Formal Specification and Verification of Communication in Network-On-Chip: An Overview

https://doi.org/10.3991/ijes.v6i4.9416

Fateh Boutekkouk
University of Oum El Bouaghi, Algeria
Fateh_boutekkouk@yahoo.fr

Abstract—Network-On-Chip (NOC) is an emerging paradigm to surmount traditional bus bused Systems-On-Chip (SOC) limits especially scalability and communication performances. A NOC includes many applications that can execute concurrently. This situation may show some undesirable behaviors such as deadlock, livelock, starvation, etc. On the other hand, the application of formal methods to on-chip communication infrastructures has received more attention. Formal analysis of NOC communication will be very advantageous since it allows proving some theorems or interesting qualitative/quantitative properties on the communication behavior where simulation/emulation techniques can fail easily. In this paper we try to give an overview of the most famous formal methods applied to the verification of communication inside NOCs.

Keywords—Network-On-Chip, On-communication, Routing, Formal Verification, Model checking, Theorem proving

1 Introduction

Networks-On-Chip or NOCs [1] emerge as a new communication centric paradigm that replaces the well known Systems-On-Chip (SOC) to reduce the design complexity due to the use of multiple buses, guarantee a high level of quality of services offered by the communication infrastructure and scalability. NOCs integrate computer networking concepts for on-chip communication between various processor cores. NOCs are characterized by their topologies, their routing algorithms, switching strategies, network control flow, etc. NOCs include many communication scenarios that evolve concurrently. These scenarios may show some undesirable behaviors such as deadlock, live-lock, starvation, etc. Testing and simulation/emulation based techniques are not sufficient and often results in missing critical bugs. Consequently, a formal validation of the communication correctness will be mandatory. Formal methods [2] have been used to analyze the behavior of systems using mathematical reasoning or rigorous state-space exploration. Approaches to formal verification can be classified into two big categories: model checking and theorem proving. Model checking [3] is a mainstream formal verification method and involves the computer based mathematical modeling of the given system in the form of an automata or state-space. This model is then used within a computer to automatically verify that it meets

rigorous specifications of the intended behavior. Theorem proving [3] uses deductive techniques for verifying the relationship between the logical specification and logical implementation of the given systems. Several works have been formally verified NOC communication using model-checking, theorem proving or a combination between them. In this paper, we present an exhaustive survey of works targeting formal specification and verification of the on-communication for NOCs. Firstly, we introduce some key concepts related to NOCs and define the communication in NOCs. Next, we present the pertinent works on formal specification and verification of the communication inside NOCs. We conclude our paper by a discussion on possible future investigations in the field of formal specification and verification of NOCs.

2 Communication in NOC

Networks On Chips have been proposed as a highly structured and scalable solution to address communication problems in SOC. On-chip interconnection network bring many advantages as high-bandwidth, low-latency, low-power consumption and scalability. With regard to NOC dimension, we can find two big classes that are 2D NOC and 3D NOC. There have been various topologies for 2D NOC architecture. These include mesh, torus, ring, star, binary tree, butterfly, octagon, SPIN (Scalable Programmable Integrated Network) and reconfigurable architectures.

From the communication perspective, communication in NOC is related to many aspects that are: topology, the routing algorithm, the switching technique, the flow control, the arbitration technique, and QoS. The communication between the modules of a NOC is generally categorized as synchronous or asynchronous, with a recent shift towards Globally Asynchronous Locally Synchronous (GALS). Within the synchronous scheme all the modules are synchronized to a global clock in order to achieve their intercommunication. In GALS systems, each module can be synchronized to a private local clock but their intercommunication is established asynchronously, relying on the request and acknowledgement communication handshakes and eliminating the need for a global system clock with its undesirably long interconnect wires. One of the main challenges when designing a NOC is the implementation of an efficient routing algorithm, which is used to determine the path traversed by packets from a source to a destination node. Performance, reliability, energy consumption, and fault tolerance represent just a short list of the major common metrics affected by the routing algorithm. Another important requirement of a routing algorithm is guaranteeing deadlock and livelock freedom. The former, occurs when two or more packets reserve some resources and wait for each other to release the rest of the resources, which never takes place, and all packets involved get blocked and make no progress towards their final destination. The latter, appears when packets wander around and never reach their destination (especially when non-minimal routes are allowed).

3 Formal methods for NOC

Many experts in the field of NOC design have discussed the need for formal methods to generate a design which is correct by construction. By applying formal methods, a very abstract formal model is established then it will be refined over many steps to finally obtain an implementation which is correct. The abstract model can be used for early functional correctness verification, design exploration or non-functional requirements validation. Formal methods have been applied continuously in NOC communication analysis to prove using model checking or theorem proving the absence of some undesirable properties like deadlock, livelock, starvation or the presence of some desirable properties like liveness, security, fairness, etc. or the satisfaction of QoS constraints such as latency, bandwidth, response time or power consumption using some combination of model checking and simulation/static analysis methods. In this context, many formal specification languages and associated tools have been developed at the aim of formal verification of NOC communication and especially the routing. Some of these languages are standard and software-oriented. Some others are not standard but specific to NOC.

