Microprocessors and Microsystems 37 (2013) 544–553

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

System scenarios-based architecture level exploration of SDR application using a network-on-chip simulation framework

Nikolaos Zompakis [a,*], Alexandros Bartzas [a], Francky Catthoor [b], Dimitrios Soudris [a]

[a] ECE School, National Technical Univ. of Athens, 15780 Zografou, Greece
[b] IMEC vzw, Kapeldreef 75, 3001 Heverlee, Belgium

ARTICLE INFO

Article history:
Available online 12 July 2013

Keywords:
System
Scenarios
NoC
SDR
Framework
Energy consumption

ABSTRACT

Software-defined radio (SDR) terminals are critical to enable concrete and consecutive inter-working between fourth generation wireless access systems or communication modes. Next generation of SDR terminals will have heavy hardware requirements and the switching between the different resource utilization modes will introduce dynamism in respect to timing and size of resource requests. This study exploits the flexibility of a system-level framework, combining a cycle-accurate Network-on-Chip (NoC) simulation environment with an SDR simulator, applying the systems scenario methodology. The aim of our framework is to characterize an SDR platform considering the design trade-offs at an early development phase. The main scope is to define a design space of optimal platform configuration points for a potential SDR resource manager. The purpose is to achieve an efficient resource utilization retaining the reconfiguration cost in reasonable levels. To handle the complexity of the design space, we introduced in the framework the concept of the system scenarios. The system scenario methodology classifies all the run-time situations of the system into cost representative situations. However, this introduces an overhead which is admeasured at the reconfiguration cost. In our case study, we achieve significant resource savings decreasing the energy consumption by 45–73% (based on the system's deadlines).

1. Introduction

Today, nearly all processor chips use multiple cores attempting to deliver higher system performance within their power-constrained environment. The related trend at the embedded systems is the appearance of an increasing number of hardware components on a single chip. The combination of this trend and the power consumption issues led to the rise of the Multi-Processor System-on-Chip (MPSoC) paradigm where multiple Processing Elements (PEs) are linked via a Network-on-Chip (NoC) interconnection network, which manages the on-chip data communication between the PEs and the memories.

It is widely acknowledged that early design decisions have the most significant impact on the final system performance and power consumption [1]. Furthermore, new applications such as Software Defined Radio (SDR) make usage of the additional resources offered by MPSoCs. The traditional approach to design and develop SDR and their successor Cognitive Radio platforms is based on the worst-case scenario of the dynamic software requirements in order to determine the hardware resource characteristics. The SDR system can dynamically switch between the available operation modes to get the optimal quality of the communication service [2,3]. The increased demands on performance combined with an increase in user control and interactivity with the environment significantly impact the resource requests. The latter is becoming more dynamic and event-driven because different wireless protocols (with different symbol rate and performance requirements) must be switched at unknown timing moments.

It has been proved that run-time resource management optimizations can reduce resource requests without affecting significantly the Quality of Service (QoS) and the interaction between user and application [4]. The key for the software developers who design dynamic wireless applications is to incorporate such run-time mechanisms without over-provisioning the resources. These approaches can be classified in:

(1) Reactive techniques: the application is executed, the unused processor cycles are detected and the processor frequency is reduced to take advantage of the saved time.
(2) Proactive techniques: detect in advance the unused cycles and adapt the processor frequency accordingly. One of the most promising proactive design methodologies is "system scenarios" [6].

* Corresponding author.
E-mail addresses: nzompaki@microlab.ntua.gr (N. Zompakis), alexis@microlab.ntua.gr (A. Bartzas), catthoor@imec.be (F. Catthoor), dsoudris@microlab.ntua.gr (D. Soudris).

0141-9331/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.micpro.2013.07.001

The scenario methodology classifies the system behavior from a cost perspective and provides the necessary information for an effective system tuning. The real challenge is to estimate the performance characteristics of the software including the impact of run time mechanisms, at a time when no source code exists.

Essential in designing a dynamic application is the utilization of a hardware–software co-design simulation framework capable of dealing with a combination of hardware, software and run-time models, thus taking into account the various dependencies between hardware and software development and their refinement efforts. A simulation framework designed for exploration of SDR terminals in an early development phase has been proposed in [31], extending the run-time resource management simulator and the hardware component simulator. The development of two automatic wrappers offers automation of the whole simulation and evaluation process of different SDR platforms. In addition the whole evaluation process is performed on a cycle-accurate NoC simulation framework. The first wrapper encapsulates the coarse-grain simulation and automates the exploration of platform parameters design space. The second wrapper establishes an interface layer between the High-Level (HL) SDR coarse-grain platform and Low-Level (LL) cycle-accurate NoC simulation layer. Furthermore, the second wrapper encapsulates the NoC simulation (supporting functions such as the automatic NoC topology exploration) and also implements the automatic transfer, split and mapping of the interconnection traffic dimensioning produced at the higher layer. The aforementioned simulation framework can be configured according to run-time situations with different inputs (i.e., higher that the Data Link OSI layer), different wireless protocols (i.e., Data Link and Physical OSI layers), different run-time resource management policies (i.e., for scheduling, memory management and on-chip communication) and finally for different hardware component configurations.

