International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 5, No.2 (March-2016)

http://dx.doi.org/10.12785/ijcds/050207

Embedded Systems Design Using Event-B Theories

Abdelhamid HARICHE [1], Mostefa BELARBI [1] and Abdallah CHOUARFIA [2]

[1] *LIM Research Laboratory, University Ibn khaldoun of Tiaret, Tiaret, Algeria*
[2] *University of Science and Technology of Oran, Algeria*

Received 15 October 2015, Revised 15 December 2015, Accepted 1 January 2016, Published 1 March 2016

Abstract: FPGA-Based Embedded systems have a strong impact in everyday life, these systems must be reliable, and their specifications and design should be clear, understandable and must follow specific rules. Therefore, because of these requirements, the method of formal development Event-B based on theories delivered by the Rodin tool-set gives not only a good emphasizing of both static and dynamic aspects for this kind of systems but also a powerful insurance and the guarantee of their accuracy with respect to issues of safety and security by the creation of theories that cover all properties of a system. In this paper we illustrate this concept by modelling NoCs, colored graphs and VHDL language. Thus all these last theories are deployed, discharged and used in Event-B models to represent and enhance the performance of this self-organization reliability solution for the wireless sensors network of NoC-based system.

Keywords: Formal methods, Event-B,theories ,VHDL, Embedded systems, Network on chip, self recovering

1. INTRODUCTION

The need for high performance, available and reliable embedded systems has made computing systems increasingly complex. Faced with such complexity, to design an FPGA-based embedded system project with only one language has still insufficient. Verification is a used to demonstrate whether the functionality of the project under development is in accordance with their specification or not [1], the most popular types of verification are the formal, functional and hybrid verification. According to [1, 2], around 70% of the resources employed in a hardware design are used in the functional verification step [1].

The ultimate functional techniques designed to validate the implementation of embedded solutions such as multiprocessor systems on a chip (MPSoC), which have the advantage not only dependability but also more gain of implementation time for the system. Having this work is to start to improve (in terms of reduced time and reliability), the validation phase of a high-level design of a MPSOC system based on NOC network architecture by providing a satisfiable result for the VHDL (Very High Speed Integrated Circuit Hardware Description Language)code generation.

The exploited Formal methods refer to the mathematical techniques employed for the specification, the development and the verification of software and hardware systems [3] in an incremental fashion where each intermediate step is verified. Formal methods lead to deeper understanding and higher consistency of specification and design than informal or semi-formal methods. In order to manage system complexity, abstraction and refinement are key methods for structuring the formal modelling effort since they support separation of concerns and layered reasoning. Event-B [4] is a formal method based on the B method [5] that has a good industrial strength and covered by the Rodin platform [6] which provides a toolset to carry out of specifications, refinements and proofs in Event-B. It based on set theory that allows precise system descriptions to be written in an abstract way (models). Manipulating various tools of the famous Rodin platform called with the fundamental formalism Event-B where the refinement mechanism pulling as a major criterion of selection also based on the notion of theory (theory Plug-in) to express NoC desired properties that represent the VHDL code behavior of this kind of systems. Rodin proposes for the user a reactive modelling environment to link models, proof obligations and their corresponding proofs. Since proofs are important to the modelling activity, Rodin provides a proof infrastructure that is extensible with external provers (e.g., Atelier-B provers [7, 8]) that can also be used in conjunction with the Rodin internal prover. Every model contains theories in form of different types (integers, sets, relations, functions …), so when systems require additional mathematical structures (e.g: lists, trees,

E-mail: haricheabdelhamid@gmail.com, mbelarbi@univ-tiaret.dz, chouarfia@univ-usto.dz

graphs, reals) which might defined axiomatically in models, where no polymorphism is supported, no direct provers extension is possible and also the definitions and proof rules are not ensured, this case leads to a new extension of modelling using theory plug-in (in form of polymorphic) which allows users to define new mathematical operators and data types, add proof rules to Rodin prover (interactive rules added to automated tactics) and generate soundness POs for new definitions and proof rules.