3.1 Z notation

The Z notation is an ISO standard, used to specify data-dominated systems. Z formal language belongs to the class of state or model-based formal specification languages. It is based mainly on the axiomatic set theory, lambda calculus and the first order predicate logic. The unit specification in Z is called schema. There exist two schema types: data schema and operation schema. Z schemas can then be combined to form the whole specification. System data are constrained by their invariants and the operations by their pre/post-conditions. Z uses the theorem proving technique to reason about the correctness and the coherence of the system. Authors in [4] presented a Z specification of the routing in a mesh NOC with six routers and asymmetric links between them. The first objective of this work is to evaluate the routing scheme considering latency, status link and shortest route. In order to better understand the dynamism of the operations, and more specifically those related to routing techniques, authors used the animation tool namely Possum.

3.2 Event B

Event B is an extension of the B method to support reactive systems.

Invented by Jean-Raymond Abrial, the B method is a formal method for specification and verification of data oriented systems. The B method follows a successive refinement approach to finally generate a correct by construction program from abstract specifications. There are three kinds of specification unities in B; the abstract machine, the refinement and the implementation. The verification process in B is based on what we call proofs obligations (PO). As Z notation, B language is qualified as a state-based formal specification language. Event B method introduces two new

concepts: the event and the context concepts. In the context of NOCs, Event B was used by many researchers to formally verify the correctness of communication. Authors in [5] used the event B method to design a fault tolerant mesh NOC for System-on-Chip (SOC)-based reconfigurable Field- Programmable Gate Array (FPGA). The formalization process is based on an incremental and validated correct-by construction development of the NOC architecture. Indeed, this work provides a framework for developing the Network-on-Chip Architecture and the XY routing algorithm using essential safety properties together with a formal proof that asserts its correctness. The complexity of the development is measured by the number of proof obligations which are automatically/manually discharged. In another work, Kamali [6] proposed three (3) abstract models namely M0, M1, and M2 at three increasing levels of detail for 3D NOCs. These can be used for modeling and verifying 3D NOC-designs in the early stages of the system development. Each of these models provides templates for communication constraints and guarantees the communication correctness. She showed how to apply such an abstract model to verify a concrete 3D NOC routing algorithm. This is achieved by modeling the correctness condition via invariants; as each model added detail to the previous model, the invariant needed to reflect these added details in a consistent manner. In order for the invariant to be satisfied by a model, a number of proof obligations needs to be discharged. Moreover, in order for the models to respect the refinement relation some other proof obligations need to be generated. As she have employed the RODIN platform to specify the 3D NOC modeling, many of these proof obligations have been automatically discharged, while for the rest it was possible to discharge them interactively. Model checking can also be applied via the model checker Pro-B that is associated to the RODIN platform to verify the liveness property.

3.3 Concurrent Haskell

Concurrent Haskell is an extension of the purely functional language Haskell which expresses concurrency at a high level of abstraction.

Authors in [7] presented an approach which uses Concurrent Haskell to produce concise formal specifications of a real-world NOC that is the SpiNNaker. The latter is specifically for the real-time simulation of large-scale spiking neural networks. It is a multicast communication network. It places a heavy emphasis on fault tolerance. The system is implemented as a 2D array of nodes interconnected through bi-directional links in a triangular formation. The NOC is asynchronous, and serves to decouple the many different clock domains across the chip. Using Concurrent Haskell, SpiNNaker chips are modeled as concurrent processes, communicating with each other via channels which represent the inter-chip links. For the processor-NOC links, three initial assumptions are considered: the NoC-processor links will never fail, the processors and NOC fabric will always process packets fast enough to prevent deadlock across the NOC-processor links and the NOC arbiter will be fair in so much as all input requests will be serviced eventually. This work used a very simple model for the ARM cores, which simply accepts and generates traffic; they are functions that wait to take packets and then emit new ones, with their own address as the source address.

3.4 Promela

PROMELA (PROcess MEta LAnguage) is a high level language to specify systems descriptions supported by the SPIN tool. Authors in [8] presented a new formal model of the Hermes NOC router with five bi-directional ports and the bounded buffers at input port, including the XY routing algorithm, priority based arbitration logic, wormhole switching and the request-ack handshake protocol of the communication scheme using a new formal language called Heterogeneous Protocol Automata (HPA) which is then mapped manually to PROMELA. HPA is a finite state machine with bounded counters that models clocked and clockless systems. The definition allows modeling of event and clock based transitions. Communication interfaces are modeled in HPA as Synchronous Finite State Machines (SFSM) and Asynchronous Finite State Machines (AFSM). The target NOC is a Hermes mesh NOC where packets always take a deterministic route in XY routing. The simulator and model checker inbuilt in the SPIN tool are used for verification. The properties are specified in linear temporal logic (LTL). Using the SPIN model checker, authors verified that:

(a) All the packets sent from source node are received correctly at the right destination node b. (b) The packets flow through a valid path and take the same path every time. A valid path implies that packets are routed only through the routers that are part of the mesh network. In another work [9], authors used PROMELA to specify Programmable Network-On-Chip (PNOC), which is a circuit, switched NOC architecture. The topology contains series of subnets where multiple nodes are connected to a single router through the router port interfaces. A light handshaking mechanism is required to establish or remove connection from a node. This flexible system allows runtime insertion and removal of nodes. Properties including deadlock freedom, starvation freedom, mutual exclusion and liveness are verified using the SPIN model checker. Authors in [10] proposed a new deadlock recovery algorithm and they described a PROMELA model which allows the formal verification for the SOCIN NOC with 2D grid topology. In the PROMELA model, authors changed SOCIN's routing algorithm (originally based on dimensional ordering) and have forced configurations of deadlocks to test the proposed algorithm.