This paper presents a significant extension of [31] introducing the concept of the system scenarios, classifying, from cost perspective, the behavior of an SDR platform into system scenarios, at an early design phase, using the provided flexibility of the simulation framework for evaluation purposes. More precisely, we focus on the dynamic conditions that an SDR platform can face. The aim is to define a design space of optimal platform configuration points that an SDR resource manager has to take into consideration to achieve efficient resource utilization. This will decrease the added overhead for the implementation of the response mechanisms of the platform, but at the same time we have to ensure the representativeness of these scenarios to not diverge remarkably with these platform situations, which are not taken into account. Since our analysis concentrates on SDR applications with strict QoS requirements, one scenario is the worst case from resource utilization point of view, for the cluster of situations that it represents. This inevitably introduces an overestimation at the estimation of the required system resources at run time, but this is compensated by the decrease of the implementation overhead of the resource manager. More precisely at the case study at Section 6 we achieve a decrease of energy consumption by 45–73% (based on the application deadlines).

The rest of the paper is organized as follows. The related work is outlined in Section 2. The system scenario methodology is presented in Section 3. An overview of the application characteristics is given in Section 4 and the simulation flow is presented in Section 5. The simulation results are presented in Section 6 and finally, the conclusions are drawn is Section 7.

2. Related work

Scenario based design has been used for a long time in different system design areas [7], and especially at the development of the embedded systems [8]. Scenarios describe, at an early design phase the future system functionality including the human–computer interaction. The scenarios appear like narrative descriptions of envisioned usage episodes, and in case of object oriented software engineering like a UML use-case diagram which enumerate, from functional and timing point of view, all possible user actions and the system reactions that are required to meet a proposed system function. These scenarios are called use-case scenarios [7]. In this work, we concentrate on a different kind of scenarios, so-called system scenarios (first introduced in [6]), which characterize the system from the resource usage perspective.

The system scenario methodology has been described in a fully systematic way in [9]. The aim is to capture the data-dependent dynamic behavior inside a thread in order to better schedule a multi-thread application on a heterogeneous multi-processor architecture. Usually, such applications are streaming and have to deliver a given throughput imposing specific time constraints. A design methodology that provides a systematic way of detecting and exploiting system scenarios for streaming applications is presented in [10]. A scenario is defined as the application behavior for a specific type of input data, i.e. a group of execution paths for that particular group of input data. The system scenario concept was also outlined in [11], where the tasks are written using a combination of a hierarchical Finite State Machine with a Synchronous Dataflow model. The disadvantage of that method is that the applications must be written using a limited model, which is a time consuming and error-prone operation.

One of the main challenges, at the embedded system design, is the development of energy-aware techniques. Such techniques adapt the system processing frequency based on the running workload volume with reasonable overhead. An intra-task voltage scheduling mechanism (DVS), which changes at run-time the supply voltage based on the splitting of a task in several slots was proposed in [12,13]. The first DVS approach for hard real-time systems was presented in [14], performing in advance identification of the slack using the combined data- and control-flow information of the program. Its disadvantages are that the data-flow analysis cannot be applied easily outside a procedure, the run-time overhead cannot be controlled and there is no easy way to detect if this overhead leads to increased energy consumption. These limitations can be removed through the proper usage of system scenarios [15], by automatically detecting the parameters with the highest influence on the worst-case execution cycles (WCEC), and using them to define scenarios. Based on the same DVS-aware scheduling algorithm [16], and using real-life multimedia benchmarks, energy reductions between 14% and 52% are obtained, whereas reductions between 4% and 40% are reported in [14]. But these approaches are not very powerful and not suitable for soft real-time systems, as the difference between the estimated WCEC and the real number of execution cycles may be substantial due to the hardware unpredictability and WCEC analysis limitations. To overcome this issue a profiling driven approach to detect and characterize scenarios is employed in [17], deriving and inserting predictors in the application code to predict scenarios at run-time.

Today, many tools have been developed, which put together the hardware and software components (such as the CoCentric System Studio [18], Matlab/Simulink, Metropolis [19], and Chinook [20]). The POLIS project [21] provides a hardware–software co-synthesis tool for design and synthesis of embedded micro-controllers. Additionally, UML has emerged as the de-facto standard for software systems modelling, and specific UML profiles have been developed for real time embedded systems [22]. In our case, our study based on a system-level framework, which combines a cycle-accurate NoC (Network-on-Chip) simulation environment with a pre-existing SDR simulator, thus enabling a cycle accurate simulation and exploration of complex, dynamic hardware/software SDR designs.