Based refinement Verification platform for QNoC Architectures

H.DAOUD, M.BELARBI.
IBN Khaldoun University, LIM
hayat.daoud@hotmail.fr, master.dept.inf@gmail.com

***Abstract*—Formal models play an important role of the requirements that lead to models of the design for a network on chip which is a reconfigurable FPGA-based (Field Programmable Gate Array) technology for faulty tolerance System-on-Chip, where the main challenge was how to achieve a conceptual design of multiprocessor System On Chip (MPSoC). The use of formal methods with the progressive basis and the proof theory has become an essential step to design and validate this architecture. Event-B is a formal modelling language, which supports refinement as a based-formal concept of development to models and proves the industry of MPSoCs. The purpose of this article is to provide a formal verification of Network-On-Chip (NoC) architecture using the Event-B method. This process is delivered by a correct and validated formalization based on the correct-by-construction development approach.**

***Index Terms*—Network on chip, Switch, Adaptive-routing, machine, context, Model, specification, refinement, Formal proof, Correct-by-construction, Active Zone.**

I. INTRODUCTION

Formal methods have the ability to produce critical systems for large industrial projects, and this by creating an original mathematical model that can be formally refined in levels until the final refinement that contain enough of details for an implementation. Before the verification simulation does not allow the detection of all possible design errors [1]. That's why we use the formal methods Event-B in our work, and in particular the correct-by-construction paradigm [2, 3] to specify hardware systems. The paradigm correct-by-construction offers an alternative approach to prove and define correct systems and architectures, for the reconstruction of a target system using progressive refinement and validated methodological techniques [4]. Our goal is to complete the simulation time in the design flow with a formal proof method. The preconditions for the formal development of microelectronics architecture are given the description and /or design of the architecture.

The large amount of work has focused on the use of formal methods to verify communication systems and protocols. Most use model-checking, or its composition with proving theorems. The work of Clarkeet al, published in [5] to check the temporal properties of parameterized ring networks and binary tree. A first step is to use a free-context grammar to models network communication systems when temporal properties are verified using a model checker. In [6] Amjaduses a model checker implemented in HOL to verify AMBAAHB protocols and PDB. Bharadwajet al. satisfies a broadcast protocol in a binary tree network using the SPIN model checker demonstrator and Coq [7]. In [8] Curzondevelopsa structural model of ATM switch Fairsile and compares its behavioral specification using HOL. The free deadlock in the network Ae the real was verified by Gebre Michael and al. Using the PVS tool [9] Some studies based on semi-formal methods were also proposed. They essentially designed to detect and debug failures. Chenard and al proposed in [10] include assertions listeners PSL [11] synthesized using NGC tool [12] in a network on chip. Analytical approaches do not carry out of the dynamic

behaviour and performance of a system, but to analyze it statically. Model checking is an automated technique to verify each models of a system satisfies its specification.

[13] The model is described in a kind of state machine and the specification is described in a temporal logic. A model control algorithm uses the transition function associated with the state machine to explore the state space and define States that do not meet the specifications. If finds is a state, the state and the trace leading to this state are reported. If such a state is not found, the system is proved correct. Model checking is widely adopted by universities and industry, primarily because it is fully automatic and can provide against-examples. The major problem is a combinatorial blow-upin the number of states that must be explored, called state space explosion. This severely limits the scalability of model checking. Theorem proving is a technique where the evidence of a mathematical theorem is formalized so that a computer program can guarantee their accuracy. The main advantage of the theorem is the ability to deal with the parametric systems.

The aim of this work is the verification of SoC communication [14] describes the main challenges in the design of NoC [15] and discusses some aspects of audit networks or formal methods are useful. The dynamic reconfigurable NoC are adequate for FPGA-based systems, where the main problem arises when IP (intellectual property) components must be at run time defined dynamically. Given the rapidly changing and highly complex MPSoCs (multiprocessor system-on-chip), the constraints related to the complexity and the increasing number of interconnected modules or IP such as the cost and performance must be resolved. Current communication networks on chips implement the data transmission between the interconnected nodes. Sometimes the communication of this kind of networks is difficult or even impossible. This is the main reason why XY fault-tolerant routing algorithms (such networks) have been established. [16] Routers can control the miss-routing of previous detectors (eg packet on the path XY, etc). In addition, new techniques and adaptive faulty-tolerance routing with error detection and path routing based on the well-known XY model, have been introduced.