3.5 Timed automata and their extensions

Timed automata are extensions of classical finite automata with clock variables and logical formulas on the clock (temporal constraints). Extended Timed Automata (ETA) extend timed automata by adding non-clock variables (discrete variables) with their corresponding transition functions.

Authors in [11] used timed automata for the modeling and formal verification for the global validation of the behavior of a passive integrated photonic routing structure. Optical Networks on Chip are the alternative solution for Metallic interconnects. This work proposed a two steps methodology. The first step works on high level of abstraction and the second step works on low level of abstraction and precisely on the network segments. The passive transport part is represented by λ-router, which is shared by all the IP blocks to route the data through the network. The routing in the

optical network presented is realized by a 4x4 λ-router. Using UPPAAl, the models were simulated and formally verified. The formal verification consists of checking some properties of the system. The verified properties are: the absence of deadlock, the absence of contention (i.e. simultaneous wavelength can be sent through the network and/or through the router, from the same initiator, in the same time), the verification of the truth table (i.e. there is one and only one wavelength that connects one initiator with one target), and all the locations in the automaton representing the switch are eventually taken. Authors in [12] modeled the router designs of BiNOC: Bidirectional NOC architecture with dynamic self-reconfigurable channel in ETA. BiNOC is a conventional mesh-based NoC design with XY-based wormhole routing. It allows configuring the direction of a channel dynamically. The ETA models are composed into a system state graph, which will be input to the model checker State Graph Manipulator (SGM). The other input to SGM will be the given property, which is specified into the CTL formula. Four crucial properties of the NOC router, namely, mutual exclusion, starvation freedom, deadlock freedom, and conditions for traffic congestions are investigated.

3.6 Lotos

LOTOS (Language of Temporal Ordering Specification) an ISO standard, is an algebraic language based on temporal ordering of events. LOTOS is used for protocol specification. It consists of two parts: a part for the description of data and operations, based on abstract data types, and a part for the description of concurrent processes, based on process calculus. Authors in [13] used value-passing CHP (Communicating Hardware Processes) to generate LOTOS specification. The latter is then verified using the CADP verification toolbox for LOTOS. CHP are a natural approach for the description of asynchronous hardware architectures. These calculi are extensions of standard process calculi with particular synchronization features implemented using handshake protocols. As a case study, authors considered the ANOC (Asynchronous NOC) architecture, which is used as the backbone of Faust, a 4th generation wireless telecom baseband. ANOC implements the GALS paradigm and connects the nodes in a 2D-Mesh topology. The ANOC model in CHP was abstracted. In this case study, authors dealt in particular with the verification of the most complex component of a node of ANOC, namely the input controller with several realistic scenarios. The functional properties describing the protocol behavior of the input controller are: protocol correctness (i.e. the input controller must comply at its inputs with the ANOC protocol, and transmit the incoming data to an output controller), data integrity (i.e. the contents of the communications must be preserved by the input controller), and correct packet routing (i.e. the input controller has to route all the flits of a packet in the right direction). These properties were verified using a generic SVL (Scripting Verification Language) script, calling the model checking and equivalence checking tools of CADP.

3.7 ACL2

ACL2 is a theorem prover and its specifications are written in an applicative subset of Common Lisp and are thus executable. An important feature of ACL2 is to denote both a powerful theorem proving system and an execution engine in the same environment. The theorem proving system has a high degree of automation. Authors in [14] addressed the formal verification of NOCs by means of ACL2. In their previous work, the authors presented the generic network on chip model (GeNoC). It consists of a meta-model of on-chip communication architectures and its implementation in the logic of a theorem proving system. It is a highly generic and parameterized object. The main function of the meta-model is function GeNoC. It takes as main argument a list of messages emitted at source nodes and returns a list of messages received at destination nodes. Function GeNoC produces two output lists: the list of messages that have reached their destination, and the list of messages that are still at their source, or traveling in the network. The model is implemented in the ACL2 theorem prover. The enhanced formalization presented in [14] constitutes a significant progress, both mathematically simpler and offering a much larger expressive power. The methodology is demonstrated on a realistic and state-of-the-art design, the Spidergon network from STMicroelectronics. It is an extension of the Octagon network to an arbitrary even number of nodes. Spidergon forms a regular architecture, where all nodes are connected to three neighbors and a local IP. Twenty three (23) proof obligations were defined. Once the different proof obligations of each component have been proven, the general correctness property is directly obtained from the instantiated proof obligations. This generic aspect of GENOC reduces the verification to discharging proof obligations local to each constituent and provides a compositional approach. Verified instances of the constituents can easily be re-used. Authors in [15] proposed a methodology that supports the specification of parametric NOCs. They provide sufficient constraints that ensure deadlock-free routing, functional correctness, and liveness in a parametric NOC inspired by the HERMES architecture. The latter is based on 2D mesh architecture. The routing policy is based on the XY routing algorithm. HERMES uses the wormhole switching technique. The contribution of this work is the addition of two new theorems to GeNoC. The first one ensures that the routing function is deadlock free. The second one proves that all messages injected in the network eventually leave the network, i.e., liveness. To prove the original GeNoC correctness theorem, one has to prove that the routing function terminates and ends in the correct destination. This only proves that computing one route for one message is correct when there is no other message in the network. Deadlock and evacuation are global properties that depend on the interaction of several messages. All theorems were implemented using the ACL2 'Sedan', an Eclipse interface to the ACL2 theorem prover.

