Configurable network-on-chip router macrocells

SergioSaponara, LucaFanucci

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.

Keywords

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

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], [3], [4], [5], Mango [6], Aethereal [7], Arteris [8], Sonics [9], SoCbus [10] and xPipes [11], [12], [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], [16], [17], [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.

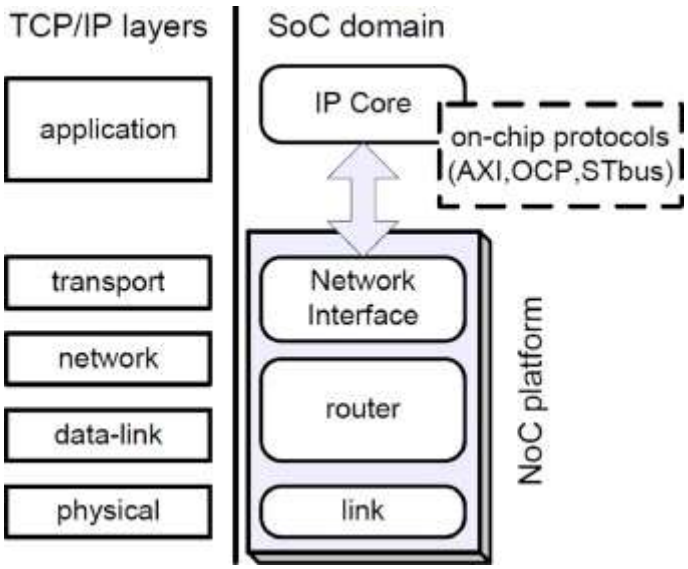

Fig. 1. Typical TCP-IP layers for Internet applications and their mapping onto NoC components.

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.

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], [20], [21], [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], [3], [4], [24], [27], [28], [29], [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 MPSoC. 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.

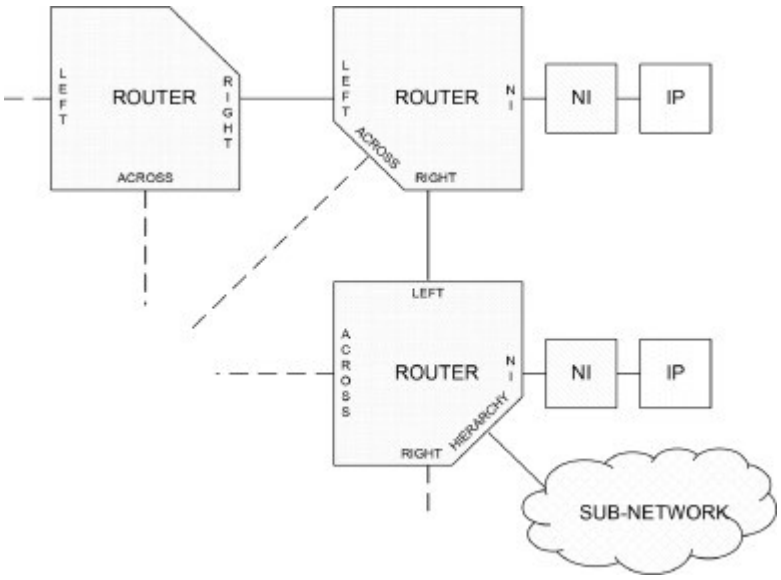

Fig. 2. Router connections in a NoC.

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. The limits of the 2D mesh topology are that the network diameter can be very large, and the topology is not regular (less bandwidth is available for the nodes placed at the corners and edges of the MPSoC). The router architecture is connected through two unidirectional links with up to four routers in the network. There is a fifth connection to the local NI. To avoid doubling the names of the router ports, independently of the topology (Ring, Octagon, Spidergon, 2D mesh), the router ports will be named as follows: NI, L for Left or West, R for Right or East, A for Across or South, H for Hierarchy or North.

The router adopts wormhole packet-switching, where a packet is subdivided into flits having their unique flow control. Once the first flit of a packet is routed, the remaining flits follow the same path reserved to the header. This way the amount of buffering is reduced, since queues are sized with the granularity of a flit instead of a packet. This approach allows deeply pipelined data paths when compared to virtual

cut-through and store & forward [29]. The routing algorithm is deterministic, so that always the same path is chosen between a source and a destination node, even if multiple paths exist. This choice avoids costly flit reordering at packet reception. The router uses a source-based routing: the entire path is encoded in the packet header, which has a fixed size due to the symmetry of the topology. This enables fast routing decision at each router, because the information is simply extracted from the header; no computation or routing lookup table is required. Adaptive routing algorithms have been proposed for NoC in literature. They make use of information about the state of the network to make routing decisions. By exploiting path diversity they allow better performance in network load balancing, but for an increased implementation complexity.

By deploying Virtual Channels (VCs), the router avoids end-to-end deadlocks. VCs provide logical links over the same shared physical channels, by establishing a number of independently allocated flit buffers in the transmitter/receiver nodes. Request and response Virtual Networks (VNs) are implemented on top of two disjoint VCs for sharing the physical link bandwidth. The parametric number of VCs supported by the router can lead to advanced routing schemes or independent Quality of Service (QoS) traffic classes for real time and low latency flows.

2.2. Router architecture

The architecture of the new proposed router consists of the following main blocks, instantiated as many times as the number of links (see Figs. 3 and 4):

- Downstream Interface (DS ITF), connected to the input link and buffering incoming flits.
- Routing Computation Unit (RCU), which extracts routing information from packets' headers.
- Switch, which routes inputs to outputs.
- Optional Output Queue (OQ) with bypass capability, which stores outgoing flits.

-

OQ Arbiter, which arbitrates among different inputs requesting the same Output Queue. It is in charge of the per-packet allocation of the OQ, and so of the Virtual Channel.

-

VN Arbiter, which arbitrates between the two Virtual Networks requesting link access. This is a per-flit arbitration, because the two packets of the two VNs can be interleaved on the link, depending on credits availability.

