Towards State-Based RT Analysis of FSM-SADFGs on MPSoCs with Shared Memory Communication

Conference Paper · April 2016

CITATIONS
0

READS
58

4 authors, including:

Maher Fakih
OFFIS e.V.
15 PUBLICATIONS 22 CITATIONS

Kim Grüttner
OFFIS e.V.
81 PUBLICATIONS 217 CITATIONS

Some of the authors of this publication are also working on these related projects:

3Ccar - Integrated Components for Complexity Control in affordable electrified cars View project

SAFEPOWER View project

All content following this page was uploaded by Kim Grüttner on 17 May 2016.

The user has requested enhancement of the downloaded file.
Towards State-Based RT Analysis of FSM-SADFGs on MPSoCs with Shared Memory Communication

Ralf Stemmer†, Maher Fakih*, Kim Grüttnert* and Wolfgang Nebel†
†OFFIS – Institute for Information Technology, Oldenburg, Germany
‡Carl von Ossietzky Universität Oldenburg, Germany

I. INTRODUCTION

Scenario-aware Data-Flow Graphs (SADFGs), first introduced in [3], achieve a good trade-off between expressiveness (allowing the expression of more dynamic behaviors than Synchronous Data-Flow [2] graphs (SDFGs)) and analyzability. Finite-State-Machine Scenario-Aware-Data-Flow (FSM-SADF) graphs are a further simplification of the general SADF graphs. Both extend SDFGs with scenarios. In an FSM-SADF MoC a set of typical scenarios are pre-defined (through a finite state machine) for a specific SDF application. The SDF application reacts to every scenario in a different manner leading to more efficiency and better throughput. Fig. 1a shows an example FSM-SADF of an MPEG decoder as introduced by [5]. The vertices of the graph can be either kernels (VLD, MC, IDCT and RC) or detectors (FD). The solid edges are called data channels while the dashed edges are called control channels. Every input port of the kernels/detector in an FSM-SADF has a consumption rate and likewise each output port has a production rate. In each scenario, kernels can have different production and consumption rates. While general SADF graphs enable multiple scenario changes during one iteration by allowing multiple scenario detectors, FSM-SADF graphs support only one detector per graph which simplifies the analysis of such models. Initial tokens are visualized by dots on the edges. For example the data channel from the kernel RC to the detector FD is initialized by three tokens, so that the detector can be executed three times before the kernel RC can execute to generate new tokens.

In this work, we extend our previous model-checking based real-time analysis approach in [1] for the analysis of timing bounds for FSM-SADFGs mapped on a shared memory multi-core architecture [15]. In our previous work only analysis of SDFGs was supported. We utilize timed automata (TA) as a common semantic model to represent worst-case execution times (WCET) of kernels, detectors and shared communication resources. The analysis model furthermore supports analysis of access protocols for buses, DMA, private local and shared memories of the MPSoC. For given FSM-SADFGs, MPSoC architecture, FSM-SADF to MPSoC mapping, execution & communication times, and scheduling a network of TA can be generated. Using the UPPAAL model-checker, safe timing bounds for FSM-SADF-MoC can be obtained [4]. To avoid non-determinism in the FSM the execution time of the FSM is covered by the best-case and worst-case execution time of the detector. During the write phase, the detector produces control tokens but may also produce data tokens.

Table I: I, {P_0, P_s} are the states of the non-deterministic FSM and become {P_{0/80}, P_{30/99}} after abstracting to a deterministic FSM in Fig. 1b with best-case (BC) and worst-case (WC) rates.

| Rate | Nondet. Scenarios | Det. Scenarios (BC/WC) |
|------|-------------------|-------------------------|
| I    | P_0 | P_s | I | P_{0/80} | P_{30/99} |
| a    | 0   | 0   | 0 | 0 | 0 / 80 | 30 / 99 |
| b    | 0   | 0   | x | 0 | 0 / 80 | 30 / 99 |
| c    | 99  | 1   | x | 99 | 1 / 80 | 30 / 99 |
| d    | 1   | 0   | 1 | 1 | 0 / 1 | 1 / 1 |
| e    | 99  | 0   | x | 99 | 0 / 80 | 30 / 99 |