3.8 DEVS based model checking

Discrete Event System Specification (DEVS) is a formalism for specifying modular, hierarchical coupled models composed of atomic models suited for simulation

purposes. Authors in [16] presented a DEVS-based model checking verification method for NOC models. For this purpose, a constrained version of the atomic DEVS modeling formalism is formulated and applied to verification of an NOC router. This is achieved by adding constraints on the state size and the number of transitions in the atomic model. A model-checker is introduced into the DEVS-Suite simulator for verifying constrained DEVS models. In order to demonstrate how the modeling and implementation of constrained DEVS models are carried out for NOC components, authors modeled and verified an atomic router model in a mesh 5x5 NOC using the DEVS-Suite simulator. The authors used Java to implement a data collector using Experimental Frame (EF) in the DEVS-Suite simulator that gives the designer flexibility and support for creating experiments and calculating measurements based on the states of models. In the proposed model-checking framework, the generator injects various combinations of inputs to the model and the transducers are charged with the task of gathering data from the model. These transducers can collect state data as well as data derived from states and check them against some desired properties.

3.9 CSP and FSP

CSP (Communicating Sequential Processes) is a formal language for describing patterns of interaction in concurrent systems. It belongs to the family of mathematical theories of concurrency known as process algebras, or process calculi, based on message passing via channels. In the context of a master thesis [17], the author formally verified the OASIS NOC using the model checking technique. The OASIS NOC is a complexity effective on-chip interconnection network designed and prototyped on a FPGA. The OASIS NOC is a 2-D 4×4 mesh topology. Both refinement model checking and probabilistic model checking techniques are used to verify the OASIS NOC properties. The OASIS NOC is first formalized in CSP and then verified with the FDR (Failure-Divergence Refinement) model checker for deadlock freedom. FDR is the de facto model checker for the CSP language from Formal Systems Europe Limited [FSEL]. Its method of establishing whether a property holds is to test for the refinement of a transition system capturing the property by the candidate machine. The author also showed that PRISM model checker which is designed for verifying probabilistic properties can be used to verify non probabilistic properties as deadlock freedom. The behavior of a system to be verified by PRISM model checker is specified using a simple module-based language inspired by Reactive Modules formalism of Alur and Henzinger. The fundamental components of the PRISM language are modules and variables. The verification result of both FDR and PRISM shows that the OASIS NOC is free from deadlock. Using PRISM, the OASIS NOC behaves as a message buffer; receiving and delivering flits without any losses.

Authors in [18] presented a systematic approach that models concurrent processes in a mesh NOC using Finite State Process (FSP); this model is exhaustively tested with Labeled Transition State Analyzer (LTSA) tool developed by Jeff Magee and Jeff Kramer. FSP (Finite State Processes) is a process specification language based closely on the CSP. The authors proposed a set of rules that should be followed for modeling a NOC concurrent system.

3.10 SVA

SVA (System Verilog Assertions) is an assertion-based language offering a very powerful way to describe design properties and temporal behaviors. An assertion is a statement that validates assumptions or checks conditions in a program. It can be seen as an observer that observes the state of the program and, if built that way, can block further execution of the code, but it cannot alter the program itself. Assertions are primarily used to validate the behavior of a design. They may also be used to provide functional coverage information for a design. Assertions can be checked dynamically by simulation, or statically by a model checker. In System Verilog, there are two kinds of assertions: immediate (assert) and concurrent (assert property). Coverage statements (cover property) are concurrent and have the same syntax as concurrent assertions. Authors in [20] presented MCENOC, a Network-on-Chip (NoC) switching architecture with predictable, formally verifiable timing behavior. Firstly, they used Benes networks to model the MCENOC. A Benes network is a special case of the Clos network; it refines the Clos concept into a topology of 2x2 switching elements. It retains the same properties, but is constructed from the smallest possible switch size. A Clos network is a kind of multistage circuit switching network used in the field of telecommunications. Using SystemVerilog Assertions (SVA), formal properties are defined helping the refinement of the specification of the design as well as enabling the implementation to be exhaustively formally verified. The authors demonstrated the performance of the design in terms of size, throughput and predictability, and discussed the application level considerations needed to exploit this architecture.

Properties are defined in SVA, and then asserted for each of the ports or instances defined in any configuration of the system. Checking of these properties is performed using the Cadence JasperGold formal verification tool.

3.11 PVS

PVS is a verification system, combining language expressiveness with automated tools. The language is based on higher-order logic, and is strongly typed. It includes types and terms such as: numbers, records, tuples, functions, quantifiers, and recursive definitions [21]. The PVS prover is interactive, but with a large amount of automation built in. PVS includes an integrated model checker that is based on the μ-calculus. To use the model checker, a finite transition system must be defined with an initial predicate and a transition relation. The PVS prelude provides CTL temporal operators, as well as several definitions of fairness. Authors in [22] discussed a formal specification of the ÆTHEREAL protocol and its underlying network in terms of the PVS specification language. The ÆTHEREAL protocol has been proposed by Philips to enable both guaranteed and best-effort communication in an on-chip packet switching network. Using PVS the authors proved the absence of deadlock for an abstract version of the model.