Formal studies have focused on NoC performance [17,18], latency [19], bandwidth [20], the estimation of consumption[20], detection and error correction[21,22], and the surface are used. Others propose methods of free-deadlock routing [23, 24] to characterize the traffic. [25] In this article, we use Event-B to specify, verify and demonstrate the NoC behavior. The paper is organized as follows. Section 2 presents an overview of the Event-B method. Section 3 presents the NoC architecture studied with the audit results of the verification Formula. Section 4 describes the architecture of the faulty tolerance. Section 5 Model description and we concludes this paper with future works

II. EVENT-B

The Event B modeling language can express safety properties [26], which are invariants, theorems or safety properties in a machine corresponding to the system. Event B allows a progressive development of models through refinements. The two main structures available in Event B are:

- Contexts express static information about the model.
- Machines express dynamic information about the model, invariants, safety properties, and events.

An Event B model is defined either as a context or as a machine. A machine organizes events (or actions) modifying state variables and uses static information defined in a context. The refinement of models provides a mechanism for relating an abstract model and a concrete model by adding new events or variables. This feature allows to develop gradually Event-B models and to validate each decision step using the proof tool. The refinement relationship should be expressed as follows: a model M is refined by a model P, when P simulates M. Thus, from a given model M, a new model P can be built and asserted to be a refinement of M describing the architecture. Model M is an abstraction of P, and model P is a refinement (concrete version) of M. Likewise, context C, seen by a model M, can be refined to a context D, which may be seen by P. The final concrete model is close to the behavior of real system that executes events using real source code. The relationships between contexts, machines and events are illustrated by the next diagrams, which consider refinements of events and machines. The refinement of a formal model allows us to enrich the model via a step-by-step approach and is the foundation of our correct-by-construction approach [27]. Refinement provides a way to strengthen invariants and to add details to a model. It is also used to transform an abstract model to a more concrete version by modifying the state description. This is done by extending the list of state variables (possibly suppressing some of them), by refining each abstract event to a set of possible concrete versions, and by adding new events. In fact, the refinement-based development of Event B requires a very careful derivation process, integrating possible tough interactive proofs for discharging generated proof obligations, at each step of development.

Event B also is supported by a complete toolset RODIN [28] providing features like refinement, proof obligations generation, proof assistants and model-checking facilities. Rodin Platform tool, called Proof Obligation Generator, decides what is to be proved in order to ensure the correctness of the model. Moreover, it is now being improved and extended by other "plug-ins" [26].

III. ARCHITECTURE DESCRIPTION

A QNoC Switch (see Figure 1) [29] consists of routing logic and control logic with inputs / outputs each direction. This micro-electronic architecture communicates with four neighboring elements.

The computing elements associated with the NoC network communicate through messages. A message consists of a fixed number of packets.

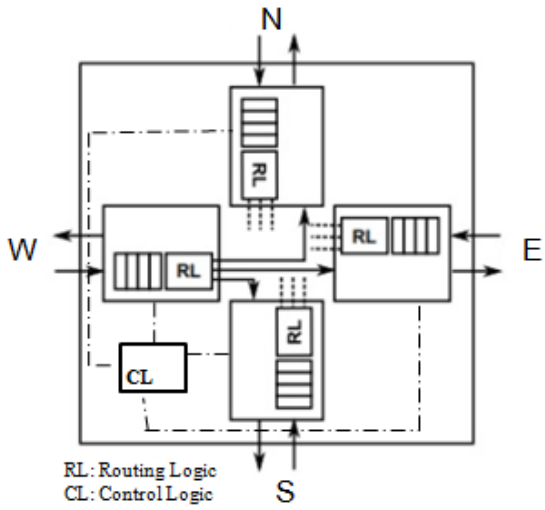


Fig.1. QNoC Switch

An incremental development of a Network-on-chip Architecture using the event B formalism. [25] The formalization of the architecture is presented from an abstract level to a more concrete level in a hierarchical way.

The first model ***xyM0*** is an abstract description of the service offered by the NoC Architecture: the sending of a packet (p) by a switch source and the receiving of (p) by a switch destination.

The machine xyM1 refines ***xyM0*** and introduces a network (*a graph*) between the sources and destinations of packets. Some properties on the graph are defined in context ***xyC1***: graph is non-empty, non-transitive and is symmetrical.