This work is a presentation of the Event-B and the Rodin tool and their presence for the validation of systems, later network on chip technology and its use in a wireless network is explored. And the presented formal verification with Event-B of this NoC-based architecture using the properties of the different theories deployed. Our goal in this article is to show the power of the theories in a form of polymorphic [9,10, 11, 12] for the formal verification of NoC-based Wireless sensors network (WSNoC) system by proposing set of basic theories according to the requirements of this kind of systems. Following the cycle of formal validation for this architecture (a particular type of NoC), first a theory that covers most of the commune properties of NoC after its formal validation was performed[13, 14],then it started to prepare an extended theory for the well running of the self-recovering and especially the exchange of data during a software failure inside the nodes and this lead us to generalize a theory validated strategy for adaptation of Wireless Sensors Network of NoC in the step that follows it still have to check every possible scenarios for the self-adaptation of this system, however simulations for these sketches will not give us an exhaustive audit of all properties even formal verification that check this strategy where every node is separately manage its situation which means that these verifications without checking the network properties still incomplete, so when the network could be represented as a graph then it is obvious to extend the theory of graphs produced by J.R Abrial[15,16] into a colored graph theory to animate and increase the complexity verifying the reconfiguration strategy for the failure nodes inside a network. In the final phase, and after verifying all the properties for this architecture, it can be represented with carrying about its VHDL behaviour using VHDL theory, so this work that formal verification based on theory gives us a rich verification that affects all the feasible scenarios for our system following different axis and in a methodological way.

This paper is organized as follows: Section.3 Explains the Event-B theory and its use during modelling on Rodin, the section 4 presents the formalization of NoC systems into a theory using Event-B, when in section 5 presents the wireless sensors network of NoC as a created theory, then in the section 6 shows the colored graph theory and the theory of closure after that the section 7. Presents the VHDL theory and it shows the benefit of creating it by giving the similarities between some Event-B syntax and the real VHDL code and After All that we enrich our paper by the deployment of a case study that combined all this theories to check all the Wireless sensors network of NoC as showing in the section 8 to finally get to the Section 9 that concludes this paper along with the future work.

2. RELATED WORKS

This section introduces some current works that had a relation with the verification of embedded systems in general, and then specify a brief study for the projects issued with our particular wireless network systems.

In the aim of catching up with the embedded systems components complexity, the ESL (Embedded System Level) tools [17] have emerged for the reason of improving the level of abstraction of hardware developments (simulate a design at Transaction Level Modeling to synthesize hardware architecture directly from a Register Transfer Level description). This approach have been first proposed in the model [18], generalized by the Model Driven Development approach and standardized by the Model Driven Architecture OMG's standard. Moreover, in order to allow faster design, system under study must be modeled and validated at several levels of abstraction [19]. There are no real conditions about the required abstraction levels even if there are some efforts to define and standardize it [20].

More of challenges are appeared according to POOSL analysis based on simulation where the results are estimation results. To this reason, a transformation from a POOSL model of a motion control system into an UPPAAL model to enable formal verification [21].The next step is covered by showing that worst-case latencies and functional behavior can indeed be verified with the obtained UPPAAL model. Also by proposing a method for recording time (clock value) such that estimated worst-case (best-case) latencies can be obtained with UPPAAL simulator and then be used as a starting point for determining the exact worst-case (best-case) latencies by UPPAAL verification. This work experiments show that the performance results verified with UPPAAL match the estimation results obtained from the POOSL model.

While expressivity is needed for mastering heterogeneity and complexity about system correctness and performance at design time of embedded systems. BIP is currently equipped with a series of highly

expressive runtime verification and simulation engines. While those facilities cannot be used to assess the overall correctness of any entire system, SBIP the stochastic extension of the BIP formalism and toolset allows specifying systems with both non-deterministic and stochastic aspects. However, the semantics of such systems will be purely stochastic behavior it must be added to atomic components in BIP by randomizing individual transitions. Indeed, it suffices to randomize the assignments of variables, which can be practically done in the C functions used on transition. Hence, from the user point of view, dealing with SBIP is as easy as dealing with BIP.