3.12 ABS

ABS is a formal, executable modeling language for concurrent and distributed systems. It combines functional, imperative, and object-oriented programming styles allowing intuitive, modular, high-level modeling of concepts, domain and data.

ABS models are fully executable and model system behavior precisely. ABS can model synchronous as well as asynchronous communication. ABS has been developed to provide the foundations for scalable formal verification: there is a program logic as well as a compositional proof system that makes possible to prove global system properties by reasoning about object-local invariants; ABS comes with an IDE and a range of analysis as well as productivity tools, specifically, there is a formal verification tool called KeY-ABS. Authors in [23] developed an approach with a case study of a Network-on-Chip packet switching platform. They provided an executable formal model in ABS of a generic MxN mesh chip with an unbounded number of packets and verified several crucial properties. Their concern was formal verification of unbounded concurrent systems. They showed how scalable verification can be achieved by compositional and local reasoning about history-based specifications of observable behavior. The approach is realized in the proof system KeY-ABS. They demonstrated the viability of the verification approach by proving the correctness of safety properties for an ABS model of an ASPIN (Asynchronous Scalable Packet Switching Integrated Network) NOC of arbitrary, unbounded size. Formal proofs of global properties such as “no packets are lost" and “a packet is never sent in a circle" were reported.

3.13 Petri Nets and their extensions

Generally speaking, Petri Nets have been used to model and simulate traffic in NOC systems. Authors in [25] presented formal models for multicast traffic in NOC architectures. They formalized multistage interconnection networks with semantics inspired by the high-level version of the Petri box algebra, which allows one to represent concurrent communication systems in a compositional way.

Authors in [26] applied deterministic and stochastic Petri-Nets (DSPNs) to model on-chip communication. In their contribution, the modeling of basic NOC communication scenarios featuring different processor cores, network topologies and communication schemes is presented. In order to provide a test bed for the verification of modeling results a state-of-the-art FPGA-platform has been utilized. This platform allows to instantiate a soft-core processor network, which can be adapted in terms of communication network topologies and communication schemes. Quantitative results for modeling effort and computational complexity were presented. Furthermore, a more complex hierarchical NOC has been modeled and the influence of parameters like the distribution of read and writes accesses has been studied.

3.14 PSL

PSL (Property Specification Language) has emerged as one of the standard assertion languages and is on its way to becoming an IEEE standard. PSL is designed to capture design intent in an executable, formal, unambiguous manner. Authors in [27] presented a methodology to use assertions in NOC designs to facilitate debugging, monitoring, and ensuring reliable and failure-free operation. They relied on their own assertion-checker generator (MBAC) to produce efficient RTL-level checkers from high-level temporal assertions, with optional debugging features. The MBAC assertion compiler generates hardware that only monitors the circuit behavior, guaranteeing that the hardware will behave as originally intended. Furthermore, automatically translating PSL assertions into hardware checkers at RTL code eliminates the risk of introducing errors into the assertion circuitry itself, which is likely to occur if the translations were done manually. To optimize the usefulness of the assertion checkers, they applied small modifications on the existing NOC to provide the chronology of errors. To this effect, they introduced a NOC-level high-priority messaging system.

3.15 SDL

The Specification and Description Language (SDL) is a specification language targeted at the unambiguous specification and description of the behavior of reactive and distributed systems, defined by ITU-T in Recommendations Z.100 to Z.106, originally focused on telecommunication systems.

Authors in [30] described the design of a NOC simulator using SDL. Features of SDL for representing structural hierarchy using blocks, concurrent processes and dynamic generation of processes, communication channels of user defined data types and timers are useful for modeling a NOC architecture at various levels of communication protocols. The event driven SDL simulator carried out interesting experiments to evaluate various architectural options like buffer size in switches, and their effect on the performance like delay and packet loss. The target NOC is the KTH-VTT NOC Architecture consisting of a two-dimensional mesh of switches or micro-routers. Cores (called resources) are placed in the slots formed by the switches. To have flexibility as well as efficiency of simulation, authors provided a mode switch in the model so that it is possible to simulate the model either at data-link layer or at the network layer. Since data-link layer does not add information on how the routing and buffers work, it would be possible to leave this layer out when making evaluations of network performance at network layer level.

3.16 Timed Rebeca

Timed Rebeca is an extension of Rebeca formal specification language, capable of modeling functional and timing behaviors of distributed reactive systems.

Rebeca is an actor based modeling language with a Java-like syntax. Actors can be considered as a reference model for concurrent computation. A Rebeca model consists of reactive classes and a main part that contains instantiation of reactive objects (re-

becs) from reactive classes. Authors in [31] targeted maximum end-to-end packet latency for comparison of different routing algorithms. Network topology, router buffers, routing algorithm, communication policy, storage strategy and channels are modeled. Timing behaviors like link delay and the delay for writing and reading to/from buffers are also considered in the model. Using an actor based modeling language; they efficiently mapped the constituents of GALS NOC to actor model. To this end, a formal model for GALS NOC was presented using high level modeling language Rebeca. The model was then used for comparison between three routing algorithms, namely XY (deterministic), Odd-Even (adaptive) and DyAD (dynamically adaptive and deterministic) with respect to the maximum end-to- end packet latency. Results of comparison were presented under two different traffic patterns and showed that under distributed traffic a deterministic routing could better work. However, in a directed traffic -that is of more interest in real applications- adaptive routing algorithms are better. The routing performance results obtained through Rebeca model checking confirm the same previously published results in simulations.

