Microprocessors and Microsystems 45 (2016) 141–150

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Configurable network-on-chip router macrocells

Sergio Saponara*, Luca Fanucci

Department of Information Engineering, University of Pisa, Pisa, Italy

ARTICLE INFO

Article history:
Received 23 October 2015
Revised 19 March 2016
Accepted 22 April 2016
Available online 29 April 2016

Keywords:
Network-on-chip (NoC)
Multi-processor system-on-chip (MPSoC)
Router
Configurable core
Design methodology

ABSTRACT

This paper presents a configurable architecture for Network-on-Chip (NoC) router macrocells, and a methodology to streamline their design and configuration. The methodology addresses the typical problems experienced by design and verification engineers when coding highly configurable intellectual property macrocells at Register Transfer Level (RTL) with hundreds of parameters and thousands of resulting configurations. A NoC infrastructure for a Multi Processor System-on-Chip (MPSoC) may require tens or hundreds of router macrocells. Therefore, managing the configuration design space is becoming a bottleneck for the design and verification of many-core processing systems. The proposed generation flow is illustrated on a real-world NoC router core. Its configurable architecture is compliant with several NoC topologies such as Ring, Octagon, Spidergon and 2D mesh typically used in many-core processing platforms. The generation flow allows for a reduction in the database code size, up to 70% in our experiments, and a contraction of three orders of magnitudes of the verification space vs. conventional design flows of RTL macrocells. The validity of the approach is also confirmed by synthesizing the generated router macrocells in nanoscale CMOS technology. The achieved performance compare well to the state-of-the-art in terms of low latency and low circuit complexity.

1. Introduction

A key element in the design of MPSoC is the global on-chip communication infrastructure, because its throughput, latency and power consumption set the limit to the overall performance of the computing platform. The traditional shared bus approach exhibits its limits as the number of integrated Intellectual Property (IP) cores increases. While gate delay scales with each new technology node, global wire delay increases and can be kept constant only by inserting repeaters [1,2]. For this reason shared bus communications standards are being substituted by multi-layer interconnects, now commonly referred as NoC, when designing many core systems. The NoC paradigm leverages the networking and parallel computing domain experience into the SoC world. It implements packet-switched micro-networks with a TCP/IP-like protocol stack. Examples of NoC, proposed by industry or academia, include Spidergon STNoC [2–5], Mango [6], Aethereal [7], Arteris [8], Sonics [9], SoCbus [10] and xPipes [11–13]. Fig. 1 illustrates the building blocks of a NoC and the corresponding layers in the TCP/IP protocol stack. The Network Interface (NI) connects the IP cores (e.g. processors, memories, DSP engines, ...) to the NoC domain. The NI, whose design has been detailed by the authors in [2], is made up of two separate components: shell and kernel. The shell encapsulates the transport layer and transforms local core transactions into NoC packets. The kernel implements the network layer and provides features such as data bus size and frequency conversion between the core and the NoC domain. Splitting transport and network layers into separate sub-components simplifies plug & play design style. The network is composed of a number of routers that pass packets between nodes. The router implements network and data-link layers. The physical link is responsible for actual signal propagation among routers and/or network interfaces. One of the major challenges when designing a communication platform is to minimize the design effort while attempting to cover the widest application space in terms of traffic requirements (high and/or guaranteed bandwidth, low latency, etc.) and implementation requirements (area, clocking scheme, power consumption) [14]. A number of algorithms [15–18] support high-level decisions like network topology, routing schemes, partitioning of clock domains. However, the actual implementation would not be feasible without NoC building blocks (router, NI, link) that provide the configurability necessary to match these high-level requirements. Particularly, the router building block is the core of the NoC communication infrastructure.

The goal of this paper is to introduce a novel configurable router architecture, particularly suited for low latency and low circuit complexity, and a methodology, named *metacoding*, which supports the design of the configurable components by providing a proper abstraction of the coding process.

* Corresponding author: Fax +39 050 2217 522.
E-mail address: sergio.saponara@iet.unipi.it (S. Saponara).

http://dx.doi.org/10.1016/j.micpro.2016.04.008
0141-9331/

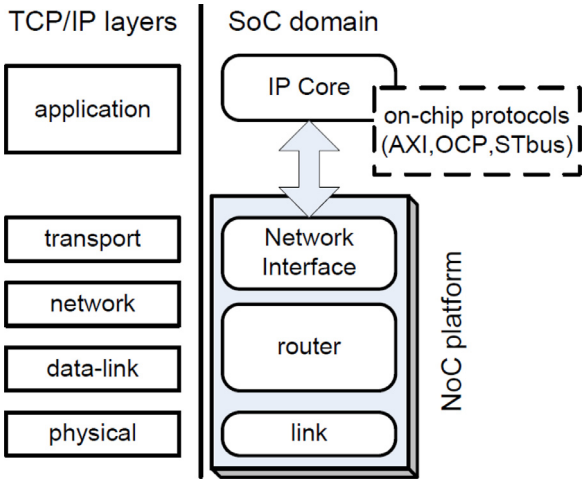

Fig. 1. Typical TCP-IP layers for Internet applications and their mapping onto NoC components.

