Journal of Systems Architecture 60 (2014) 553–561

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

End-to-end schedulability tests for multiprocessor embedded systems based on networks-on-chip with priority-preemptive arbitration

Leandro Soares Indrusiak *

Real-Time Systems Group, Department of Computer Science, University of York, York, United Kingdom

ARTICLE INFO

Article history:
Received 24 February 2014
Received in revised form 7 May 2014
Accepted 21 May 2014
Available online 13 June 2014

Keywords:
Network-on-chip
Embedded systems
Hard real-time

ABSTRACT

Simulation-based techniques can be used to evaluate whether a particular NoC-based platform configuration is able to meet the timing constraints of an application, but they can only evaluate a finite set of scenarios. In safety-critical applications with hard real-time constraints, this is clearly not sufficient because there is an expectation that the application should be schedulable on that platform in all possible scenarios. This paper presents a particular NoC-based multiprocessor architecture, as well as a number of analytical methods that can be derived from that architecture, aiming to allow designers to check, for a given platform configuration, whether all application tasks and communication messages always meet their hard real-time constraints in every possible scenario. Experiments are presented, showing the use of the proposed methods when evaluating different task mapping and platform topologies.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

Embedded systems typically have to fulfil timing constraints that are related to their application domain and usage scenarios. Constraints are usually specified as the deadline to perform specific functions. For example, a high-definition video recorder must be able to capture, compress and store 25 video frames per second. In safety-critical applications, such constraints are said to be hard real-time constraints, as there is an expectation that they have to be met by the system in all possible scenarios. Therefore, embedded systems designers must be able to evaluate which design alternatives can fulfil those constraints and, for safety-critical applications, guarantee real-time behaviour.

In this paper, we present analytical methods to evaluate whether a multicore embedded system based on a Network-on-Chip (NoC) can fulfil all its timing constraints. A NoC-based system can have tens to hundreds of processing cores interconnected by an on-chip packet-switching network that allows data to be transferred between the local caches of each core and from/to external memory. Section 2 of the paper provides more detail on this type of system architecture. It will then become clear that the performance of the NoC interconnect is as critical as the performance of the processing cores when it comes to meet timing constraints.

Throughout this paper, we will use the terms *end-to-end timing constraint* or *end-to-end deadline* of an *application task-chain*. Those terms denote constraints derived from the application domain (e.g. every video frame must be processed in 40 ms or less) that must be met by specific components of the application (i.e. chains of communicating tasks). Our goal is to establish whether all task-chains of an application have their end-to-end deadlines met by a particular NoC-based platform configuration, and this problem is referred in this paper as *end-to-end schedulability test*. Such test must consider the *end-to-end latency* of each task of a task-chain: the time it takes for a processing core to execute that task (*computation latency*) plus the time it takes for the NoC to transfer all data produced by that task to the next one on the chain (*communication latency*). In Section 3, precise definitions of all those concepts will be given, followed in Section 4 by formulations of end-to-end schedulability tests that are tailored to NoC-based multicores with priority arbitration.

Some of the schedulability tests presented in this paper are based on classic Response Time Analysis (RTA) [1] and on NoC traffic flow schedulability analysis [2]. Individually, those analyses cannot be used to evaluate and improve the schedulability of a NoC system. For example, the traffic flow schedulability analysis from [2] has been used in [3] to produce fully schedulable task mappings, but authors had to artificially limit the number of tasks mapped to each core, as the analysis does not directly consider the different interference patterns resulting from mapping the source of the traffic flows to different cores. Without a limitation on the maximum number of tasks per core, the mapping optimisation process would lead to solution with all tasks mapped to the same core (so all communications are local, instantaneous and therefore

* Tel.: +44 1904325570.
E-mail address: lsi@cs.york.ac.uk

http://dx.doi.org/10.1016/j.sysarc.2014.05.002
1383-7621/© 2014 Elsevier B.V. All rights reserved.

schedulable). Likewise, the evaluation of NoC schedulability using only RTA would be oblivious to the delays caused by network contention. Therefore, in this paper we discuss how to compose those two analytical methods to achieve correct upper bounds to the end-to-end latency, and show that the resulting analytical model is useful as a test to evaluate whether a specific task mapping is schedulable.

Schedulability tests are not always used in industry and academia. Often, system designers address the schedulability problem by simulating the system under different scenarios and checking if the obtained figures for computation and communication latencies meet the constraints. There are two main limitations to that approach. Firstly, for a complex multicore embedded system, the simulation of a few seconds of an application's execution may take hours or days [4], limiting the number of design alternatives that can be evaluated and the portion of the application lifetime that can be considered. Secondly, simulation can only verify whether constraints are met within the scenarios that are explicitly simulated. In complex embedded systems, the set of possible scenarios is too vast to be exhaustively covered, so it is not possible to check whether constraints are always met. For example, if application tasks can suffer release jitter, it would be necessary to simulate each and every possible value of jitter for each task in order to make sure that the timing constraints are met in every case. In Section 5, we use a number of benchmarks to evaluate the proposed schedulability tests, we compare the obtained figures with those obtained with simulation, and propose a design flow that benefits from the joint use of both techniques.