1 An iteration is the minimum non-zero execution (i.e. at least one kernel has been executed) such that the initial state of the graph is obtained [1].
in Fig. [15] This abstraction is valid since we are interested in getting the worst-case scenario latency of any time critical (such as Period or end-to-end latency). That is why the group of successor states of current state in the non-deterministic FSM can be abstracted in a single state representing worst-case-scenario state among this group. The same procedure could be done for the best-case scenario state. This lead to a new deterministic FSM which describes the token rates in a more coarse way. To estimate which original scenario is the worst and which one is the best, each one has to be analyzed individually as an SDF graph with a static token rate for this specific scenario. Note that also different execution times for every kernel/detector are used according to the current scenario (the execution time values were taken from [3]). This was done with the SDF2TA tool described in [1].

Table 1 shows that the channel b can be either 0 or 80 after the abstraction, while it had multiple rates ranging between 0, 40, 60 and 80 (see FSM in Fig. [15]) before. In Fig. [15] the state F0/80 is equivalent to the original state F80 (see Tab. 1) because in this scenario, the kernels RC and MC (see Fig. [15]) have to be executed most frequently and the highest amount of macro blocks need to be transferred between kernels. To get the best-case values, P0/80 becomes equivalent to the original state P0 (Tab. 1) where most data channels transfer zero tokens and kernels are activated at most once per iteration. Kernels of the FSM-SADF that can work in different scenarios get notified of their scenario by control tokens produced by the detector. In case a kernel needs to be executed multiple times during one scenario iteration, it needs the proper amount of control tokens at its input port that can be consumed during each read phase. Control tokens are propagated from the detector to other kernels using the same shared resources that data tokens use. Therefore control tokens are modeled just like the data tokens only with a different token size.

In our example, we mapped the kernels, the detector and their communication channels implemented as FIFOs on a dual core architecture. The detector FD and the kernels RC and MC are mapped on Core 1 while the kernels VLD and IDCT are mapped on Core 2 as shown in Fig. [1A] The communication channels between kernels on the same core are mapped to a local memory, FIFOs for inter-core communication are mapped to shared memory.

In a first evaluation, we analyzed the worst-case end-to-end latency for the worst and the best scenario as show in Tab. 1 for P0/80 and P30/99. The end-to-end latency of our example was 8928 cycles for the best case scenario selection and 8969 cycles for the worst case scenario.

ACKNOWLEDGEMENT

This work has been partially supported by the CONTREX project, funded by the European Commission under grant agreement FP7-611146.

REFERENCES

[1] Fakhri M, Grünther K, Frenzel M, Rettenberg A (2015) State-based real-time analysis of sdf applications on npsoCs with shared communication resources. J Syst Archit 61(9):486–509, DOI 10.1016/j.sysarc.2015.04.005
[2] Lee E, Messerschmitt D (1987) Synchronous data flow. Proceedings of the IEEE 75(9):1235–1245, DOI 10.1109/PROC.1987.13876
[3] Skelin M, Wognsen ER, Olesen MC, Hansen RR, Larsen KG (2015) Model checking of finite-state machine-based scenario-aware dataflow using timed automata. In: Industrial Embedded Systems (SIES), 2015 10th IEEE International Symposium on, IEEE, pp 1–10
[4] Stuijk S, Ghamarian AH, Theelen BD, Geilen MCW, Basten T (2008) FSM-based SADF. Tech. rep., Eindhoven University of Technology, Department of Electrical Engineering
[5] Theelen BD, Geilen MCW, Basten T, Voeten JPM, Gheorghita SV, Stuijk S (2006) A scenario-aware data flow model for combined long-run average and worst-case performance analysis. In: Proceedings of the Fourth ACM and IEEE International Conference on Formal Methods and Models for Co-Design, 2006, MEMOCODE ’06. Proceedings., IEEE Computer Society, Washington, DC, USA, MEMOCODE ’06, pp 185–194, DOI 10.1109/MEMCOD.2006.1695924, URL http://dx.doi.org/10.1109/MEMCOD.2006.1695924