-

Upstream Interface (US ITF), which is in charge of managing the credit-based output data flow.

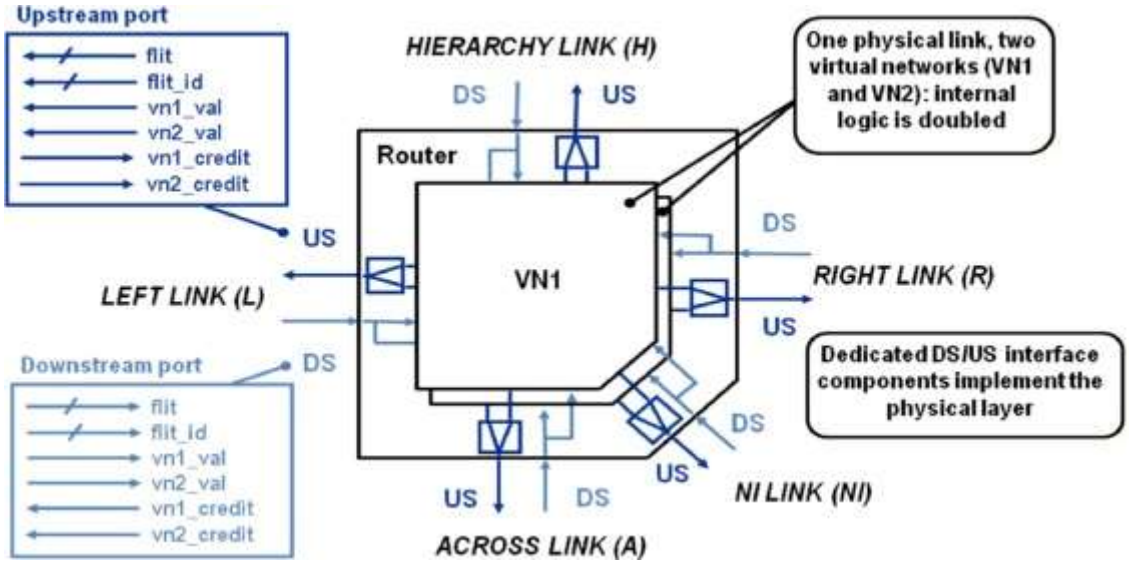

Fig. 3. Router architecture with two virtual networks.

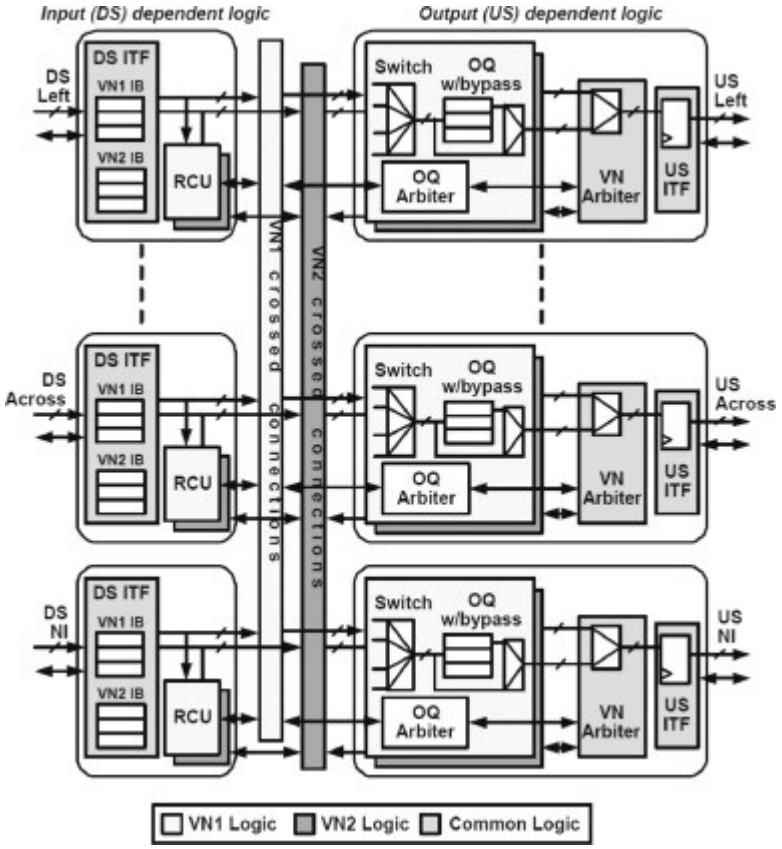

Fig. 4. Architecture of the router with two virtual Networks.

OQ and VN arbiters in Fig. 4 support different techniques such as Round Robin (RR), Least Recently Used (LRU), fixed priority. Downstream and Upstream Interfaces are dedicated components that implement the physical layer. The different links – synchronous or mesochronous [2], [31] – correspond to different DS/US Interfaces instantiated in the router and in other NoC modules. The above mentioned blocks can be grouped into DS-dependent blocks (DS Interface, RCU) and US-dependent blocks (Switch, Output Queue, OQ Arbiter, VN Arbiter and US Interface), as shown in Fig. 4. DS-dependent blocks are instantiated for a link only if the corresponding DS interface exists. US-dependent blocks are instantiated for a link only if the corresponding US interface exists. The DS-dependent block of each link is connected to the US-dependent blocks of all the other links.

At the NoC physical interface side, there are the following configurable hardwired lines for each response or request path:

-

N-bit *flits* to transfer NoC packets (headers and payloads).

-

4-bit *flit_id*, whose 2 LSBs identify first, intermediate and last flits of a packet; the flit can also be a single one. The other 2 optional bits are used to mark the end of bus packets within compound transactions (i.e. composed of a number of packets) that are translated into a single NoC packet, and to identify payload flits that cannot be aggregated in case of upsize conversion (necessary in some cases of interoperability).

-Optional K-bit *four_be* (K = N/32) to mark meaningful 32-bit pieces of data within a flit (used in end-to-end size conversion).

