Microprocessors and Microsystems 34 (2010) 301–315

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Design and implementation of high-speed buffered crossbars with efficient load balancing for multi-core SoCs

George Kornaros [a,b,*], Theofanis Orphanoudakis [c]

[a] Technical University of Crete, Electronics & Computer Engineering Department, Kounoupidiana, Chania, Crete, Greece
[b] Department of Applied Informatics & Multimedia, Technological Educational Institute of Crete, Stavromenos, Heraklion, Crete, Greece
[c] University of Peloponnese, Telecommunications Science and Technology Department, Karaiskaki Str., 22100 Tripoli, Greece

ARTICLE INFO

Article history:
Available online 17 June 2010

Keywords:
Multi-core
System-on-Chip interconnect
Load-balancing
Hardware dispatcher
Buffered crossbar
Micro-array

ABSTRACT

A large increase of the number of devices integrated in a single chip in conjunction with the significant demands of modern applications for performance has led the designers to a system development methodology based on integrating multiple pre-verified intellectual property cores. Yet, design productivity requirements push designers to focus on key micro-architectural solutions to manage more efficiently the scaling of multi-core SoCs as well as to increase the degree of design automation, particularly as rapid prototyping using reconfigurable computing is becoming mainstream. In this paper we present a novel interconnect architecture based on optimized components to efficiently manage SoCs that follow either a multi-core based approach or are built to support SIMD-style applications that can exploit the processing power of a pool of hardware resources; first we analyze the design of a crossbar featuring shared-memory combined input-crosspoint buffering as a solution for efficient implementation of on-chip interconnection; second we describe the design of a load-balancer featuring configurable proportional allocation of on-chip resources and in-order delivery as a solution for efficient scheduling and execution of processing tasks. The main focus of the paper is to describe and evaluate the mechanisms designed to distribute and manage data transfers so as to implement an efficient interconnection of the integrated cores and control access to available (either on-chip or off-chip) resources for the implementation of a number of embedded systems and applications. Each of these challenges is handled by the proposed architecture in an efficient way in terms of performance, cost in silicon and flexibility.

1. Introduction

Each new generation of semiconductor technology increases significantly the level of integration, functionality, and complexity provided on a single chip. More and more silicon vendors expect heterogeneous multi-core solutions in addition to the traditional multiprocessors, to maximize performance and bandwidth while staying within acceptable power consumption budgets. As process features shrink below 65 nm the main challenge that designers of integrated circuits face is achieving the required functionality, performance and power constraints whilst minimizing design cost and time to market [1]. The key for achieving this is a methodology that moves design from the circuit level to system level, concentrating on the selection of appropriate pre-designed intellectual property (IP) blocks and their interconnection [2,3] into a complete system. Hence, the major concern regarding next-generation embedded systems focuses on their implementation by using System-on-a-Chip (SoC) design methodologies. SoCs are complex embedded devices consisting of many hardware and software IP blocks. As the size and integration level of SoCs grows, the focus is less on the computation, and increasingly on communication, while their performance scalability requires in turn sophisticated interconnection network architectures. Communication-centric design approaches for multi-core SoCs are getting more consideration and thus Networks-on-Chip (NoCs) have emerged as an alternative interconnect solution, where the SoC is regarded as a network of components. The design complexity of SoCs is expected to grow proportionally to the number of embedded IP cores and to the required degree of interaction between these cores in order to execute complex applications with dynamically varying demands. Design approaches should therefore focus on generic adaptable methodologies that can be applicable in different application domains and consider next-generation SoCs mainly as networks of multiple nodes each representing sources generating data and requests for processing tasks and/or on-chip resources of multiple

* Corresponding author at: Technical University of Crete, Electronics & Computer Engineering Department, Kounoupidiana, Chania, Crete, Greece.
E-mail addresses: kornaros@mhl.tuc.gr (G. Kornaros), fanis@uop.gr (T. Orphanoudakis).

0141-9331/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2010.06.002

forms e.g., processing, storage, transmission etc. that are available to all other on-chip nodes. Therefore, the contribution of this paper focuses on NoC architectures and specifically on the efficiency of the interconnection architecture.

In this paper we present such a novel interconnection architecture basically comprising two building blocks: (i) a shared-memory combined input-crosspoint buffered crossbar, which can provide efficient centralized communication across all cores and (ii) an arbiter controlling access to internal resources, which can be used for load balancing and efficient resource utilization. We show that these components can be efficiently used either independently or in an integrated fashion to build arbitrarily complex SoC designs and we demonstrate their use in an example embedded application in the area of genomics/bioinformatics. The objective of this work is not to describe a specific SoC multi-core design but rather to demonstrate a practical implementation of the proposed interconnection architecture, analyze the complexities a designer of such a system has to cope with and the choices we made and finally to assess and demonstrate the performance that this architecture can achieve in a large number of applications and designs, which are growing rapidly as SoCs are becoming popular. Hence, in order to put our work in perspective we first review the potential range of applications and their basic requirements and in the following sections we review related work, describe the details of the interconnection architecture and evaluate its performance through simulation results as well as an actual experimental prototype used as a case study of the proposed design.