2. NoC-based multicores

NoCs are a common architectural template for processors with dozens of cores, and it has the potential to scale with the increase of the core count up to hundreds or thousands. Fig. 1 shows a simplified representation of a NoC architecture. It has 16 cores, each of them represented together with their own local cache as a white rectangle. Cores are directly connected to NoC switches (grey rectangles), which route data packets towards a destination (which may be another core, an interface to off-chip memory, a custom hardware accelerator, etc.).

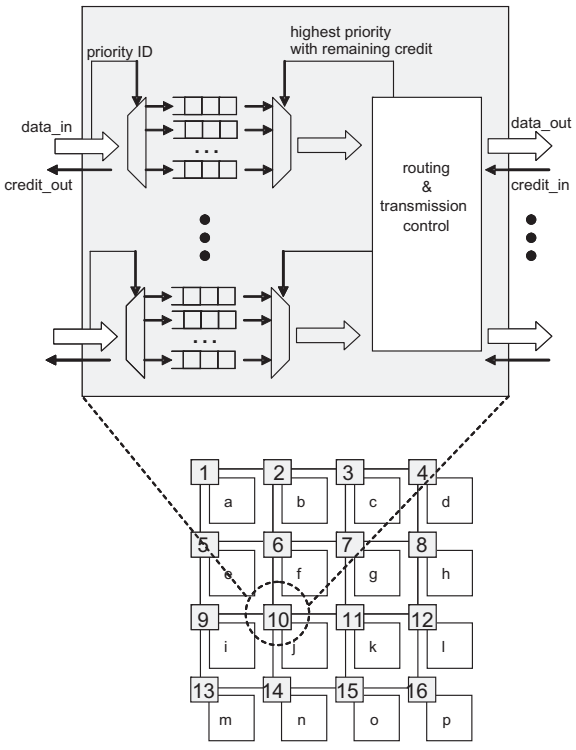

Fig. 1. NoC architecture with detail of the router structure.

Many components of the NoC template can be parameterized to better meet design goals, such as the number and type of cores, buffering, routing and arbitration policies, among others. In this paper, our choice of a specific subset within such a large number of alternatives was based on three criteria: (i) adopt architectural features that are widely used in industry and academia, (ii) use on-chip resources efficiently, and (iii) privilege techniques that are amenable to the type of schedulability tests we are investigating.

Following criterion (i), we concentrate on the widely used 2D mesh topology [5–8]. Criterion (ii) motivates the use of wormhole switching, as its buffer overhead is much smaller than store-and-forward (SAF) approaches, and its link allocation is more efficient than circuit switching approaches: there is no need to reserve the complete path of a packet, and NoC links are only allocated on the segments of the path where there is data ready to be transferred. Finally, criterion (iii) requires some level of predictability on resource sharing policies, so we limit our approach to NoCs with non-adaptive routing and priority arbitration such as QNoC [7] or Hermes [9]. The most common implementation of priority arbitration is based on virtual channels (VCs) [10], which allow packets with higher priority to preempt the transmission of low priority ones, making it easier to predict the outcome of network contention scenarios. Fig. 1 shows a detailed view of a NoC switch with priority-arbitrated VCs: in each input port, a different FIFO buffer stores data words (flits) of packets arriving through different VCs (one for each priority level). The routing component assigns an output port for each incoming packet according to their destination. A credit-based approach [10] guarantees that data is only forwarded from a router to the next when there's enough buffer space to hold it at the right VC. At any time, a flit of a given packet will be sent through its respective output port if it has the highest priority among the packets being sent out through that port, and if it has credits (that is, buffer space on the respective buffer of the neighbouring node connected to that output port). If the highest priority packet cannot send data because it is blocked elsewhere in the network, the next highest priority packet can access the output link.

3. System model and notation

In this paper, we investigate ways to determine whether application tasks executing and communicating over a specific NoC-based multicore can meet all application-specific timing constraints. Therefore, we need a system model that covers the application as well as the NoC-based platform and its configurations.

For the application model, we recall the sporadic task model and define an application as a taskset $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ where each task τ_i is a 6-tuple $\{C_i, T_i, D_i, J_i, P_i, \varphi_i\}$ indicating respectively its worst case computation time, period (i.e. minimum inter-release time interval), deadline, release jitter and priority. The sixth element of the tuple is the only proposed addition to the sporadic task model, and represents a communication message sent by τ_i. Our initial assumption is that each task produces a single message φ_i which is sent immediately after it finishes its computation. The message is defined as a 3-tuple $\{\tau_d, Z_i, K_i\}$ representing its destination task, size and maximum release jitter. A task-chain $X = \{\tau_1, \tau_2, \ldots, \tau_x\}$ is an ordered subset of Γ where each task sends a message to the subsequent task in X, and all of them have the same period T_x. We assume that all task-chains in a particular application Γ are disjoint subsets of Γ, and that loops are not allowed (i.e. the sixth

Download English Version:

https://daneshyari.com/en/article/457729

Download Persian Version:

https://daneshyari.com/article/457729

Daneshyari.com