Statistical Optimisations of Asynchronous Dynamic Dataflow

Paulo Garcia, Heriot-Watt University
Robert Stewart, Heriot-Watt University

Pr

CCS Concepts: •**Hardware** → **Reconfigurable logic and FPGAs; High-level and register-transfer level synthesis; Hardware description languages and compilation; Logic synthesis;** •**Software and its engineering** → **General programming languages; Architecture description languages;**

ACM Reference Format:
Paulo Garcia, Robert Stewart. Statistical Optimisations of Asynchronous Dynamic Dataflow *ACM Trans. Architec. Code Optim.* 9, 4, Article 39 (March 2016), 4 pages.
DOI: 0000001.0000001

1. INTRODUCTION

The dataflow model(s) of computation is a paradigm that can be used to implement, represent and reason about myriad processes. Dataflow models processes as concurrent computational blocks (actors) communicating through data (tokens) sent across point-to-point channels. This paradigm is applicable to hardware pipelines, parallel software, distributed systems, etc.; and has been used to implement a broad range of solutions, using one of the many dataflow sub-models, depending on the nature of the application.

The different models of dataflow differ in two key areas: synchronicity and token throughput. *Synchronicity* refers to the timing semantics of different actors: are their actions scheduled according to a global time frame, which can be predicted statically? Or are execution semantics unpredictably variable throughout runtime? *Token throughput* refers to token consumption and production rates: do actors consume/produce a fixed number of tokens per action (or cyclic sequences of tokens), which can be used to reason about global throughput? Or is the throughput unpredictable as well?

There is an inversely proportional relation between predictability and applicability. The simplest models, which exhibit highly predictable execution semantics, can be used to design and represent equally simple systems: e.g., synchronous state-less hardware. The most complex model -dynamic asynchronous- can be used to design and represent virtually any system at any scale (e.g., distributed computing), at the cost of predictability: i.e., it is more complex to reason about its execution semantics, namely timing.

Timing analysis, however, is of paramount importance to *optimisation functions*: manual or automated methods that consume a dataflow network and produce and optimised version. Re-designing or refactoring a dataflow system to optimise a particular metric (e.g., performance, power consumption) requires trustworthy assumptions about the behaviour of that system. State of the art timing analysis methodologies,

This work is supported by
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2016 ACM. 1544-3566/2016/03-ART39 $15.00
DOI: 0000001.0000001

however, are based on formal semantics that rely on static token throughput and synchronisation.

In this paper, we investigate the timing analysis of dynamic asynchronous dataflow and its application to optimisation functions. We abandon the notion of precise timing knowledge, and instead build statistical timing models that can guide optimisation functions. Specifically, this paper offers the following contributions:

— We identify and discuss the limitations of state of the art timing analysis techniques for asynchronous dynamic dataflow, and how these limitations prevent optimisations across a broad range of dataflow implementations.
— We present a methodology for statistical timing analysis based on Probablity Density Functions (PDFs), which can be applied to asynchronous dynamic dataflow.
— We present a methodology that applies PDF analysis to dataflow optimisations, namely scheduling strategies for software implementations and clock gating strategies for hardware implementations.

The remainder of this paper is organized as follow: in Section 2, we revise the different types of dataflow models and the state of the art in dataflow timing analysis, and highlight the current limitations of applying such techniques to asynchronous dynamic dataflow. In Section 3, we present our methodology for statistical timing analysis of asynchronous dynamic dataflow. In Section 4, we describe how our timing analysis can be applied to scheduling optimisations of dataflow software implementations, and in Section 5 we describe how it can be applied to clock gating optimisations of dataflow hardware implementations. In Section 6, we evaluate our approaches on a suite of dataflow applications, implemented on CPU and FPGA. Finally, we present our conclusions and identify future work in Section 7.

2. BACKGROUND AND RELATED WORK

In order to have this paper self-contained, we review three dataflow models: static synchronous, cyclo-static synchronous, and dynamic asynchronous (the interested reader may refer to [Mirza et al. 2014] and [Bouakaz et al. 2017] for a more detailed discussion of other models such as cyclo-static asynchronous). Although other model variants exist, we focus our exposé on these three types, which suffice to illustrate the key aspects of interest in this paper.