The second refinement decomposes the event FORWARD of ***xyM1*** into two events:

- A refinement of the event FORWARD depicts the passing of a packet (p) from a switch (x) to a channel (ch), leading to a neighbour (y).
- An event FROM_CHANNEL_TO_NODE models the transfer of a packet (p) from a channel (ch) to a connected switch (n).

The third refinement allows us to introduce the structure of a switch gradually. We express, in xyM13, that switches possess output ports.

The fourth refinement (xyM14) adds input ports to the structure of a switch.

The fifth refinement introduces the storage of packets in a switch: each output port of a switch can store a number of packets up to a limit (outputplaces) of three messages. Packets can be blocked in a switch, because of the "wait" or "occupation" signals from neighbours. The event SWITCH_CONTROL is refined, and adds the fact that following the transition of a packet from an input port of a switch (x) to an output port, if the switch (x) is not busy anymore, it sends a release signal to the previous switch linked to the input port. A new event RECEIVE_BUFFER_CREDIT models the receiving of a release signal by a switch (n).

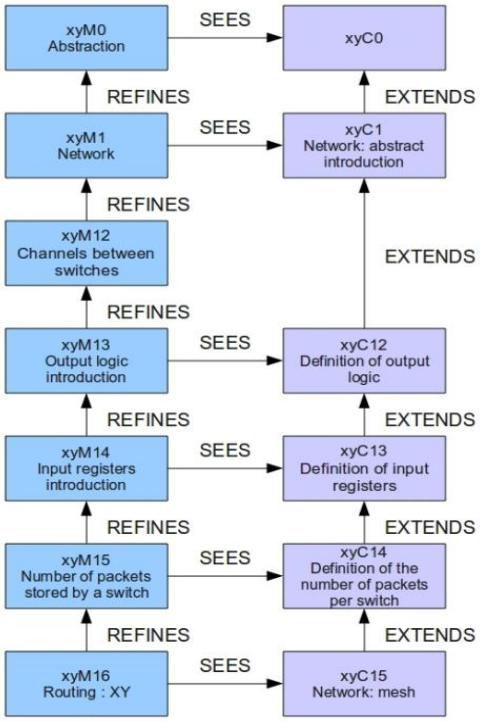


Figure 5. *Step-by-step Modeling of NoC Architecture*

The last model xyM16 describes the architecture of the network (graph): graph has a mesh topology (see Figure.12). A numerical limit (nsize) is introduced to bound the number of routers in the dimensions x and y of the network topology; the network will be a regular 2D-Mesh, with a size (nsize _ nsize); each switch is coupled with unique coordinates (x; y), with x 2 [0::nsize 1] and y 2 [0::nsize 1].

IV. Fault Tolerance

A. The adaptive routing algorithm

The Switch produced in stating that packets are routed along the X axis then to the Y-axis direction of the network. If in the routing packets encounter modules that prevent them to go through the traditional way then the routing algorithm used allows circumvention with the control logic for each router distinguishes the entity type (router or calculation module) connected to a router. So this algorithm avoids deadlock situations that may happen in the QNoC and also solves the problem of packet arrival at a network node order. Indeed, if during operation of the network, there is no dynamic investments between two compute modules, the paths taken by packets sent from one module to another recipient module will be identical and of the same length (same position as the algorithm XY), while maintaining the order of transmission and reception of packets. As against, the packets of a message sent by a calculation module, can be nested to the destination with the packets of other messages sent from other computing modules in

the network. Uses a routing algorithm based on the classical XY algorithm that can be used initially in the reconfigurable network because it is not suited to irregular situations.

B. definition

A disabled network region is the rest of the network not belonging to the active area.

If a network does not have an activated area (no faulty nodes or regions), it is fully disabled. All nodes belonging to the disabled area are disabled.

If there is a network in an active region formed around a failed node. In this case, only the routing nodes surrounding the failed node routing change status and become activated. The nodes belonging to the rest of the network do not change their way and remain disabled.

A deactivated node routes a data packet according to the XY algorithm. First, it routes the packet according to the X axis and then along the Y axis until the data packet is not delivered to the destination. If the packet arrives to the activated before reaching its final destination area, new routing rules are then applied.

The activated routing nodes do not obey the same rules as routing nodes routing disabled. These rules are described as follows [29].

Rule 1: A peer and activated node cannot route packets from the North (North) to the East (East) and vice versa.

