Currently he is an imec fellow. He is part-time full professor at the EE department of the K.U.Leuven. In 1986 he received the Young Scientist Award from the Marconi International Fellowship Council. He has been associate editor for several IEEE and ACM journals, like Trans. on VLSI Signal Processing, Trans. on Multi-media, and ACM TODAES. He was the program chair of several conferences including ISSS97 and SIPS01. He has been elected an IEEE fellow in 2005.

Sverre Hendseth serves as an associate professor in the Department of Engineering Cybernetics at the Norwegian University of Science and Technology. He received his Siv.Ing. (MSc) and Dr.Ing. (PhD) degree in engineering cybernetics from the Norwegian Institute of Technology (NTH) in 1987 and 1995, respectively. His research interests include real-time systems and embedded software.