Microprocessors and Microsystems 52 (2017) 439–460

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Worst-case performance analysis of SDF-based parameterized dataflow☆

Mladen Skelin[a,b,*], Marc Geilen[b], Francky Catthoor[c], Sverre Hendseth[a]

[a] Norwegian University of Science and Technology, 7491 Trondheim, Norway
[b] Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands
[c] IMEC vzw, 3001 Leuven, Belgium

ARTICLE INFO

Article history:
Received 26 December 2015
Revised 26 October 2016
Accepted 7 December 2016
Available online 12 December 2016

Keywords:
Parameterized dataflow
Synchronous dataflow
SDF -based parameterized dataflow
Max-plus algebra
Worst-case performance

ABSTRACT

Dynamic dataflow models of computation (MoCs) have been introduced to provide designers with sufficient expressive power to capture increasing levels of dynamism in present-day streaming applications. Among dynamic dataflow MoCs, parameterized dataflow MoCs hold an important place. This is due to the fact that they allow for a compact representation of fine-grained data-dependent dynamics inherent to many present-day streaming applications.

However, these models have been primarily analyzed for functional behavior and correctness, while the (parametric) analysis of their temporal behavior has attracted less attention.

In this work, we (in a parametric fashion) analyze worst-case performance metrics (throughput and latency) of an important class of parameterized dataflow MoCs based on synchronous dataflow (SDF). We refer to such models as SDF-based parameterized dataflow (SDF-PDF). We show that parametric analysis in many cases allows to derive tighter conservative worst-case throughput and latency guarantees than the existing (nonparametric) techniques that rely on the creation of "worst-case SDF abstractions" of original parameterized specifications. Furthermore, we discuss how by using parametric analysis we can help address the scalability issues of enumerative analysis techniques.

To achieve this, we first introduce the Max-plus algebraic semantics of SDF-PDF. Thereafter, we model run-time adaptation of parameters using the theory of Max-plus automata. Finally, we show how to derive the worst-case performance metrics from the resulting Max-plus automaton structure.

We evaluate our approach on a representative case study from the multimedia domain.

1. Introduction

Dataflow models of computation (MoCs) have proven their value in modeling of streaming applications. Dataflow MoCs are instantiated as dataflow graphs. In such 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*. The essential property of dataflow is that of an actor *firing*. Simply put, actor firing denotes the execution of an actor. Actor firing is an atomic action during which the actor consumes a certain number of tokens from input channels through its input ports, executes some behavior and produces a certain number of tokens at its output ports that are put on its output channels [1]. These token production and consumption numbers are called actor port *rates*. Actors fire according to a set of firing rules which specify what and how many tokens must be available at input ports for the firing to be enabled. In presence of feedback loops, actors in the loop would never become enabled because they depend on each other for tokens. This would lead the graph to a deadlock. Therefore, *initial tokens* must be placed on feedback channels. In timed dataflow under consideration in this paper, actor firing takes a finite amount of time called the *actor firing delay*. Furthermore, port rates are viewed as part of the actor *type signature* along with the type of the tokens [2,3]. Port rates can be used to define a graph *iteration*, or a set of actor firings that has no net-effect on the token distribution of the graph.

Dataflow MoCs have been traditionally divided into two classes: static dataflow MoCs [4] and dynamic dataflow MoCs [5].

☆ Manuscript received December 26, 2015; revised October 26, 2016; accepted December 7, 2016. This work was partly supported by ITEA 3 project 14014 ASSUME.

* Corresponding author.
E-mail address: m.skelin@tue.nl (M. Skelin).

http://dx.doi.org/10.1016/j.micpro.2016.12.004
0141-9331/

Static dataflow MoCs are in wide use due to their predictability, strong formal properties and amenability to powerful optimization techniques [5].

Most well-known representatives of static dataflow are synchronous dataflow (SDF) [6] and cyclo-static dataflow (CSDF) [7]. In SDF, actor type signatures are fixed and known at compile-time. In CSDF, actor type signatures can vary between actor firings as long as the variation complies to a certain type of a periodic pattern.

Static dataflow MoCs owe their "nice" properties to their restricted semantics. This restricted semantics, however, makes them an inadequate tool choice for capturing the dynamic behavior inherent to present-day streaming applications. The need for expressive power beyond that offered by static dataflow MoCs brings us to the class of dataflow MoCs we call dynamic dataflow MoCs. Dynamic dataflow MoCs can be viewed as dataflow formalisms in which actor type signatures (port rates) and actor firing delays for timed models vary in ways not entirely predictable at compile-time [5].

In relation to the concept used to represent the dataflow dynamics, the work of [5] defines two subclasses of dynamic dataflow MoCs. First subclass refers to dataflow MoCs that are developed around an interacting combination of finite-state machines (FSM) and dataflow graphs. Models such as FSM -based scenario-aware dataflow (FSM-SADF) [8] and heterochronous dataflow (HDF) [2] are well-known examples of such FSM /dataflow hybrids where an FSM is used to decouple control from concurrency.