Rule 2: An odd and enabled node cannot route packets from the southern direction (South) to the West direction (West) and vice versa.

Rule 3: All nodes enabled by default, cannot route packets from each of the north and east directions to the south and west directions.

Rule 4: All nodes are not activated by default route packets from respectively the South and West directions to the north and east directions.

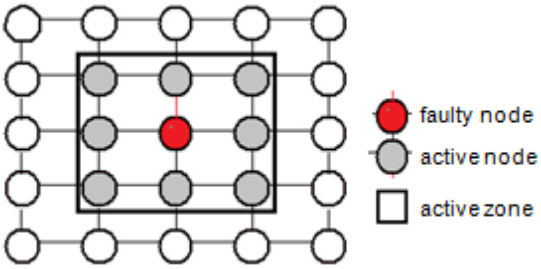
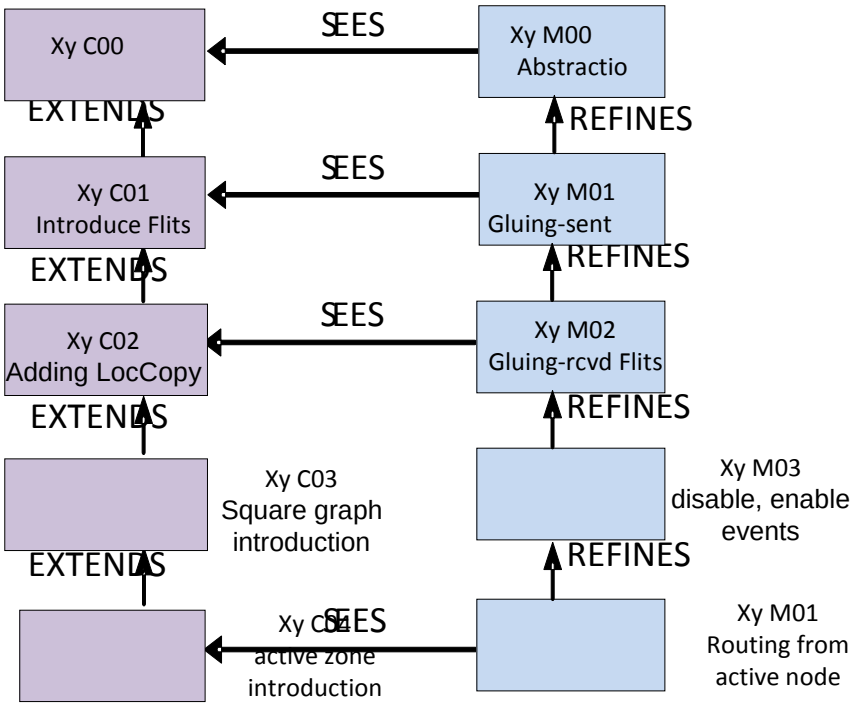


Fig.3. *Exemple d'une zone activée.*

Since a message is broken into packets, which are also broken down into conflicts, then we can rewrite the rules as follows:

- A flit of a packet that is in the active region may circulate in the x-axis and not in the Y axis;
- A flit of a packet that is in focus can move from north to west but not east;
- It can be routed arriving from South to East not west.



V. MODEL DESCRIPTION

The formal development of the fault tolerant routing scheme of the considered NoC Architecture. This proving formal is based on refinement which allows breaking the operation complexity of the routing algorithm and performing this formalization with different levels of abstraction carried out step-by step **[11]**. Fig.5 presents the step-step modeling of the proposed fault tolerant routing scheme.

Fig. 5. Step-by-step Modeling of fault tolerant routing algorithm suitable for NoC.

The abstract level defines the role of the network to send an infinite number of messages which are packetized and encapsulate (*Flitization*) into sequence of packets from a source (S) to a destination (D).

The machine ***xyM01*** refines *xyM00* and introduce cutting packets on flits (**xyC01:** FLITS is new set introduced by this context, cutting each packet on flits (axm1), and the flits of each packet are different from those of other packets (axm2)

The second refinement add LocCopy variable, this variable (in the context; the local copy of the package is in the original sources, and in the sources of these packages, and Theorem which states that the local copy are originally in one place on the network).

xy M03 a refinement of the behavior of a node in case it is broken and when it returns to normal is expressed in both disable and enable events. This level also allows us to create the variable locCopy to ensure flits sends a packet without losing

disable node is a node *n* becomes faulty / off: it can no longer receive or route messages of its neighbors, the new graph *new-gr* will be the current without bidirectional links between the node n and its neighbors, however Enable is a node n becomes active: it can again receive and route messages from its neighbors when n no longer part of the current graph, gives n in the current graph with bidirectional links with his former neighbors.

