Conservative Abstraction of Dataflow Graphs

Gabriela Breaban, Sander Stuijk, and Kees Goossens

Eindhoven University of Technology, Postbus 513, 5600 MB Eindhoven. Tel: +3140-247 3614

{g.breaban, s.stuijk, k.g.w.goossens}@tue.nl

Abstract

The dataflow model of computation provides useful semantics to model the behavior of real-time embedded systems. Dataflow graphs can be used to compute the performance applications running on a given platform. This document presents a framework to define a conservative abstract model of a given dataflow graph.

The synchronous dataflow model is a restricted version of dataflow. In synchronous dataflow graphs (SDF), the actor production and consumption rates are constant. To perform timing analysis, we assume that the actors have a execution time. When a designer creates an application model he can choose the level of implementation detail of the model. A refined model is more precise and, hence, the performance analysis can give a tighter result. However, adding details increases the size of the model. As a consequence, the analysis takes more time to complete. Thus, there is trade-off between expressiveness and analysability. This motivates us to find a method that reduces the size of a dataflow graph through a conservative abstraction. The resulting abstract model should provide conservative results in terms of performance analysis metrics, such as throughput and latency. Consequently, the abstract model will offer performance guarantees that are no worse than the original detailed model. To give a concrete example, let us consider an application mapped to a multi-core platform. The communication between two tasks mapped to different cores can involve several platform components such as DMA's, network on chip, and arbiters. When creating a binding-aware graph for the application, we need to incorporate all the above mentioned components. A detailed model that can be easily correlated to the hardware behavior would include at least one actor for each component. Figure 1 shows such a binding-aware graph modeling the communication between a source application actor, $input$ and a destination application actor, $output$. Furthermore, if we take into account that such a communication model would be replicated for all the application graph edges that cross different cores, and that this model can be further refined for each of the included hardware components, we can see that the size of the resulting graph can increase significantly.

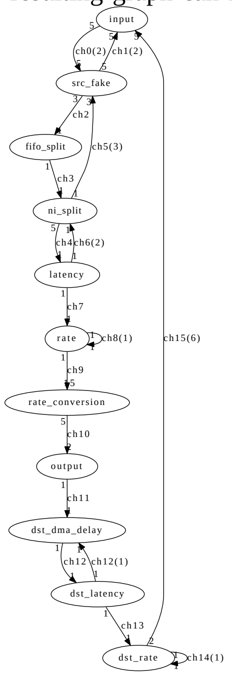
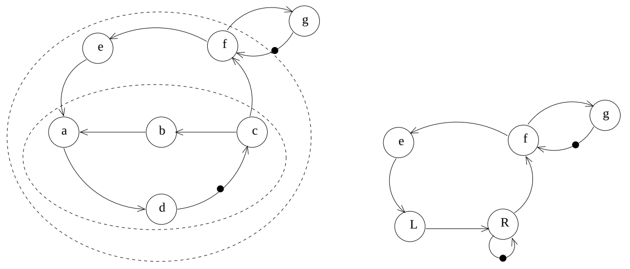


Fig. 1. Example of inter-tile communication model



(a) Example SDF graph (b) Abstracted graph

Fig. 2. Example of input graph and abstracted graph

Our current abstraction technique consists of taking an input SDF graph and selecting a subset of actors and the edges that connect them. For this subset of actors and edges, we define the subgraph that we want to abstract. The result of the abstraction is a chain of two actors, one actor which can fire concurrently followed by a sequential actor. This is called a Latency-Rate model and it has been widely used to model resource arbitration schemes, such as Time Division Multiple Access, Round-Robin, Credit-Controlled Static Priority. It was derived from the Real-Time Calculus formalism which characterizes resources in terms of input and output *arrival curves* and provided and remaining *service curves* [1]. Since the graph H that we abstract, i.e. replace with the Latency-Rate component, is part of another SDF graph, G, which forms the context for H, it follows, implicitly, that we have to preserve the properties of G, such as consistency and liveliness throughout the abstraction.

To illustrate our technique, let us consider the graph shown in Figure 2a. We show two possible choices for the abstraction. The first one selects actors a, b and c and the edges which connect them to define the subgraph to be abstracted, while the second one selects actors a, b, c, d and e and the connecting edges as the subgraph to be abstracted. The resulting abstracted graph will be a copy of the original input graph in which the selected subgraph is replaced with the Latency-Rate model. The result of the abstraction for the example SDF graph for which the innermost subgraph has been selected can be seen in Figure 2b. Furthermore, for the same set of selected actors, we can generate multiple Latency-Rate abstractions that trade latency and throughput while making sure that both metrics remain conservative with respect to the original ones. This means that we can chose different approximations for the latency and throughput and each such pair will impact the analysis time in a different way.

Keywords: Model of Computation; Dataflow Graph; Abstraction

REFERENCES

[1] 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 of the 10th International Workshop on Software &Amp; Compilers for Embedded Systems*, ser. SCOPES '07. New York, NY, USA: ACM, 2007, pp. 11–22. [Online]. Available: http://doi.acm.org/10.1145/1269843.1269846