Computers and Electrical Engineering 40 (2014) 1858–1873

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Communication storage optimization for static dataflow with access patterns under periodic scheduling and throughput constraint ☆

G.Q. Wang [a,*], R. Allen [a], H.A. Andrade [a], A. Sangiovanni-Vincentelli [b]

[a] *R&D, National Instruments Corp., Berkeley, CA 94704, USA*
[b] *Department of EECS, University of California, Berkeley, CA 94720, USA*

ARTICLE INFO

Article history:
Available online 2 June 2014

ABSTRACT

We address a recently introduced static dataflow model: the Static Dataflow with Access Patterns (SDF-AP) model. For this model we present (1) a generalization of an existing regular periodic scheduling scheme to regular 1-periodic scheduling for flexibility to achieve a smaller schedule period and additional room for optimization on communication storage; (2) a method based on Integer Linear Programming (ILP) to minimize communication buffers under periodic scheduling and user-specified throughput constraints. Experimental results on a set of test cases show that buffer sizes using this approach can be reduced dramatically when compared to the traditional SDF models. The optimal sizing result may serve as an important criterion to evaluate and fine-tune any heuristics-based buffer sizing approach for the SDF-AP model of computation.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

In the Static (or Synchronous) Dataflow (SDF) model of computation [1], data are passed in the form of tokens among actors. Actors perform computation on tokens and in the process they may create new tokens. This model of computation is typically represented as a graph, in which nodes represent actors and edges denote inter-actor communication of data. Edges are also commonly called “channels” connecting actor “ports”. SDF models require that the number of tokens consumed and produced in a single firing (also known as consumption and production rates, respectively) be specified at design time. For instance, a 2:1 decimation filter actor with one input port and one output port would consume two tokens from its input port when firing and produce one result to its output port. When all the rates in an SDF graph are one, it is *homogeneous*; otherwise it is *multi-rate*.

Given that rates are specified statically, SDF models can be analyzed to verify properties such as consistency of production and consumption rates [1] and to ensure that enough communication buffer space is allocated for correct execution (e.g. deadlock freedom for self-timed execution). SDF is inherently an untimed model of computation. It is common, however, to associate execution times with actors to enable static timing analysis. SDF models have been widely used for specifying embedded real-time applications in the domains of multimedia and digital signal processing.

☆ Reviews processed and proposed for publication to Editor-in-Chief by Associate Editor Dr. Jian Li.
* Corresponding author. Tel.: +1 510 225 0622.
E-mail addresses: gerald.wang@ni.com (G.Q. Wang), randy.allen@ni.com (R. Allen), hugo.andrade@ni.com (H.A. Andrade), alberto@berkeley.edu (A. Sangiovanni-Vincentelli).

http://dx.doi.org/10.1016/j.compeleceng.2014.05.002
0045-7906/© 2014 Elsevier Ltd. All rights reserved.

Systems specified using SDF models can be implemented as either software [2,3] or hardware [4,5]. Schedule and throughput vary with the amount of memory allocated for inter-actor communication. Analysis techniques for these models generally overestimate communication storage requirements, which is detrimental for embedded real-time systems where memory is at a premium. The Static Dataflow with Access Patterns (SDF-AP) model of computation was recently introduced to address this deficiency [6]. SDF-AP models specify in addition to token rates the specific cycles (relative to the start of the firing of an actor instance) in which individual tokens are produced or consumed. The potential of saving communication storage using SDF-AP under the same throughput constraint as that of the corresponding SDF is also demonstrated in [6].

SDF models can be executed in a self-timed manner once the required number of tokens are available on its input channels and enough vacant space exists on its output channels to hold the tokens it will produce. FIFO (First In, First Out) buffers are typically used for inter-actor communication. An actor reads its inputs from the respective FIFOs in the order in which the input tokens were produced and outputs its tokens in order to its production FIFOs. Actors can use handshaking to implement the self-timing: an actor stalls until necessary resources (both input tokens and output space) are available, and its neighbors inform it when necessary inputs have been produced or output space has been freed up. Self-timed implementations are common, but it is also possible to statically schedule SDF models.