Metacoding overcomes the limit of current Hardware Description Languages (HDLs) in capturing configuration intents and reflecting them into an optimized and easy to use RTL code database, which satisfies the following requisites:

- **Coded consistently**: unnecessary code is never generated, the internal components are the smallest required to provide a given functionality, unconnected signals and ports are removed, unused control signals are driven with proper values.
- **Neutral to tools:** the code database is read as-is by *any* standard front-end toolchain (LINT checking, functional simulation, RTL synthesis).
- **Verification friendly**: high code-/functional-coverage scores are achieved with a reasonable number of configurations.

The generated code through *metacoding* is a macrocell, i.e. a RTL HDL description that can be simulated and synthetized in any standard-cells library, or FPGA technology, with conventional Electronic Design Automation (EDA) tools used in digital IC design from vendors such as Synopsys, Cadence, Mentor (or custom FPGA-vendor tools for the FPGA flow). Once generated and verified the gate-level netlist, a conventional flow up to GDS-II data base finalization can be done. In this work, the generated code is in VHDL, but other languages for hardware description and synthesis, such as Verilog, can be used too.

Differently from other works in literature, that present top-to-bottom system level design flows [19–22], the approach proposed in this paper raises the level of abstraction of the design description, not of the design itself: the input of the flow is not a high-level specification, but RTL code templates and a set of properly defined rules to assemble them. The rationale of this choice is to streamline the generation of those design aspects that are repeatable, structured and prone to errors, while retaining the full advantages of manually coding RTL blocks, like the ability of achieving timing closure with extremely tight constraints. Using the terminology of the new IP-XACT standard [23,24], i.e. the XML format used to package reusable IP cores, the subject of this work should be referred to as a 'code generator', i.e. a software plug-in that customizes the core during the configuration activity.

Hereafter, Section 2 presents a novel architecture of a router, patented in [25,26]. The router architecture template can be configured to connect a sender or receiver IP core (e.g. processor, on-chip memory or controller for off-chip DRAM), through an NI, and up to other 4 neighbouring routers. This way it supports several network topologies typically used in MPSoC such as Ring, Octagon, Spidergon and 2D mesh. Section 3 analyzes the router configuration space and points out which classical RTL coding technique would be used to implement each feature. Section 4 presents *metarouter*, the object-oriented model that applies the *metacoding* principle to the new router architecture. Section 5 provides synthesis results of the generated macrocells in nanoscale CMOS technology. Section 6 compares the achieved performances vs. the state-of-the-art. Section 7 draws some conclusions.

2. Router features and architecture

All router features are configurable at synthesis time, from supported topologies to routing strategies, arbitration policies and clocking schemes.

2.1. Topology and routing Strategy

The router can support several topologies [2–4,24,27–30] such as Ring, Octagon, Spidergon, 2D mesh plus a family of customized topologies that can be derived from them. Each topology has some advantages and disadvantages, briefly reviewed hereafter. In a ring architecture, all nodes are connected in a ring fashion and each node has two neighbours independently from the size of the MP-SoC. All routers in a ring topology have the same number of links, just 3: one link to the NI (connecting the sender/receiver memory or processor to the NoC) and two links, typically named Left (L) and Right (R) or West (W) and East (E). The strengths of the Ring topology are: its small degree, the low-complexity of the router, faults can be easily detected and located. Its simplicity is paid in terms of a large ring diameter for MPSoC with a large number of cores. Moreover, a single fault in a link can disrupt the entire network and a high latency value is payed for the communication between two nodes at the opposite side of the NoC. Hence, Ring is suited for simple MPSoC platforms with few cores.

A basic Octagon NoC consists of 8 nodes and 12 bidirectional links. Each router has 4 links: one associated with the sender/receiver IP and 3 with neighbouring switches in Left (L), Right (R) and Across direction, which is new vs. the basic Ring topology. Thanks to the Across link an Octagon NoC features a low latency for the communication between two nodes at the opposite side of the NoC. The Octagon topology is characterized by a simple routing strategy: the idea is to move clockwise or counter clockwise along the ring to reach destination nodes which are near the source node, and to use the Across link as first or last hop to jump to a part of the network that is far away from the source node. Spidergon topology extends the capability of Octagon beyond the limit of 8 nodes, and presents the possibility of a router with a 5th link called Hierarchy (H). The Hierarchy link, see Fig. 2, can be used to connect a Spidergon NoC to another NoC domain thus creating a hierarchy in the network where the router acts as a gateway between the two NoC sub-networks. One limit of Spidergon in complex MPSoC is that the links in Across direction can be much longer than the other links. The insertion of repeaters may be needed, but this leads to an asymmetrical behaviour between the Across link and the other links.

In a 2D mesh the nodes are connected as a grid. Architecture expansion is easy for meshes and low effort is needed when adding more IP cores to the existing architecture. The presence of multiple paths between a pair of nodes makes a mesh NoC tolerant to link failure. However, in a 2D mesh NoC the nodes have different degrees according to their locations within the mesh. Corner nodes have degree of 2. Edge nodes have degree of 3. Inner nodes have degree of 4. Different routers are required in a 2D mesh NoC, since the number of links can be from 3 to 5: one to connect the sender/receiver IP core through the NI, and the others called North (N), South (S), East (E) and West (W) to the neighbouring routers.