- Optional 2-bit *flit_id_error* for signalling a slave side error or an interconnect error.

- Optional *flit_id_atomic* that enables support of atomic operations: an NI can lock paths towards a Slave IP so that a Master IP can perform a generic number of consecutive operations without any interference from other masters.

- C*redit* and *valid (val)* signals for credit-based flow control.

2.3. Buffering and latency

The router adopts a credit-based flow control (*credit* and *val* signals in Fig. 3), which works on a flit per flit basis. A flit is sent only when there is room enough to receive it: neither retransmission nor flit dropping is allowed. This is done automatically by setting an initial number of credits in the US interface (in its Credit Manager), equal to the size of the Input Buffer in the DS interface it communicates with. Since the US interface sends flits only if the connected DS interface can accept them, there are no pending flits on the link wires. This approach allows virtual channel flit-level interleaving, so that separate virtual networks can share the same physical link. To guarantee the maximum link throughput, the Input Buffer is sized according to the credit round-trip delay. This delay can be defined as the minimum time between two consecutive credits for the same buffer location and it depends on the credit pipeline between two consecutive routers. The value of this delay is configurable, thus different design trade-offs in terms of working frequency and buffer resources are possible. The router uses Output Queues for enhanced performance, avoiding head-of-line blocking. Queues are shared among input flows. They have a bypass feature to reduce the router crossing latency in case of low traffic conditions. Output Queues are optional and are usually not instantiated in low cost implementations. It is optionally possible to implement a separate Output Queue for each input flow

targeting a given output. This configuration improves the offloading capability of a router and it is useful when multiple incoming heavy traffic streams target the same output port, e.g. the NI port.
The router has a configurable crossing latency, from one up to two clock cycles. This is obtained by means of a flexible pipeline in the data path: one stage is represented by the Input Buffer and the other stage is the optional output retiming register. The Output Queue, when instantiated, can be bypassed in case of low traffic conditions, thus not affecting the overall router crossing latency. Registering input and output port signals allows for orthogonality between the link and the router delay. For very low-complex NoC applications, e.g. connecting few IPs in a SoC with a small area, where buffering flits in the NoC is not required, the latency and area overhead of the router can be further reduced bypassing also the Input Buffer. In such case a zero-cycle latency router is obtained and for correct data sampling the NoC relies on the output and input registers of the NIs where are connected the sender and the receiver IPs, respectively.

2.4. Quality of service

QoS is based on a Fair Bandwidth Allocation (FBA) mechanism. It allows for a flexible, scalable and low cost management of the allocation of the available bandwidth. The principle of FBA arbitration is to share the Slave available bandwidth among the Masters during peak request period. Since the arbitration is distributed among the different router crossed by the packets, traditional weighted Round Robin arbitration algorithms cannot be used. As detailed in [2], where the architecture of the NI is discussed by the authors, the solution is to apply a faction tag at packet injection, i.e. in the NI. Packets with the same tag are kept together, within the same faction round, in the interconnect, i.e., in the routers, where the distributed arbitration is performed. This scheme should not be confused with the TDMA approach: the faction round duration is variable; if only packets belonging to a new tag are received, they win the arbitration, so that there is not wasted bandwidth.
In the FBA QoS scheme, the NI tags the packets with a faction identifier and, if needed, with their priority. Each injected flow specifies the requested bandwidth, which is the global amount of data (computed in bytes from the opcode size) transferred by the considered NI flow in a given faction round at the specific target. The round at the specific target is a given number of available accesses; the number

of bytes read or written in that round represents the percentage of available bandwidth (bandwidth in the round) demanded by this initiator flow. The requested bandwidth corresponds to a threshold that must be reached by a counter to switch the faction identifier bit. The counter (inside the shell of an Initiator NI) computes from the opcode the number of bytes that flow to each target and enables the faction bit switching when the threshold is reached. Two different schemes can be enabled: one offering to configure a separate threshold (or bandwidth) for each target, and a simplified scheme with a unique threshold for all targets. To be noted that the requested bandwidth value is programmed at the injection point (NI) and is not explicitly linked to the path of a data flow through the router as it is done in other NoC architectures [6], [7]. The arbitration logic in the router is thus simpler. Proposed routers do not need to be programmed. They implement simple arbiters, since only a two-steps arbitration process is required based on the information available in the network, without any need of slow complex logic. Speculation-like techniques allow the two arbitrations to be performed in parallel, thus reducing the critical path delay. The same scheme is used for any VNs. If all data flows have the same bandwidth reservation, the arbitration degenerates into a Round Robin, Least Recently Used or fixed priority scheme. A QoS bandwidth control mechanism for NoC, which is not explicitly linked to the path of a data flow through the router, but is programmed at the injection point (NI) has been proposed in [32]. The proposed FBA QOS technique, implemented at NI side by means of a tagging mechanism based on a simple counter (with a negligible hardware overhead, as we demonstrated in [2]), stands for its simplicity vs. the solution in [32]. The latter is based on a more sophisticated (σ, ρ) network calculus, inherited from [33], where the parameters should be proper determined to guarantee the minimal bandwidth requirements, the absence of FIFO queue overflow in the routers, and the fairness of bandwidth allocation between the different flows.

3. Router design configuration

The switching matrix of the proposed router is fully configurable: the existence of *each* stream direction (up and down) in *any* virtual network can be independently specified for *each* link (L, R, A, NI, H). This subset of the configuration space is referred to as *router topology* or *backbone*. The ability to fully control the router topology enables to strip away unnecessary paths in the network, thus saving hardware resources (e.g. FIFO buffers, arbiters, muxes, etc.). A convenient

representation for the router topology is shown in Fig. 5(a), where a switching matrix is defined for each virtual channel. Rows represent DS ports, while columns are associated to US ports. A non-null element at position (*i, j*) indicates that traffic entering port *i* can be routed toward port *j*. A null row/column represents a missing DS/US port, as illustrated in the equivalent graph representation of Fig. 5(b). Note that DS and US port sets overlap if and only if the interconnection matrix is symmetric. The router configuration space is summarized in Table 1. Features are classified according to their functional meaning. The *configuration layers* identify the coding technique required to implement that specific feature in a configurable fashion.