4 Discussion

According to our analysis, we can state that:

- There is a panoply of formal specification languages and tools for NOC communication verification.
- Most of these languages and associated tools are not standard, consequently their interoperability is infeasible.
- Most of related works verify packet switching routing in 2D mesh topology and XY routing algorithm.
- A lack in formal methods for circuit switching, other topologies expect the mesh one, and importantly 3D NOCs.

Model checking suits well control-dominated systems (i.e. systems whose behavior can be expressed as state-space), it brings many advantages due to its automatic, and hence user-friendly, nature. Moreover, the ability to provide counter examples in case of failures makes model checking a more preferable choice for industrial usage as compared to the other interactive formal verification approaches like theorem proving. However, model checking suffers from the space explosion problem and its applicability is restricted to finite state systems. In the literature, many solutions have been proposed to solve the big problem of the state explosion, among them, we find notably symbolic, probabilistic and distributed model checking. The latter solution seems to be very promising. On the other hand, theorem-proving suits well data-dominated systems and appropriate for infinite systems. A major challenge with theorem proving is its automation especially for systems using higher order logic. Most of theorem provers are interactive. Since NOCs are generally, data/control dominated system, a combination of both model checking and theorem proving seems a promising solution. Amjad [34] investigates the idea of combination between model checking and theorem proving and apply it to verify the AMBA protocol. Complement

formal verification techniques with trace-based simulation is also an interesting solution and many works have investigated this idea.

In terms of formal specification languages for NOCs, authors have proposed many formalisms and theories from both software and hardware communities. Here, we discuss some of these languages especially the well-known ones and highlight their appropriateness to NOC modeling and verification.

Z language is a too abstract and does not support hardware aspects such as timing, reactivity and concurrency, synchronization, etc. Furthermore, the refinement in Z is done manually. In its current form, we see that Z is not very suitable for NOC verification. Comparatively to Z, Event B seems to be more appropriate for NOC verification since it supports many hardware aspects at higher level of abstraction like reactivity and concurrency. The most important in Event B is the refinement and automatic PO (proofs obligation) generation and verification. A clear drawback in Event B is the lack of timing and QoS support. PROMELA, which is a C like language, does not offer the necessary abstraction for NOCs modeling. PROMELA is qualified as a transition-based language. Petri Nets and their variants match well the formal quantitative analysis of NOCs. SDL and due to its graphical notations and simulation/formal verification capacities are good candidates for NOC verification. PSL and SVA are assertion-based languages and very suitable for the formal verification of the hardware part of NOCs. Establish a link between formal techniques and state of the art system level standard languages for NOCs, such as SystemC may be an alternative good solution. In this context, authors in [35] proposed to integrate model checking into SystemC. Authors in [36] models SystemC fixed-point arithmetic in the theorem prover HOL. Similarly to the standard SystemC, authors have defined a sub set of VHDL and defined its semantics in the theorem prover ACL2 [37], or in a certain model checker as CV [38]. Boutekkouk [39] formalized a sub set of VHDL into Maude language. The latter is based on equational algebra and rewriting logic theory. The idea of SystemC/VHDL formal verification suits well legacy code reuse. However, this idea deals with low level models and consequently it consumes much time for verification. Abstraction of these models can alleviate this problem but at the prize of imprecise results. ESTEREL and SIGNAL are two synchronous languages that seem very appropriate to model and verify ANOC, since they implement well the GALS philosophy [40]. The Architecture Analysis & Design Language (AADL) is another good candidate for NOC simulation and formal verification. AADL has been used to model both software and hardware architecture of an embedded, real-time system. AADL includes many tools for simulation, performance estimation and formal verification using model-checking [41]. What we need is a standard language for NOC specification and verification. This language has to satisfy the following requirements:

- More friendly and supports well the graphical notations.
- Multi-paradigm (i.e. supports a variety of Models of Computation and semantics)
- Has the capacity to model and refine both software and hardware through different levels of abstraction.
- Generic enough to model different topologies (2D, 3D), different routing, switching and control flow techniques.

- Supports both model checking, theorem proving and trace-based simulation integrated into one unified environment.
- Supports tools for integration and interoperability with other tools and existing state of the art formal languages for NOCs.

5 References

[1] Agarwal, A., Iskander, C. and Shankar, R. (2009). Survey of Network on Chip (NoC) Architectures & Contributions. Journal of Engineering, Computing and Architecture, Vol.3 (1).

[2] Wing, J.M. (1994). Formal Methods. In Encyclopedia of Software Engineering, John Wiley & Sons, Inc., New York, pp. 504-517. Revision in Second Edition.

[3] Ouimet, M. (2007). Formal Software Verification: Model Checking and Theorem Proving, Embedded Systems Laboratory Technical Report ESL-TIK-00214.

[4] Ramos, K.D.N., Ribeiro, C.M.F.A., Moreira, A.M. and Silva, I.S. (2006). A Formal approach for Network-On-Chip design. In iiWAS2006: The 8th International Conference on Information Integration and Web-based Application & Services, 4 - 6 December, Yogyakarta Indonesia.