In HDF, each FSM state is mode-refined by a submodel, where each refinement has different actor port rates. In FSM-SADF, each FSM state is associated with an SDF model of a scenario the state corresponds to. This has the effect that across FSM-SADF iterations actors operate in different *modes* or *scenarios*. In different scenarios, actors have different firing delays and port rates.

In the second subclass, a member of which we focus in this paper, dataflow dynamics are represented by alternative means. This is advantageous for the users of design tools that are accustomed to working in the dataflow domain and for which the FSM integration may represent an experimental concept [9].

Examples of such models are Boolean dataflow (BDF) [10], dynamic dataflow (DDF) [10] and parameterized dataflow [5].

In this paper, we are interested in parameterized dataflow as a meta-modeling technique that integrates parameters and run-time adaptation of parameters into a certain class of dataflow MoCs we refer to as *base models* [5]. This way, using parameters, one is able to express fine-grained data-dependent dynamics of present-day streaming applications in a compact way. In particular, we are interested in parameterized dataflow MoCs where SDF serves as the base model. Such models are of special importance, as SDF is considered the most stable and mature dataflow MoC.

Examples are parameterized SDF (PSDF) [9], schedulable parametric dataflow (SPDF) [11], Boolean parametric dataflow (BPDF) [12] and variable rate dataflow (VRDF) [13].

We refer to such models, obtained by parameterization of SDF (in terms of rates and actor firing delays in the timed dataflow context) as SDF -based parameterized dataflow (SDF-PDF). Although such models have been parametrically analyzed for functional behavior and correctness, the parametric analysis of their temporal behavior, in particular analysis of their performance metrics such as throughput and latency, has received far less attention. However, there are remedies to this. In certain cases it is possible to create a "worst-case SDF abstraction" of the original parameterized specification that can be subjected to standard SDF performance analysis techniques [14,15]. The information needed to construct such "worst-case SDF abstraction" would include the upper endpoints of parameterized actor firing delays assuming that these are initially box constrained. The validity of such an abstraction follows from the monotonicity property of SDF [16] that SDF-PDF inherits.

However, using upper endpoints of firing delay parameters will often incur significant amounts of pessimism. E.g., if actors are implemented in software their firing delays correspond to worst-case execution times (WCETs) of associated software modules. It is often the case that these WCETs depend on the module inputs in very complex ways. Paper [17] lists a few examples of applications where WCETs are expressed as polynomial functions of application inputs. Therefore, taking the upper endpoints of default parameter intervals and not considering these dependencies will most definitely incur a significant amount of pessimism which results in a decrease of the optimization margin a designer has at hers/his disposal.

The case of graphs containing parameterized rates is even more complicated, as these necessarily do not influence the temporal behavior of the model in a monotonic way. In particular, an increase in rate value can lead to a decrease in the duration of graph iteration. Things get even worse if these rates show functional dependence on characteristics of the input signal.

A solution to this problem is enumeration, where one would consider all possible parameter value combinations. However, the run-time of enumerative analysis will in many cases in practice be prohibitive due to large spans of values the parameters can attain (compactness issues).

The aforementioned justifies the need for novel worst-case parametric performance analysis techniques that by operating directly on graph parameters remove the need for the touchy construction process of "worst-case SDF abstractions" of original parameterized specifications. Furthermore, we require that the parametric analysis can account for complex parameter inter-dependencies, and so avoid the pessimism the analysis based on "worst-case SDF abstraction" suffers from because it disregards these inter-dependencies and considers only the upper parameter interval endpoints. Finally, by working directly with parameters we remove the need for successive analysis of all parameter value combinations and so we help address the scalability problems the enumerative analysis is prone to.

In this work we present such a worst-case performance analysis framework for SDF-PDF specifications in consideration of certain technical constraints we impose on the input graph structures. Within the framework, we consider self-timed execution of SDF-PDF structures. Self-timed execution is a schedule where every actor fires as soon as possible. The self-timed execution is of special importance as it defines the tightest bound that can be given on the temporal behavior of the system captured by an SDF /SDF-PDF model [16]. We base our approach on the Max-plus algebraic [18] semantics of self-timed execution of SDF that SDF-PDF inherits. We model parameter reconfigurations using the theory of Max-plus automata [19] by exploiting the Max-plus semantic equivalence of SDF-PDF parameter reconfigurations and scenario transitions in FSM-SADF. By subjecting the derived Max-plus automaton structure to appropriate analysis, we are able to derive the relevant worst-case throughput and latency metrics for SDF-PDF.

The remainder of this paper is structured as follows. In Section 2 we illustrate the importance of parameterized dataflow MoCs for modeling applications exposing fine-grained data-dependent dynamics and we outline the performance analysis challenges for such specifications. Section 3 discusses the related work. Section 4 presents preliminary concepts. Section 5 formally defines SDF-PDF followed by Section 6 that develops its Max-plus semantics. Section 7 formally defines the performance metrics of interest and Section 8 presents techniques for computation of those metrics. Section 9 demonstrates the application of our techniques on a realistic-case study from the multimedia domain. Finally, Section 10 concludes.