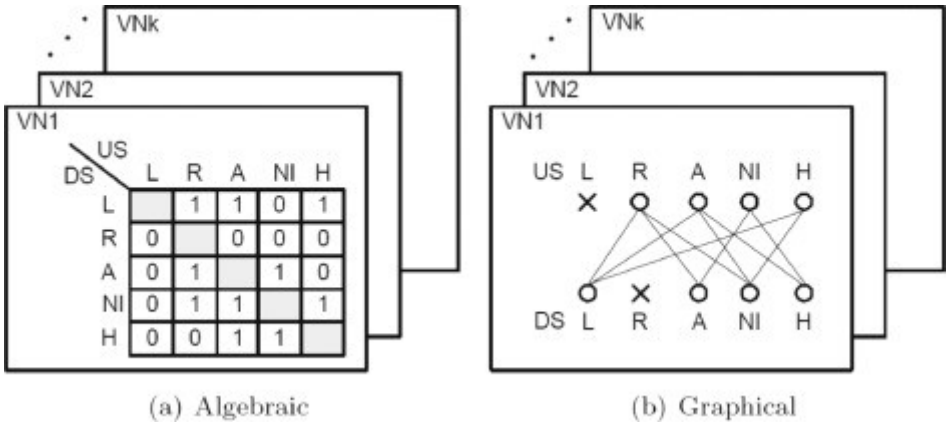

Fig. 5. Representation of router configurable topology.

Table 1. Router configuration space.

Feature	Family	Configuration layer
Link	Topology	Layer1
Stream		
VN		
Reserved queues	Buffering & latency	
Queue size		Layer3
Crossing latency		Layer2
Credit round-trip delay		
Arbitration schemes	QoS	
Clocking scheme	link	
Flit size		Layer3

Features in Layer1 represent design parts that can either exist or not, thus reflecting the router topology.

Features in Layer2 need different code layouts for each legal configuration of Layer1. Indeed, the type of sub-components to instantiate and the signals connecting them to the surrounding logic depends on the router topology. These features are described with conditional instance statements (e.g. IF..GENERATE in VHDL/Verilog2001) and a consistent scenario for any possible configurations has to be considered during the coding phase. Moreover, conditional statements do not allow modification of port maps, thus additional pre-processor directives are required. For example, when configuring the crossing latency, registers are either inserted or removed across the data-path and mutual connections need to be changed accordingly. If different arbitration schemes are selected, then different arbiter components should be instantiated and properly connected to the surrounding logic.

Layer3 includes parametric features such as size of buses and memories. Parametric features are easily implemented by means of array-like data types supported in any HDL.

4. The metacoding approach

As explained in previous section, the RTL coding of the router requires an effort that grows exponentially with the dimension of Layer1 space. Considering two virtual channels, five links and independent US and DS paths for each link, the number of possible configurations is greater than 2^{16}. Hand-writing the whole code database is not only time-consuming, but also prone to errors, because of the high similarity among code portions related to the same sub-component. Lastly, since each configuration comes from the combination of different portions of code, full code coverage can be achieved only by verifying any legal router topology. The greatest limitation coming from a traditional coding style based on conditional instance statements and pre-processor directives is the *locality* of such statements. Portions of code activated by the same condition, but located in different sub-components have to be replicated, because each statement queries the configuration space independently from each other. This work refers to the abstraction process as *metacoding*, while *metadesign* is used to denote the code generator. Consistently with the terminology adopted in [34], the term *codelet* designates a group of HDL statements describing some part of the design. The metadesign is an equivalent representation of the design composed of three modules that interact as illustrated in Fig. 6:

- **Checker**: constraints the configuration space by filtering any illegal set of parameters.
- **Backbone**: stores the configuration and provides the codelets with methods for querying parameters' value.
- **Codelet**: a basic class equipped with methods for HDL code customization. Since different codelets may require different customization procedures, the codelet class can be specialized for each of them.

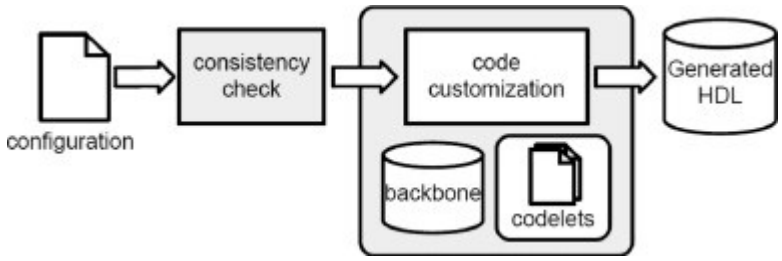

Fig. 6. Main steps in the metacoding flow.

A commercial tool for IP packaging (Synopsys coreTools) embodies the checker. The configuration parameters are annotated with proper expressions that the tool uses to determine whether a configuration is legal or not. The backbone and the codelets are custom software components, which are aware of the design structure and are hooked to the above mentioned tool. The backbone captures the configuration intents in such a way that the codelets can query to customize themselves. Codelet objects are in turn composed of:

An HDL template describing some parts of the design without explicit references to design items (components, signals, ports).

A set of rules for substituting instance specific items within the template.

A signature, used as a key for querying the backbone. The signature is built concatenating the coordinates of the codelet in the Layer1 space (i.e. Layer1 parameters determining the existence of that codelet in the HDL).