[5] Andriamiarina, M.B., Daoud, H., Belarbi, M., Méry, D. and Tanougast, C. (2012). Formal Verification of Fault Tolerant NoC-based Architecture. In IWMCS2012: First International Workshop on Mathematics and Computer Science, December, Tiaret, Algeria.

[6] Kamali, M., Petre, L., Sere, K. and Daneshtalab, M. (2011). Refinement-Based Modeling of 3D NoCs. In FSEN 2011: International Conference on Fundamentals of Software Engineering, April 20-22, Tehran, Iran.

[7] Lester, D. and Richards, D. (2008). Specification of a network-on-chip. In Proceedings of the 20th UK Asynchronous Forum (UK-ASYNC).

[8] Palaniveloo, V.A. and Sowmya, A. (2011). Application of Formal Methods for System-Level Verification of Network on Chip. In 2011 IEEE Computer Society Annual Symposium on VLSI, 4-6 July, Chennai, India. https://doi.org/10.1109/ISVLSI.2011.57

[9] Zaman, A. and Hasan, O. (2014). Formal verification of circuit-switched Network on chip (NoC) architectures using SPIN. In International Symposium on System-on-Chip (SoC), 28-29 October, Tampere, Finland. https://doi.org/10.1109/ISSOC.2014.6972449

[10] Otero, J.C., Dillenburg, F.C., Ribeiro, L. and Wagner, F.R. (2011). Formal Verification of a Deadlock Recovering Algorithm for Heterogeneus NoC Routing Support. Available at ftp://ftp.inf.ufrgs.br/pub/simoo/papers/dsnoc11.pdf

[11] Gheorghe, L., Nicolescu, G. and IO'Connor, I. (2009). Modeling and formal verification of a passive optical Network On Chip behavior. In ECEASST 21, Electronic Communication of the European Association of Software Science and Technology, Vol. 21.

[12] Chen, Y.R, Su, W.T., Hsiung, P.A, Lan, Y.C, Hu, Y.H. and Chen, S.J. (2010). Formal Modeling and Verification for Network-on-Chip. In The 2010 International Conference on Green Circuits and Systems, 21-23 June Shanghai, China. https://doi.org/10.1109/ICGCS.2010.5543050

[13] Garavel, H., Salaün, G. and Serwe, W. (2009). On the semantics of communicating hardware processes and their translation into LOTOS for the verification of asynchronous circuits with CADP. Journal of Science of Computer Programming, Vol.74 Issue 3, pp.100-127. https://doi.org/10.1016/j.scico.2008.09.011

[14] Borrione, D., Helmy, A., Pierre, L. and Schmaltz, J. (2009). A Formal Approach to the Verification of Networks on Chip. EURASIP Journal on Embedded Systems, Vol.2009. https://doi.org/10.1155/2009/548324
[15] Verbeek, F. and Schmaltz, J. (2010). Formal Specification of Networks-on-Chips: Deadlock and Evacuation. In DATE 2010: Design, Automation & Test in Europe Conference & Exhibition, 8-12 March, Dresden, Germany.
[16] Gholami, S. and Sarjoughian, H.S. (2017). Modeling and verification of Network-On-Chip using Constrained-DEVS. In TMS/DEVS '17, Proceedings of the Symposium on Theory of Modeling & Simulation, 23 – 26 April, Virginia Beach, Virginia, USA.
[17] Kwaku, B.S. (2009). Formal verification of a Network On Chip, a Master thesis in computer science, submitted to the African University of Science and Technology, December, Abuja, Nigeria.
[18] Agarwal, A. and Shankar, R. (2006). Modeling Concurrency in NOC for Embedded Systems. In High Performance Embedded Computing (HPEC) Workshop, 19-21 September, Massachusetts Institute of Technology, MIT Lincoln Labs, Boston, MA, USA.
[19] Gastel, B.V. and Schmaltz, J. (2013). A formalisation of XMAS. In ACL2 2013: Workshop on the ACL2 Theorem Prover and its Applications, May 30-31, Laramie, Wyoming, USA. https://doi.org/10.4204/EPTCS.114.9
[20] Kerrison, S., May, D. and Eder, K. (2016). A Benes Based NoC Switching Architecture for Mixed Criticality Embedded Systems. In MCSOC: IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip, 21-23 September, Lyon, France. https://doi.org/10.1109/MCSoC.2016.50
[21] Owre, S. and Shankar, N. (2008). A Brief Overview of PVS. In TPHOLs '08: the 21st International Conference on Theorem Proving in Higher Order Logics, 18-21 August, Montreal, P.Q., Canada. https://doi.org/10.1007/978-3-540-71067-7_5
[22] Gebremichael, B., Vaandrager, F., Zhang, M., Goossens, K., Rijpkema, E. and Radulescu, A. (2005). Deadlock Prevention in the ÆTHEREAL Protocol. In CHARME'05: 13 IFIP WG 10.5 international conference on Correct Hardware Design and Verification Methods, 3-6 October, Saarbrücken, Germany. https://doi.org/10.1007/11560548_28
[23] Din, C.C, Tarifa, S.L.T, Hahnle, R. and Johnsen, E.B. (2015). History-Based Specification and Verification of Scalable Concurrent and Distributed Systems. In ICFEM 2015: 17th International Conference on Formal Engineering Methods, 3-5 November, Paris, France.
[24] Holcomb, D.E. (2013). Formal Verification and Synthesis for Quality-of-Service in On-Chip Networks, A PhD dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Electrical Engineering and Computer Sciences in the Graduate Division of the University of California, Berkeley.
[25] Pelz, E. and Tutsch, D. (2007). Formal Models for Multicast Traffic in Network on Chip Architectures with Compositional High-Level Petri Nets. In ICATPN 2007: 28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, 25-29 June, Siedlce, Poland. https://doi.org/10.1007/978-3-540-73094-1_23
[26] Blume, H., von Sydow, T., Becker, D. and Noll, T.G. (2005). Modeling NoC Architectures by Means of Deterministic and Stochastic Petri Nets. In Embedded Computer Systems: Architectures, Modeling, and Simulation. SAMOS2005. Lecture Notes in Computer Science, vol. 3553.
[27] Chenard, J.S., Bourduas, S., Azuelos, N., Boule, M. and Zilic, Z. (2007). Hardware Assertion Checkers in On-line Detection of Faults in a Hierarchical-Ring Network-On-Chip. In DATE 2007: Design Automation and Test in Europe Workshop on Diagnostic Services in Networks-on-Chips, Apvril 20, Nice, France.

