Modeling Communication in Multi–Processor Systems–on–Chip Using Modular Connectors

Leonidas Tsiopoulos, Åbo Akademi University, Finland
Kaisa Sere, Åbo Akademi University, Finland
Juha Plosila, University of Turku, Finland

ABSTRACT

Formal methods of concurrent programming can be used to develop and verify complex Multi–Processor Systems–On–Chip in order to ensure that these systems satisfy their functional and communication requirements. The authors use the Action Systems formalism and show how asynchronous communication of Multi–Processor Systems–on–Chip can be modeled using generic connectors composed out of simple channel components. The paper proposes a new approach to modeling generic and hierarchical connectors for handling the complexity of on–chip communication and data flow. The authors’ goal is to avoid overloaded bus–based architectures and give a distributed framework. A case study presents the authors’ modeling methodology.

Keywords: Action Systems, Asynchronous Communication, Composition, Formal Methods, Multi–Processor Systems–On–Chip

1. INTRODUCTION

As the technologies improved and the electronic systems became increasingly complex with many independent modules communicating with each other on a single PCB board, a need for a well defined inter–communication scheme arose. Several manufacturers introduced so called bus–based System–on–Chip (SoC) communication frameworks in order to deal with the increased complexity. One of such products is the AMBA 2 AHB interface bus (ARM, 2005).

The common feature of these frameworks is that there is a bus to which master modules, such as processor cores and DSPs, must connect in order to access slave modules such as memories. In such bus–based SoC architectures the bus is a shared communication resource with the masters competing for the access to the slaves. Because it is a shared resource, it can become overloaded quickly, especially when several master modules are connected to it (Lahiri, Raghunathan, & Dey, 2001).

In order to overcome the single bus overloading, bus matrix–based architectures (Nakajima, Yamamoto, Ozaki, Sezaki, Kanakagi,
Furuzono, Sakamoto, Aruga, Sumita, Tsutsumi, Ueda, & Ichinomiya, (2002) appeared consisting of several parallel buses to offer the bandwidth required for complex SoCs. The main drawback of this approach is that every master is connected to every slave in the system, resulting in a very large number of buses. To overcome this disadvantage, approaches for creating a partial bus matrix for a given application have been proposed (Pasricha, Ben–Romdhane, & Dutt, 2007) in order to reduce complexity and power consumption as well as increase efficiency. During the same time, the Network–on–Chip communication paradigm (Hemani, Jantch, Kumar, Postula, Öberg, Millberg, & Lindqvist, 2000) has been introduced offering very high communication bandwidth and network scalability through its regular structure. Depending on the requirements and priorities of a given application (high communication bandwidth, latency, reduced power consumption, etc) a system designer can choose a suitable communication scheme out of the aforementioned ones.

Multi–Processor Systems–on–Chips (MP–SoC) are used in many important applications, from home use electronics and embedded systems to large control and data transfer systems. Hence efficient methods are needed in order to specify, model the communication and verify their design. Formal methods of concurrent programming, such as CSP (Hoare, 1985), Petri Nets (Desel & Juhás, 2001) and Action Systems (Back & Kurki – Suonio, 1983) can be used to design these systems. Action Systems, the modeling language of this paper, provides a rigorous framework for MPSoC design (Back, Martin, & Sere, 1995; Plosila, Seceleanu, & Sere, 2004) and it is sufficient for the work presented here. There is adequate tool support from the B Method (Abrial, 1996), by converting Action Systems to B Action Systems (Waldén & Sere, 1998), as well as the notion of stepwise refinement (Back & Sere, 1996) is built in the framework which allows an abstract system to be refined to a more concrete one preserving its original functionality. Note that refinement is out of the scope of this paper.

In this paper we present a well structured way to model communication infrastructures in heterogeneous MPSoCs and focus on formal specification and modeling. We propose a development method where we create the specification in a component–based and hierarchical manner. We have a Globally Asynchronous Locally Synchronous (GALS) systems point of view (Chapiro, 1984) where each computation and storage unit can be synchronized to a private local clock but their communication is established asynchronously eliminating the need for a global system clock. Within our framework modular connectors are composed out of asynchronous communication channels relying on the request and acknowledgement phases of the asynchronous communication. The needed arbitration is achieved within the composed connectors depending on the channels’ functionality, their topological placement and the connector’s own logic.

Nowadays MPSoCs are heterogeneous and may consist of many different regions requiring different communication infrastructures, thus generic and correct–by–construction connectors can be used to manage the complexity and fulfill the communication requirements. We believe that the absence of complicated non–modular and non–hierarchical arbiter modules will help in reducing the cost (in time and in money) of developing MPSoCs. The already proved functionality of the subsystems aids component reuse and helps to reduce the complexity of MPSoC arbitration.

The structure of the rest of the paper is as follows. In Section 2 we give an overview of the Action Systems formalism following from the work presented in (Tsiopoulos, Sere, & Plosila, 2006). We formally define what a connector is within our approach and we give a theorem showing when a connector is internally consistent. In Section 3 we formally specify some basic asynchronous channels and we present how to compose modular connectors. In Section 4 we give an example in order to present how the asynchronous connectors of Section 3 can interconnect MPSoC modules. In Section 5 we present some related work. Finally, in Section