A main motivation behind SoC designs is that cost, in silicon, energy, and complexity, of making a single processing unit run a single instruction stream ever faster has eventually reached a limit imposed by the physical limitations of circuit technology. The trend today for pure performance is fortunately following a different approach stemming from the varying degree of concurrency of most applications; today's emerging multi-core systems are capable of sharing work and executing tasks on independent execution cores concurrently. If the concurrent functions of a SoC cannot be statically decomposed at system design time, an alternative approach is to build a coherent symmetric multiprocessing (SMP) cluster of processor cores; within such a cluster, multiple processing units are available as a pool to run the available tasks, which are assigned to processors "on-the-fly". The price to be paid for this flexibility is that it requires a sophisticated interconnect between the cores; scaling to tens or even hundred of cores forces the interconnect requirements to be relatively large and of high-bandwidth. This negates the area and power advantages alluded to above for functionally partitioned multi-core systems, but can still be a good tradeoff. Internally, an on-chip interconnect could comprise a single or multi-stage fabric, or independent set of fabrics, each with its own properties. At the top-level however, the programming interface is desirable to allow for a uniform access to each pool of resources.

The foremost features of a future interconnect architecture supporting multi-core single chip designs should include:

- An abstraction layer at the top level that allows developing applications independent of the number and capacity of the leaf resources.
- Dispatchers that dynamically allocate and de-allocate resources according to criteria imposed by system (i.e. energy constraints), or by the run-time needs of an application.
- A scalable and flexible interconnect scheme capable to provide independent paths or partitions that operate non-intrusively and non-blocking.
- Run-time monitoring of the system for power management, reliable operation, or run-time functional reconfiguration.

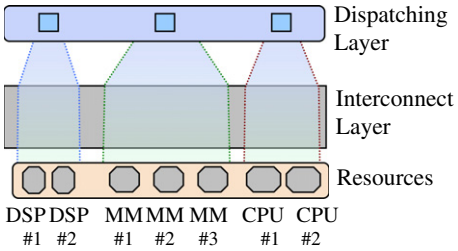

Fig. 1. Abstract organization of a multi-core System-on-Chip. An instance of mapped resources in time is depicted; depending on the application or run-time requirements the assignment and usage of interconnect fabric and resources may change dynamically in time.

Fig. 1 shows an example organization of a multi-core SoC abstracting a desirable computation model of today's embedded applications. The on-chip nodes are organized with respect to their functionality as discussed above. Additionally, we also identify a layered architecture distinguishing the role of on-chip components responsible for managing and efficiently executing the exchange of data and the completion of the intermediate processing steps in order to run a multitask or multithread application.

The example of a system instantiation depicted in Fig. 1 above includes three memory management units (MM) for moving large amounts of data to and from peripheral memory modules and two sets of processing units, one for general purpose processing (CPU) and one for digital signal processing (DSP). To this end multiple resources could be the attached as leaf nodes tailored to specific application requirements as long as the upper layers provide enough performance, flexibility and manageability at low cost.

Given the above issues we describe an innovative architecture that addresses the requirements of the dispatching and interconnect layers (as depicted in Fig. 1) of multi-core SoCs. The contribution of this paper is twofold: first we describe the design of a crossbar featuring shared-memory combined input-crosspoint buffering as a solution for efficient implementation of on-chip interconnection; second we describe the design of a load-balancer featuring configurable proportional allocation of on-chip resources and in-order delivery as a solution for efficient scheduling and execution of processing tasks and implementation of complex embedded applications with stringent requirements for real-time execution with higher performance. These components can be considered as standalone IP blocks that can be integrated in the overall SoC design and can be used in conjunction or independently.

The rest of the paper is organized as follows. Since the proposed on-chip interconnect aims to address the basic requirements that stem from the latest SoC design trends, we briefly review in Section 2 the basic properties of modern SoC designs in order to better identify the different types of IP blocks and their interconnection requirements as well as related work regarding interconnection architectures. Section 3 details analytically the contribution of this paper describing the design architecture and its implementation. Specifically, Section 3.1 describes the integrated view of the interconnection architecture, Section 3.2 presents the internal architecture of the buffered crossbar with shared buffering and Section 3.3 describes the organization of the dispatcher. In Section 4 we evaluate the performance of the above components in terms of several figures of merit and several experimental and simulation results are given to validate their performance in terms of implementation cost, latency, throughput, load-balancing and in-order delivery features. Section 5 demonstrates an application on a prototype developed based on the architectural components analyzed previously and discusses the results of the implementation of the proposed architecture in this case study. Finally, Section 6 offers the concluding remarks.