Xy C04: introduction of operators for calculating the active zone surrounding a knot near faulty zone. The rectangle given by z (a) and including a contains n nodes whose coordinates (x, y) are defined as: LimXmin(a) $\leq x \leq$ *LimXmax(a) et LimYmin(a)* $\leq y \leq$ *LimYmax(a)*

Xy M04: This machine contains a refinement of two events

- The routing flits in different directions depending on the destination:

If after the node (s) is transmitted flit (f) to the node (y), (x) still has flits of f, the local copy does not change and x no longer has flits of f, the local copy of p changes from x to y. This is expressed in the following warning:

VI. CONCLUSION

B-event method is a formal method for the development of computer systems, the accuracy must be formally established.

The proofs of QNoC architecture did not need tough efforts (neither importing hypotheses or simplifying goals, etc), the mere usage/ running of provers (provided by RODIN platform) allowed us to discharge these obligations. Contrary to the verification by simulation only, our 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.

Our experience shows that many models still contain proven breaches of etiquette, which are detected with a facilitator or a model checker. In addition, although the proof obligations for the absence of deadlocks are provided in Event-B, they are not yet implemented in RODIN. The reason is that the proof obligation in the form of a large disjunction (the disjunction of the guards of all events), which is often very difficult to prove.

It has been found with the Plug-in PROB that this strategy may be applied to the network if they are part of the active area. In the future other strategies will be adopted to this critical situation, we need to formally prove their

A new adaptive routing algorithm based on the rules of circumvention, and improved by a routing strategy. Developed and integrated into a network on chip (RKT-Switch). Implemented on FPGA

Rodin is a platform to edit, animate, and prove-prove against models in an integrated way.

By detecting problems in a model that are not covered by the proof obligations (such as deadlock or other unexpected behaviour). It has been found with the Plug-in PROB that this strategy may be applied to the network if they are part of the active zone. In the future other strategies will be adopted in this critical situation, A new adaptive routing algorithm based on the rules by pass is improved by a routing strategy, developed and integrated into a network on chip and implemented on FPGA.

I. References

[01] D. Cansell, C. Tanougast, and Y. Beviller. integration of the proof process in the design of microelectronic architecture for bitrate measurement instrumentation of transport stream program mpeg-2 dvb-t. 2004.

[02] G. T. Leavens, J.-R. Abrial, D. S. Batory, M. J. Butler, A. Coglio, K. Fisler, E. C. R. Hehner, C. B. Jones, D. Miller, S. L. P. Jones, M. Sitaraman, D. R. Smith, and A. Stump. Roadmap for enhanced languages and methods to aid verification. In S. Jarzabek, D. C. Schmidt, and T. L. Veldhuizen, editors, GPCE, pages 221–236. ACM, 2006.

[03] Event-B.org (2011) Rodin platform version 2.2.2. Released 6 Jan 2011. http://www.event-b.org/

[04] J.-R. Abrial, D. Cansell, and D. Méry. A mechanically proved and incremental development of ieee 1394 tree identify protocol. Formal Asp. Comput., 14(3):215–227, 2003.

[05] E. M. Clarke, O. Grumberg, and S. Jha. Verifying parameterized networks. ACM Transactions on Programming Languages and Systems (TOPLAS), 19(5):726 – 750, September 1997.

[06] R. Bharadwaj, A. Felty, and F. Stomp. Formalizing Inductive Proofs of Network Algorithms. In Proceedings of 1995 Asian Computing Science Conference, 1995.

[07] P. Curzon. Experiences formally verifying a network component. In Proceedings of IEEE Conference on Computer Assurance. IEEE Press, 1994.

[08] M. Gordon and T. Melham. Introduction to HOL: A Theorem Proving Environment for Higher Order Logic. Cambridge University Press, 1993.

[09] J.S. Chenard, S. Bourduas, N. Azuelos, M. Boul´e, and Z. Zilic. Hardware Assertion Checkers in On-line Detection of Network-on-Chip Faults. In Proceedings of the Workshop on Diagnostic Services in Networks-onChips,2007.