Whether self-timed or statically scheduled, actors in an SDF model share one characteristic – they do not execute until all necessary inputs are available and all required output space is free. Because it takes time to create tokens and fill output FIFOs, actors are guaranteed to be stalled some portion of the time while one of those two processes is going on. One way to speed up execution (and reduce memory requirements for FIFOs) is to exploit those gaps by pipelining actors' computation by starting work before all input tokens are available and all required output space is free. This is the idea behind SDF-AP. By starting computation when some inputs are available, rather than waiting for all, and by producing tokens when some output space is available, rather than waiting for all required space, actors can overlap computation and memory use more effectively. However, they must also be careful not to consume tokens that are not yet ready or to overwrite previous outputs before they have been consumed. This implies fine-grained scheduling techniques are needed for SDF-AP.

While both exact and approximate buffer size optimization techniques have been studied for SDF models, no detailed buffer sizing technique for SDF-AP models has been reported in the literature. This paper addresses the following question: "How much extra benefit in terms of schedule and buffer size can be gained by exploiting access pattern information?".

This paper makes two contributions: (1) it demonstrates that access pattern can be used to enhance scheduling with existing fine-grained regular periodic scheduling techniques and quantifies the benefits of the additional information; (2) it proposes a new scheduling scheme – regular 1-periodic scheduling – to overcome some of the restrictions inherent in the regular periodic scheduling scheme. The paper presents detailed derivation of scheduling precedence constraints and buffer sizing constraints that guarantee a valid execution. It formulates communication buffer sizing for applications specified in SDF-AP as an optimization problem based on Integer Linear Programming (ILP) [7]. Finally experiments on a set of benchmark applications are presented to illustrate how much communication storage reduction can be obtained when leveraging access patterns. Our ILP-based exact approach can always find the optimal communication storage under a scheduling policy but it is obviously computationally challenging. We believe that fast heuristic buffer sizing techniques for SDF-AP models can be derived and their results can be compared with the exact solution of our approach to evaluate their power and quality.

The rest of the paper is organized as follows. Section 2 surveys existing scheduling and buffer sizing techniques for SDF and SDF-like graphs. Then Section 3 reviews the SDF-AP model, followed by a discussion on throughput and communication storage in Section 4. Next, Section 5 discusses scheduling constraints and buffer size requirements, and presents two periodic scheduling schemes. Details of our exact approach based on ILP to optimize communication storage under throughput constraint for SDF-AP models are presented in Section 6, followed by experimental results in Section 7. Finally Section 8 concludes the paper with conclusions and future work.

2. Background

Buffer sizing and throughput analysis for SDF and SDF-like models has been extensively researched. The system throughput is known to be determined by the implemented schedule, which can be influenced by the amount of communication storage available.

Lee and Messerschmitt [1] introduce Synchronous Dataflow to specify digital signal processing applications for concurrent implementation on parallel hardware. In an SDF model, the number of data samples/tokens produced or consumed on each input and output port by each node on each invocation is specified a priori. Given sample rate information, SDF nodes can be statically scheduled. The same authors present necessary conditions for statically scheduling programs described in SDF graphs onto processors (single or multiple) in [8].

In order to exploit static scheduling and remove expressibility limitations of SDF models, Bilsen et al. [9] present Cyclo-Static Dataflow (CSDF) as a model for system specification. The authors derive the necessary and sufficient conditions for the existence of a static schedule of a CSDF graph and detail techniques for construction of a static schedule.

Stuijk et al. [2] propose a model checking-based approach to trade off throughput and buffer storage for SDF and CSDF models. Based on the charted Pareto space, the minimal buffer space is determined to meet a user-specified throughput

Download English Version:

https://daneshyari.com/en/article/455627

Download Persian Version:

https://daneshyari.com/article/455627

Daneshyari.com