As already highlighted in Section 1, differently from [19], [20], [21], the metacoding approach raises the level of abstraction of the design description, not of

the design itself. In other words, the codelets use the same hardware abstraction layer of the design (RTL in this case). Fig. 7 illustrates the internals of the router metadesign (*metarouter*) in the form of a UML class diagram. The LinkStreamVN class impersonates the backbone. The codelet class is further specialized according to the components highlighted in Figs. 3 and 4. An intermediate level of specialization differentiates between single and multiple VN components. All classes are implemented using XOTcl, an aspect oriented scripting language based on Tcl [35].

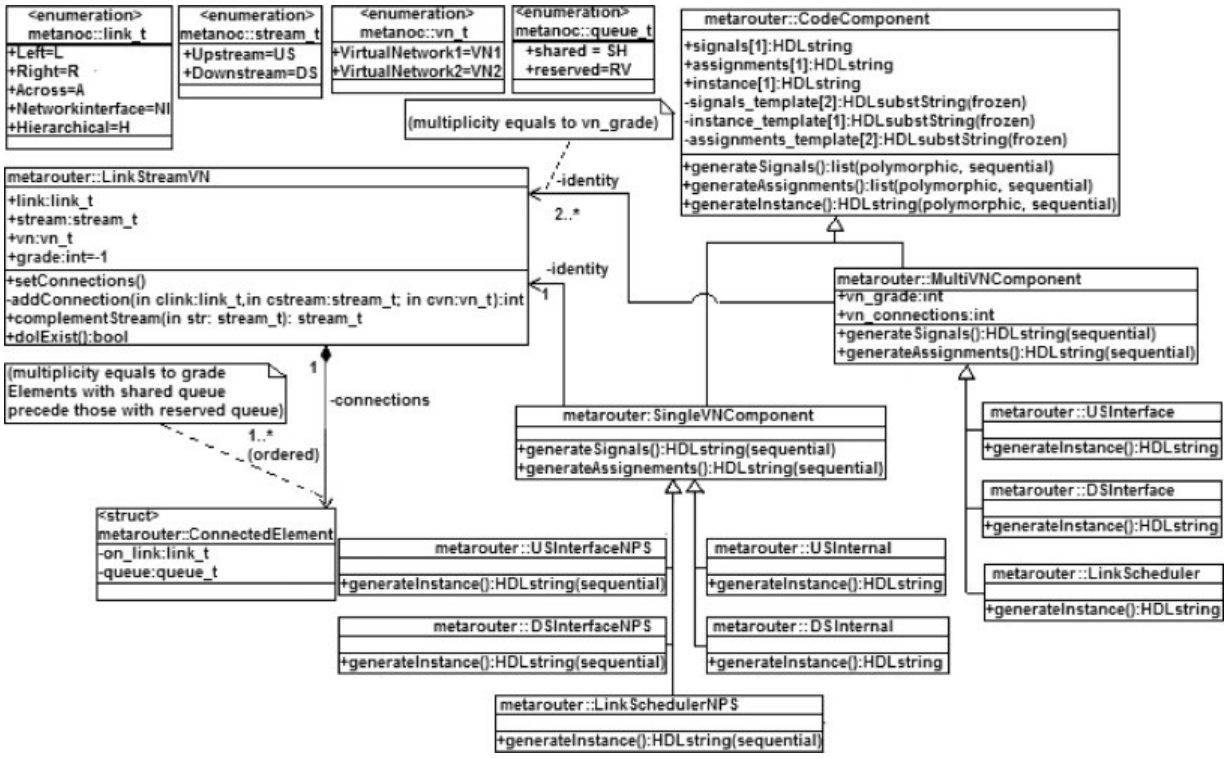

Fig. 7. UML class diagram of the metarouter plug-in.

5. CMOS implementation results

Table 2 compares the metacoding approach with a standard coding approach using the router code database as a benchmark. The router was originally implemented using plain VHDL and pre-processor directives. The migration to the metacoding approach occurred before a number of features (like additional ports) were added. The statistics reported in the '*hand-coded*' column are a projection of the original code database considering the configurability implemented in the metarouter. The metadesign achieves a reduction in the size of the code database greater than 60%. The number of distinct topologies to verify is decreased of three orders of magnitudes. This reduction is possible since router topologies can now be grouped

into equivalence classes and only one representative for each class needs to be verified. The reduction of the data base code size and of the configurations to be verified directly leads to a reduction of time and costs for design and verification [36].

Table 2. Performance of metacoding on router design.

	Hand coded	**Metadesign**	**Saving**
HDL (lines)	11,158	4093	63%
Configuration intents (lines)	3211 (preprocessor macro)	994 (Tcl/XOTcl)	69%
Topologies to verify	$> 2^{16}$	56	10^{-3}

The generated VHDL code through *metacoding* can be simulated and synthetized in any standard-cells CMOS library, or in FPGA technology, with conventional EDA tools used in digital IC design. The time needed for simulations, verification and synthesis of the code generated through *metacoding* is in line with the time needed for a handcrafted VHDL version using "if…..generate" constructs to achieve the IP configurability. The time needed to generate a configuration is not an issue, when compared to commercial tools for IP packaging and configurations, such as Core Tools from Synopsys, and to the entire HDL-based design flow, which includes much higher time-consuming tasks such as synthesis and simulations (particularly with time back-annotated netlist).