— different types of dataflow, focusing on dynamic asynchronous
— related work in timing estimation, token critical path analysis
— examples of why it doesn't work for dynamic asynchronous

3. PDF CRITICAL PATH ANALYSIS

As previously mentioned, the timing analysis of synchronous dataflow is predicated on predictable token consumption/production rates at discrete, globally synchronised points in time. This discrete scheduling may be a global clock transition in hardware implementations or a scheduling time slice in software implementations. On asynchronous dataflow, this discrete abstraction must be abandoned: absolute continuous time must instead be adopted. This is true even in Globally Asynchronous Locally Synchronous (GALS) hardware implementations: even though each actor operates under its own (discrete) clock, clock frequencies may be completely unrelated in frequency and phase. The goal of PDF analysis is to obtain a token throughput timing profile in absolute time, under unpredictable token consumption/production rates. This is done through network profiling, collecting runtime statistics about each actor's behaviour.

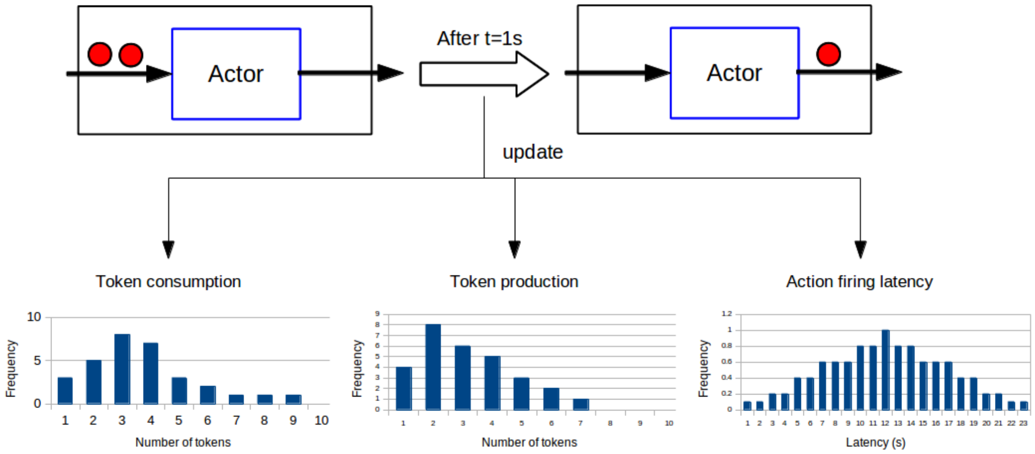

Fig. 1. Profile update after action firing: token consumption is updated by incrementing bin 2, token production is updated by incrementing bin 1, and latency is is updated by incrementing bin 1. Histogram shapes are purely illustrative.

3.1. Single Input Single Output Actors

For the simplest possible dynamic asynchronous actor, with one input and one output port, we monitor how many tokens are consumed and produced per action firing, and how long it takes between successive action firings, assuming sufficient input tokens are always available. It should be noted that we assume these two metrics to be independent: the dynamic nature of actors doen't allow for any assumptions of correlation between token consumption and latency. Fig. 1 depicts an example of such profiling, for different token throughput and action latencies, and examples of histograms for each metric.

In order to model actors' behaviour in absolute continuous time,

- each actor latency (avg of actions) as a PDF
- effect of pipeline analysis of PDF
- effect of feedback loops analysis of PDFs

4. SCHEDULING STRATEGIES

- Assuming round robin scheduling
- Counters to determine where to move to

5. GATING STRATEGIES

- Timers to determine when to gate

6. EXPERIMENTS

7. CONCLUSIONS

ACKNOWLEDGMENTS

We acknowledge the support of the Engineering and Physical Research Council, grant references EP/K009931/1 (Programmable embedded platforms for remote and compute intensive image processing applications) and EP/J015180/1 (Sensor Signal Processing).

REFERENCES

Adnan Bouakaz, Pascal Fradet, and Alain Girault. 2017. A Survey of Parametric Dataflow Models of Computation. *ACM Trans. Des. Autom. Electron. Syst.* 22, 2, Article 38 (Jan. 2017), 25 pages. DOI:http://dx.doi.org/10.1145/2999539

U. M. Mirza, M. A. Arslan, G. Cedersjo, S. M. Sulaman, and J. W. Janneck. 2014. Mapping and scheduling of dataflow graphs: A systematic map. In *2014 48th Asilomar Conference on Signals, Systems and Computers*. 1843–1847. DOI:http://dx.doi.org/10.1109/ACSSC.2014.7094787

Received February 2016; revised March 2016; accepted June 2016