[10] IEEE Standard for Property Specification Language (PSL) IEEE STD 1850. 2005.

[11] M. Boul´e and Z. Zilic. Automata-based assertion-checker synthesis of PSL properties. ACM Trans. Des. Autom. Electron. Syst., 13(1):1–21, 2008.

[12] E. M. C. Jr., O. Grumberg, and D. A. Peled. Model Checking. The MIT Press, 1999. Cited on page 6.

[13] U. Ogras, J. Hu, and R. Marculescu. Key Research

Problems in NoC Design: A Holistic Perspective. In Proceedings of the international conference on HArdware/Software Codesign and System synthesis, CODES+ISSS'2005, pages 69–74, September 2005.

[14] K. Goossens. Formal Methods for Networks on Chips. In Proceedings of the Fifth International Conference on Application of Concurrency to System Design, ACSD'05, pages 188–189. IEEE Computer Society, 2005.

[15] S. Jovanovic, C. Tanougast, and S. Weber. A new high-performance scalable dynamic interconnection for fpga-based reconfigurable systems. booktitle = {ASAP}, 2008, pages {61--66},

[16] P. Pande, G. De Micheli, C. Grecu, A. Ivanov, and R. Saleh. Design, Synthesis, and Test of Networks on Chips. IEEE Design & Test of Computers, 22(5):404–413, 2005.

[17] Bjerregaard and S. Mahadevan. A Survey of Research and Practices of Network-on-Chip. ACM Computer Surveys, 38(1), 2006.

[18] Axel Jantsch. Models of Computation for Networks on Chip. In Proceedings of the Sixth International Conference on Application of Concurrency to System Design, ACSD '06, pages 165–178, Washington, DC, USA, 2006. IEEE Computer Society.

[19] S. F. Nielsen and J. Sparsø. Analysis of low-power SoC interconnection networks. In IEEE 19th Norchip Conference, pages 77–86, nov 2001.

[20] C. Grecu, A. Ivanov, R. Saleh, E.r Sogomonyan, and P. Pande. On-line Fault Detection and Location for NoC Interconnects. In Proceedings of the International On-Line Testing Symposium, IOLTS'06, July 2006.

[21] S. Murali, G. De Micheli, L. Benini, T. Theocharides, N. Vijaykrishnan, and M.J. Irwin. Analysis of Error Recovery Schemes for Networks on Chips. Design & Test of Computers, 22(5), 2005.

[22] M. Schafer, T. Hollstein, H. Zimmer, and M. Glesner. Deadlock-free routing and Component placement for irregular mesh-based networks-on-chip In Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design, ICCAD '05, pages 238–245, Washington, DC, USA, 2005. IEEE Computer Society.

[23] B. Gebremichael, F.W. Vaandrager, M. Zhang, K. Goossens, E. Rijpkema, and A. Radulescu. Deadlock Prevention in the Æthereal Protocol. In Proceedings of CHARME'05, October 2005.

[24] Samia Jens Bendisposto-Michael Leuschel-Olivier Ligot-Mireille. La validation de modèles Event-B avec le plug-in ProB pour RODIN. 2008. volume = {27}, pages = {1065-1084}.

[25] M.B. Andriamiarina, H. Daoud, M. Belarbi, D. Mery, and C. Tanougast. Formal verification of fault tolerant NoC-base architecture. MOMA journal special issue selected papers of IWMCS 2012, volume 01-Issue 02-ISSN n°2253-0665(2012), p1

[26] SAYAR Imen. D'Event-B vers UML/OCL en passant par UML/EM-OCL. 2012.

[27] Jastram Michael. Rodin User's Handbook

[28] C. Killian, C. Tanougast, F. Monteiro, and A. Dandache, "A New Efficient and Reliable Dynamically Reconfigurable Network-on-Chip", Journal of Electrical and Computer Engineering, special issue Design and Automation for Integrated Circuits and Systems, Volume 2012, Article ID 843239, 16 pages, Hiwdawi, 2012.

[29] M.B. Andriamiarina, H. Daoud, M. Belarbi, D. Mery, and C. Tanougast. Formal verification of fault tolerant NoC-base architecture. MOMA journal special issue selected papers of IWMCS 2012, volume 01-Issue 02-ISSN n°2253-0665(2012), p1