Once generated and verified the gate-level netlist, a conventional flow up to GDS-II data base finalization can be done (based on encounter RTL-to-GDSII EDA tool). In this work, different configurations of RTL router macrocells have been synthetized in STM 65 nm and 45 nm standard-cells CMOS libraries. The standard-cells are available in a high-speed (low V_{TH}) version and in a low-leakage (high V_{TH}) one. For a sake of space the results for 5 configurations, all with only 1 clock cycle of crossing latency, are reported in Table 3. As example the router configured with 5 ports (A, R, L, NI and H), 1 VN, 128 bits of bus width, no retiming stage, with input buffer but without output que, and targeting a 400 MHz clock frequency as in [37], has a logic complexity of 32.8 kgates. The power consumption at full speed in 65 nm CMOS technology with 1.1 V supply voltage, low-leakage library version, is 10 mW. This value is dominated by dynamic power. The static power due to leakage is only 40 µW. The same router configuration, using the same technology, but with 21 VNs has a logic complexity of 60.6 kgates. Its dynamic power consumption is 18 mW and its static power is 72 µW. By using a standard-cells

library version optimized for high-speed, with the same router configuration, clock frequencies up to 1 GHz are met, as in [38]. Hence, the proposed router can sustain up to 128 Gbps data transfer per link. In this case the static power consumption increases to roughly 1 mW. As example of a very low complexity implementation, a 3-port router with 32-bit data size and a single VN, with no OQs instantiated, has a circuit complexity lower than 9 kgates. In low-leakage 65 nm CMOS technology it can run at 500 MHz clock frequency with a static power of 10 µW and a dynamic power of few mW. The same router IP core in high-speed 65 nm technology library achieves a clock frequency up to 1 GHz (i.e. a 32 Gbps data rate, enough to sustain also the traffic of multimedia processing platforms [39]) with a static and dynamic power consumption of 200 µW and roughly 7 mW, respectively.

Table 3. Performance in 65 nm CMOS technology of some router configurations.

	Clock frequency	**Complexity kgates**	**Dynamic power**	**Static power**
3-port router, 11 VN, 32 bus-width, low-leak tech	500 MHz	9	3.4 mW	10 µW
4-port router, 11 VN, 128 bus-width, low-leak tech	400 MHz	25.1	7.7 mW	31 µW
5-port router, 11 VN, 128 bus-width, low-leak tech	400 MHz	32.8	10 mW	40 µW
5-port router, 21 VN, 128 bus-width, low-leak tech	400 MHz	60.6	18.1 mW	73 µW
5-port router, 11 VN, 128 bus-width, high-speed tech	1 GHz	61.1	45.2 mW	1 mW

The proposed methodology is applied in this work to a router macrocell generation in NoC design applications, but it can be generalized also to other configurable IP cores with lots of parameters, such as configurable microprocessors. The proposed *metarouter* methodology is currently embedded in an industrial flow by STMicroelectronics. Its target is the generation and implementation of the whole interconnection NoC platform of real-world SoC for multimedia and telecom applications, fabricated in nanoscale CMOS technologies, from 65 nm down to 28 nm. References to this environment, which includes a proper user interface, STNoC GUI, and a NoC compiler, I-NoC, can be found in [40], [41], [42]. Beside the router, discussed in this work and patented in [25], [26], other key blocks of the STNoC platform are the NI, and different types of links (mesochronous or synchronous or adaptive link), that the authors have presented in [2], [31].

6. Comparison to the state-of-the-art

Comparing the results in Section 5 to other NoC router architectures in literature [37], [38], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], synthesized in CMOS technology, our design stands for these characteristics:

(i) low crossing latency, limited to 1 clock cycle, and low circuit complexity, limited to tens of kgates, as the best in class router architectures proposed for low-latency [43], [44] or low complexity [38], [48];

(ii) high configuration space leading to a high flexibility when assembling the entire NoC interconnect platform; our design supports multiple topologies (2D mesh, Spidergon, Ring, Octagon) while in state-of-the-art most routers are customized for specific topologies as in [37].

With respect to [43], [44] the same latency is achieved by our design. The works in [43], [44] can support 2D mesh topology and adopts VNs. Our work allows for a much higher throughput, up to 128 Gbps per link instead of 16 Gbps per link in [43], [44]. The absolute value of our router crossing latency, 1 ns with the high-speed CMOS library version is even better than the latency performance of asynchronous routers in [45], [46]. Circuit complexity results, in the order of tens of kgates for the generated router cells, are aligned with the best designs in the state-of-the-art [38], [48]. Particularly, Table 4 compares our router to recent works supporting VNs, and both 2D mesh and Spidergon topologies. Results from [38] refer to 65 nm CMOS technology, with 21 VNs per port and 128-bit flit size, as the configuration of our router in the last row of Table 3. Results from [48] refer to 45 nm CMOS technology, with 21 VNs per port and 64-bit flit size. With respect to [48], [49] our design exhibits a similar complexity, but allows for a lower latency and for an increased speed. With respect to [38] our design has the same working frequency, but much lower power dissipation and circuit complexity and better latency performance. Reference [49] has been realized in 90 nm CMOS. The authors of [49] have also proposed a fault-tolerant router architecture in [50] where the increased capability of fault tolerance is paid in terms of decrease of the maximum clock frequency, limited at 200 MHz, and increase of the circuit complexity, which is 48.4 kgates. In [51] an heterogeneous architecture is proposed, sized in 45 nm CMOS technology for different VN values per port (from 1 to 41 VNs), with 32-bit flit size and targeting a 500 MHz clock frequency. The trade-off between circuit complexity and working frequency of our solution

outperforms also other routers proposed in literature such as [52], [53] having a complexity above 100 kgates and a frequency below 1 GHz.

Table 4. Comparison of state-of-the-art 5-port routers for 2D mesh/Spidergon with VNs.

Router/technology	VNs	Bits/flit	Clock frequency	Complexity kgates	Power, mW	Latency, clock cycles
Our 65 nm	2	128	1 GHz	61.1	46.2	1
	1		400 MHz	32.8	10	1
[48] **45 nm**	2	64	500 MHz	50	12.8	2 to 4 with/ without prediction
[38] **65 nm**	2	128	1 GHz	163.5	108.8	15 to 25 for an 8 flit burst
[49] **90 nm**	1	36	245 MHz	33	N/A	N/A
[51] **45 nm**	1 to 4	32	500 MHz	78	N/A	> 2

7. Conclusions

This paper presented a novel configurable architecture of a NoC router and its implementation using a methodology that generates correct-by-construction technology independent RTL code databases. The router architecture can support several NoC topologies, typically used in MPSoC design, such as Ring, Octagon, Spidergon, 2D mesh plus a family of customized topologies that can be derived from them. The RTL coding itself is abstracted and modelled with an Object Oriented framework, integrated within a tool for IP packaging. Compared with traditional coding styles based on pre-processor directives, the proposed methodology produces smaller code database (more than 60% reduction) and a considerable reduction of the verification space by decoupling the verification of the codelets and of the backbone. The generated router macrocells have been synthetized in CMOS technology demonstrating good performance vs. state-of-the-art router designs. Similar techniques, not discussed in this paper, have been applied to the design of the network interface and to abstract NoC features such as network and transport packet format, routing tables and network access policy. By abstracting these aspects, it was possible to write the HDL code of the NoC platform components focusing on performance optimization without sacrificing code readability, hence maintenance. This approach also ensured consistency among all the instances of the network components.

