PiSDF: Parameterized Interfaced Synchronous Dataflow for MPSoCs Runtime Reconfiguration
Karol Desnos, Julien Heulot

To cite this version:
Karol Desnos, Julien Heulot. PiSDF: Parameterized Interfaced Synchronous Dataflow for MPSoCs Runtime Reconfiguration. 1st Workshop on MEthods and TOols for Dataflow PrOgramming (METODO), Oct 2014, Madrid, Spain. 1st Workshop on MEthods and TOols for Dataflow PrOgramming (METODO), proceedings, 2014. <hal-01075114>

HAL Id: hal-01075114
https://hal.archives-ouvertes.fr/hal-01075114
Submitted on 16 Oct 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
PiSDF: Parameterized & Interfaced Synchronous Dataflow for MPSoCs Runtime Reconfiguration

Karol Desnos  
IETR, INSA Rennes, UMR CNRS 6164, UEB  
20 avenue des Buttes de Coësmes  
35708 Rennes, France  
Email: kdesnos@insa-rennes.fr

Julien Heulot  
IETR, INSA Rennes, UMR CNRS 6164, UEB  
20 avenue des Buttes de Coësmes  
35708 Rennes, France  
Email: jheulot@insa-rennes.fr

Abstract—Dataflow models of computation are widely used for the specification, analysis, and optimization of Digital Signal Processing (DSP) applications. In this talk, we present the Parameterized and Interfaced Synchronous Dataflow (πSDF) model that addresses the important challenge of managing dynamics in DSP-oriented representations. In addition to capturing application parallelism, which is an intrinsic feature of dataflow models, πSDF enables the specification of hierarchical and reconfigurable applications. The Synchronous Parameterized and Interfaced Dataflow Embedded Runtime (SPIDER) is also presented to support the execution of πSDF specifications on heterogeneous Multiprocessor Systems-on-Chips (MPSoCs).

I. PARAMETERIZED AND INTERFACED SYNCHRONOUS DATAFLOW (πSDF)

A dataflow Model of Computation (MoC) models an application as a directed graph of computational entities, called actors, that exchange data packets, called data tokens, through a network of First-In First-Out queues (FIFOs) [1]. Synchronous Dataflow (SDF) [1] is the most commonly used dataflow MoC. Production and consumption token rates are fixed scalars in an SDF graph. A static analysis of an SDF graph ensures consistency and schedulability properties that imply deadlock-free execution and bounded FIFO memory needs.

The πSDF MoC [2] is a generalization of the SDF MoC. In addition to the actors and FIFOs of the SDF semantics, the πSDF semantics contains a set of parameters and parameter dependencies that can be used to reconfigure the production and consumption token rates of actors. The πSDF semantics also includes a hierarchy mechanism that enables the composition of graphs by using a πSDF sub-graph as a specification of the internal behavior of an actor.

A πSDF specification of an image processing is presented in Figure 1. The top-level graph of this application contains three actors. The Filter actor is a hierarchical actor whose internal behavior is specified with a sub-graph. Parameters $size$ and $N$ influence the behavior of the application at compile time and at runtime respectively.

II. SYNCHRONOUS PARAMETERIZED AND INTERFACED DATAFLOW EMBEDDED RUNTIME (SPIDER)

The SPIDER runtime [3] is a Real-Time Operating System (RTOS) whose purpose is to map and schedule πSDF graphs on heterogeneous Multiprocessor Systems-on-Chips (MPSoCs). The SPIDER runtime exploits the parallelism and the predictability of πSDF specifications to minimize the latency of applications.

Figure 2 illustrates the operating principle of the SPIDER runtime. The purpose of the Master core is to make mapping and scheduling choices and send jobs to all processing elements, including itself, through FIFOs. On job completion, a core can send new parameter values and monitoring information back to the master core through the Params and Timings FIFOs respectively. A pool of data FIFOs can be accessed by Master and Slave cores to exchange data between jobs.

REFERENCES