[28] Parikh, R., and Bertacco, V. (2011). Formally Enhanced Runtime Verification to Ensure NoC Functional Correctness. In MICRO-44: the 44th Annual IEEE/ACM, International Symposium on Microarchitecture, 3-7 December, Porto Alegre, Brazil.

[29] Dridiy, M., Lallaliy, M., Rubiniy, S., Singhoffy, F. and Diguet, J.P. (2017). Modeling and Validation of a Mixed-Criticality NoC Router Using the IF Language. In NoCArc'17: the10th International Workshop on Network on Chip Architectures, 14 October, Cambridge, MA, USA. https://doi.org/10.1145/3139540.3139543

[30] Holsmark, R., HÄogberg, M. and Kumar, S. (2003). Modelling and Evaluation of a Network on Chip Architecture using SDL. In SDL '03: System Design, 11th SDL Forum, 1-4 July, Stuttgart, Germany.

[31] Sharifi, Z., Mohammadi, S. and Sirjani, M. (2013). Comparison of NoC Routing Algorithms Using Formal Methods**.** In PDPTA'13: the International Conference on Parallel and Distributed Processing Techniques and Applications, 22-25 July, 2013, Las Vegas, USA.

[32] Zhang, Z., Serwe, W., Wu, J., Yoneda, T., Zheng, H. and Myers, C. (2016). An Improved Fault-Tolerant Routing Algorithm for a Network-on-Chip Derived with Formal Analysis. In Science of Computer Programming Journal, Vol. 118, pp.24-39. https://doi.org/10.1016/j.scico.2016.01.002

[33] Zhang, Z. (2016). *Verification Methodologies for fault-tolerant Network-On-Chip systems*, A PhD dissertation submitted to the faculty of the University of Utah, Department of Electrical and Computer Engineering, the University of Utah.

[34] Amjad, H. (2004). *Combining model checking and theorem proving*, Technical Report Number 601, UCAM-CL-TR-601, ISSN 1476-2986.

[35] Cimatti, A. Narasamdya, I. and Roveri, M. (2013). Software Model Checking SystemC. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* ,Vol.32, Issue 5. https://doi.org/10.1109/TCAD.2012.2232351

[36] Akbarpour, B. and Tahar, S. (2003). Modeling SystemC Fixed-Point Arithmetic in HOL. In ICFEM: 5th International Conference on Formal Engineering Methods, 5-7 November, Singapore.https://doi.org/10.1007/978-3-540-39893-6_13

[37] Georgelin, P., Borrione, D. and Ostier, P. (2002). A framework for VHDL combining theorem proving and symbolic simulation. In ACL2'02: the Third International Workshop on the ACL2 Theorem Prover and its Applications, March, Grenoble, France.

[38] Deharbe, D., Shankar, S. and Clarke, E.M. (1998). Formal verification of VHDL: the model checker CV. In XI Brazilian Symposium on Integrated Circuit Design, 3 October, Rio de Janeiro, Brazil. https://doi.org/10.1109/SBCCI.1998.715418

[39] Boutekkouk, F. (2012). Maude specification generation from VHDL. In AIM 2012: the Second International Conference on Advances in Information Technology and Mobile Communication, 27-28 April, Bangalore, India.

[40] Amde, M., Felicijan, T., Efthymiou, A., Edwards, D. and Lavagno, L. (2005). Asynchronous on-chip networks. In Computers and Digital Techniques, Vol.152, Issue 2, pp.273–283. https://doi.org/10.1049/ip-cdt:20045093

[41] Pellizzoni, R., Meredith, P., Nam, M.Y., Sun, M., Caccamo, M. and Sha, L. (2009). Handling Mixed-Criticality in SoC-based Real-Time Embedded Systems. In EMSOFT '09: the seventh ACM international conference on Embedded software, 12-16 October, Grenoble, France. https://doi.org/10.1145/1629335.1629367

6 Author

Fateh Boutekkouk is a senior lecturer in informatics at the University of Oum El Bouaghi-Algeria- and a member in ReLaCS2 laboratory. He works on embedded systems and Networks On Chip (NOC) modeling, performance estimation and formal verification.

Article submitted 24 August 2018. Resubmitted 13 August 2018. Final acceptance 15 August 2018. Final version published as submitted by the author.