Acknowledgements

Discusions with T. Bacchillone and N. L'Insalata now with Synopsys ltd Ireland, R. Locatelli and M. Coppola from STMicroelectronics, France, and E. Petri are acknowledged.

References

[1] R. Ho, K. Mai, M. Horowitz
The future of wire
Proc. IEEE, 89 (4) (2001), pp. 490-504

[2] S. Saponara, *et al.*
Design of an NoC interface macrocell with hardware support of advanced networking functionalities
IEEE Trans. Comput., 63 (3) (2014), pp. 609-621

[3] M. Coppola, M.D. Grammatikakis, R. Locatelli, G. Maruccia, L. Pieralisi, **Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC,** (first ed.), CRC Press (2008)

[4] L. Bononi, *et al.,* **Simulation and analysis of network on chip archi tectures: ring, spidergon and 2D mesh,** DATE, IEEE (2006), pp. 154-159,

[5] S. Saponara, M. Martina, M. Casula, G. Masera, L. Fanucci, **Motion estimation and CABAC VLSI coprocessors for real-time high-quality H.264/AVC video coding,** Microprocess. Microsyst., 34 (2010), pp. 316-328

[6] T. Bjerregaard, J. Sparso, **A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip,** DATE, IEEE (2005), pp. 1226-1231

[7] K. Goossens, *et al.*. **The aethereal network on chip after ten years: goals, evolution, lessons, and future.** DAC, IEEE (2010)

[8] J.-J. Lecler, G. Baillieu., **Application driven network-on-chip architecture exploration & refinement for a complex SoC,** Des. Autom. Emb. Syst., 15 (2) (2011), pp. 133-158

[9] T. Pontius, **Solving network challenges in advanced multicore SoCs.,** Multicore Developers Conference (2014)

[10] D. Wiklund, D. Liu. **SoCBUS: switched network on chip for hard real time embedded systems,** Proceedings International Parallel and Distributed Processing Symposium (2003)

[11] A. Pullini, *et al.,* **NoC design and implementation in 65 nm technology,** NOCS, IEEE (2007), pp. 273-282

[12] S. Stergiou, *et al.,* **Xpipes lite: a synthesis oriented design library for networks on chips,** DATE, 2, IEEE (2005), pp. 1188-1193

[13] D. Bertozzi, *et al.* **Xpipes: a network-on-chip architecture for gigascale systems-on-chip,** IEEE Circ. Syst. Mag., 4 (2) (2004), pp. 18-31

[14] T.D. Hamalainen, *et al.* **Gamification of System-on-Chip design,** IEE Int. Symp. on SoC (2014), pp. 1-8

[15] S. Murali, *et al.,* **Mapping and configuration methods for multi-use-case networks on chips,** ASPDAC, IEEE (2006), pp. 146-151

[16] D. Bertozzi, *et al.,* **NoC synthesis flow for customized domain specific multiprocessor systems-on-chip,** IEEE Trans. Parallel Distrib. Syst., 16 (2) (2005), pp. 113-129

[17] Syed M.H.A. Jafri, *et al.,* **Energy-aware fault-tolerant network-on-chips for addressing multiple traffic classes,** Microprocess. Microsyst., 37 (8) (2013), pp. 811-822

[18] D. Rhamati, *et al.,* **Power-efficient deterministic and adaptive routing in torus networks-on-chip,** Microprocess. Microsyst., 36 (2012), pp. 571-585

[19] A. Sangiovanni-Vincentelli, *et al.,* **Metamodeling: an emerging representation paradigm for system-level design,** IEEE Des. Test Comput., 26 (2009), pp. 54-69

[20] R. Damasevicius, V. Stuikys, **Application of UML for hardware design based on design process model,** ASPDAC, IEEE (2004), pp. 244-249

[21] S. Shukla, **Metamodeling: what is it good for?,** IEEE Des. Test Comput., 26 (3) (2009)

[22] J. Cadavid, *et al.,* **An analysis of metamodeling practices for MOF and OCL,** Comput. Lang. Syst. Struct., 41 (2015), pp. 42-65

[23] IEC 62014-4, IEEE STD 1658 –2009. IEEE/IEC International Standard - IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows. 2015, pp. 1–373, ISBN 978-2-8322-2265-2.

[24] El Mrabiti, *et al.,* **Design environment for the support of configurable network interfaces in NoC-based platforms,** SAMOS, IEEE (2010), pp. 63-70

[25] M. Coppola, S. Saponara, L. Fanucci et al., "Networks on-chip router", EP 2592800 A1

[26] M. Coppola, S. Saponara, L. Fanucci et al., "Zero-cycle router for networks on-chip", US 20130136129 A1

[27] M.A.AbdEl Ghany, M.A. El-Moursy, M. Ismail, **High throughput architecture for high performance NoC,** Florin Balasa (Ed.), Data Storage (2010), ISBN: 978-953-307-063-6

[28] Jie Chen, *et al.,* **Network-on-chip (NoC) topologies and performance: a review,** IEEE NECEC (Newfoudland Electrical and Computer Engineering Conference) (2011), pp. 1-6

[29] W. Dally, B. Towles, **Principles and Practices of Interconnection Networks,** Morgan Kaufmann (2003)

[30] G. Reehal, *et al.,* **Octagon architecture for low power and high performance NoC design,** NAECON, IEEE (2012), pp. 63-67

[31] F. Vitullo, *et al.,* **"Low-complexity link microarchitecture for mesochronous communication in networks-on-chip,** IEEE Trans. Comput., 57 (2008), pp. 1196-1201

[32] B. Dupont de Dinechin, *et al.,* **Guaranteed services of the NoC of a manycore processor** NoCArc, IEEE (2014), pp. 11-16,

[33] Z. Lu, *et al.,* **Flow regulation for on-chip communication,** (2009), pp. 578-581

[34] M. Frigo, **A fast fourier transform compiler,** Proceedings ACM SIGPLAN Conference, 34 (1999), pp. 169-180

[35] G. Neumann, U. Zdun, **XOTcl, an object-oriented scripting language,** 7th USENIX Tcl/Tk Conference (2000)

[36] S. Saponara, *et al.,* **Design and coverage-driven verification of a novel network-interface IP macrocell for network-on-chip interconnects,** Microprocess. Microsyst., 35 (6) (2011), pp. 579-592

[37] J.-Y. Kim, *et al.,* **A 118.4 GB/s multi-casting network-on-chip with hierarchical star-ring combined topology for real-time object recognition,** IEEE J. Solid State Circuits, 42 (7) (2010), pp. 1399-1409

[38] G. Jiang, *et al.,* **A low-latency and low-power hybrid scheme for on-chip networks** IEEE Trans. VLSI Syst., 23 (4) (2015), pp. 664-677

[39] L. Fanucci, *et al.,***A parametric VLSI architecture for video motion,** Integr. VLSI J., 31 (1) (2001), pp. 79-100

[40] F. Dubois, *et al.,* **Spidergon STNoC design flow,** IEEE/ACM International Symposium on Networks on Chip (NOCS) (2011), pp. 267-268

[41] M. Coppola, "Spidergon STNoC. The technology that add values to your system", Hot Chips 22, available from http://www.hotchips.org/wp-content/uploads/hc_archives/hc22/HC22.24.545-1-Coppola-Spidergon.pdf (accessed March 2015)

[42] M. Soulie, "Spidergon STNoC overview, and low-power specific features", available (access March 2015) from http://www.minatec-crossroads.com/sites/default/files/DTC-Minatec2010-M-Soulie_ST.pdf

[43] R. Mullins, *et al.,* **The design and implementation of a low latency on-chip network** ASPDAC, IEEE (2006), pp. 164-169

[44] R. Mullins, *et al.,* **Low-latency virtual-channel routers for on-chip networks,** ISCA, IEEE (2004), pp. 1-10

[45] N. Onizawa, *et al.,* **High-throughput compact delay-insensitive asynchronous NoC router,** IEEE Trans. Comput., 63 (3) (2014), pp. 637-649

[46] Y. Thonnart, P. Vivet, F. Clermidy, **A fully-asynchronous low-power framework for GALS NoC integration,** DATE, IEEE (2010), pp. 33-38

[47] A. Monemi, *et al.,* **Low latency network-on-chip router microarchitecture using request masking technique,** Int. J. Reconfigr. Comput., 2015 (2015), pp. 1-13

[48] H. Matsutani, *et al.,* **Prediction router: a low-latency on-chip router architecture with multiple predictors,** IEEE Trans. Comput., 60 (6) (2011), pp. 783-799

[49] J. Liu, *et al.,* **Online traffic-aware fault detection for networks-on-chip,** J. Parallel Distrib. Comput., 74 (1) (2014), pp. 1984-1993

[50] J. Liu, *et al.*. **Fault-tolerant networks-on-chip routing with coarse and fine-grained look-ahead,** IEEE Trans. on CAD, 35 (2) (2016)

[51] Y. Ben-Itzhak, *et al.,* **Heterogenous NoC routrer architecture,** IEEE Trans. Parallel Distrib. Syst., 26 (9) (2015), pp. 2479-2492

[52] C. Feng, *et al.,* **Addressing transient and permanent faults in NoC with efficient fault-tolerant detection router,** IEEE Trans. VLSI Syst., 21 (6) (2013), pp. 1053-1066

[53] Ling Xin, *et al.,* **A low-latency NoC router with lookahead bypass,** ISCAS, IEEE (2010), pp. 3981-3984

Sergio Saponara, IEEE Senior Member, got Laurea and Ph.D. degrees in Electronic Engineering from University of Pisa in 1999 and 2003, respectively. In 2002 he was Marie Curie Research Fellow at IMEC (Belgium). He is Professor of Electronics at Pisa University and at the Italian Naval Academy in Livorno. He is also CTO of Ingeniars srl. His research interests include vehicle electronics, mixed-signal integrated circuits, RF wireless and vision systems. He is co-author of more than 300 journal and conference papers and co-inventor of 17 patents. He is affiliate researcher of the national interuniversity consortia for telecommunications (CNIT) and for Informatics (CIN) and of the Italian National Institute for Nuclear Physics (INFN). He served in several SPIE and IEEE international conferences program committees. He is member of the editorial board of Springer Journal of Real-Time Image Processing, IET/IEEE Electronics Letters, MDPI Designs

Luca Fanucci got Laurea and Ph.D. degrees in Electronic Engineering from University of Pisa in 1992 and 1996, respectively. From 1992 to 1996, he was with the ESA-ESTEC, Noordwijk (NL), as research fellow. From 1996 to 2004 he was a senior researcher of the Italian National Research Council in Pisa. He is Professor of Microelectronics at Pisa University and CEO of Ingeniars srl. His research interests include several aspects of design technologies for integrated circuits and electronic systems. He is co-author of more than 400 journal and conference papers and co-inventor of more than 50 patents. He served in several technical programme committees of international conferences. He was Program Chair of DSD 2008 and DATE 2014. He is currently serving as Vice-General Chair at DATE 2015. He is a member of the editorial board of