A team work[22] concerns a clock synchronization protocol running within a distributed heterogeneous communication system (HCS) To verify such property, a stochastic model was built in SBIP implementation toolset, this model is composed by two deterministic components namely Master, Slave and two communication channels.

In our particular system Self-recovering or self-reconfiguration is one important factor during the communication inside a network of sensors. Many works focused on this strategy by attribute some consideration of the functional verification of the routing algorithms using ESL tools [23, 24]. Besides those works, UML(Unified Modeling Language) and especially MARTE(Modeling and Analysis of Real-Time Embedded systems) is used to analyze the three-axis classification scheme that used to classify a reconfigurable approach by the community[21]: mainly where (reconfiguration can either be exo-reconfigurable which is initiated and controlled by an external source for example the FPGA co-processor on a PCI bus, or endo-reconfiguration in this case the FPGA itself loads the configuration and reconfigures itself such as a hard/soft processor manages the reconfiguration) when (The reconfiguration can be either dynamic which is carried out on the fly when the FPGA is active and running and can be directed by the application and offers flexibility advantages for systems or static which requires the FPGA to be inactive) and how (reconfiguration can be either full which completely reconfigures the whole area of an FPGA where a full device bitstream is transmitted over a communication channel, or partial reconfiguration concerns only a region or regions of FPGA while the remaining portions continue their normal execution)the reconfiguration takes place.

Always taking a target of formal verification for the recovering networks starting by projects propose new rules using formal methods such as using Switches algebra [25]. All these previous works miss the details during the coverage of embedded systems properties and constraints, when Event-B uses refinement more as a technique to structure complex proofs, focusing less on preserving correctness along a sequence of models. Therefore it must mentioned that some current works used Rodin tool-set to formalize the different actions inside this self-recovering network[26] when others try to use Event-B models that express their way of interpreting some properties for common theories (graph theory, probabilistic theories..) to validate the distributed systems [27, 28, 29].

These precedent works always unlike our formal approach that do care about the architecture and the services delivered by an embedded system with the way of running for this application even the constraint of programming language and depend on the power of proposed theories and the proven polymorphic to guarantee the right preservation of all proofs for any embedded system and especially the wireless sensors networks of NoC, this kind of distributed systems lead us to use the closure theory to check all the graph properties then to ensure the multi-reconfiguration of the faulty nodes we introduced colored graph theory rules combined with the NoC theory and the WNoC theory to cover the strategy of reconfiguration inside the wireless network of a set of NoC-based switches , the final step is to carry out about the runtime application by adding the VHDL theory to finalize the cycle for the validated implementation of this embedded system using Event-B theories.

3. THEORY-BASED MODELLING IN EVENT-B

Models in Event-B are specified by means of contexts (static properties of a model) and machines (dynamic properties of a model) and during the modelling of every system, the main benefit in this work is the use of models from theories already validated to reduce the time in the formal specification of any given system . taking as target the NoC system after n-level of verified models, a theory (a new kind of Event-B component) is defined independently of any particular models. A theory component has a name, a list of global type parameters (global to the theory), and an arbitrary number of definitions and rules:

Theory *name*
type parameters *T1, . . . , Tn*
{ _Predicate Operator Definition _
/ _ Expression Operator Definition _
/ _ Data Type Definition _
/ _ Rewrite Rule _
/ _ Inference Rule _ }

The Theory plug-in provides a mechanism to extend the Event-B mathematical language as well as the prover. The main purpose of this new feature was to validate systems with a way to extend the standard Event-B mathematical language by supporting operators, basic predicates and algebraic types. Along with these additional notations, it can also define new proof rules (proof extensions).

A theory is used to hold mathematical extensions: data types, direct, recursive and axiomatic definitions of operators, and proof extensions polymorphic theorems, rewrite and inference rules. Theories are a helpful basis for the static checking and proof obligation generation which ensure that no compromise for the existing infrastructure of modelling, proof and any contributed extensions. In essence, the theory plug-in provides a systematic platform for defining, validating of the well-creation of embedded systems.

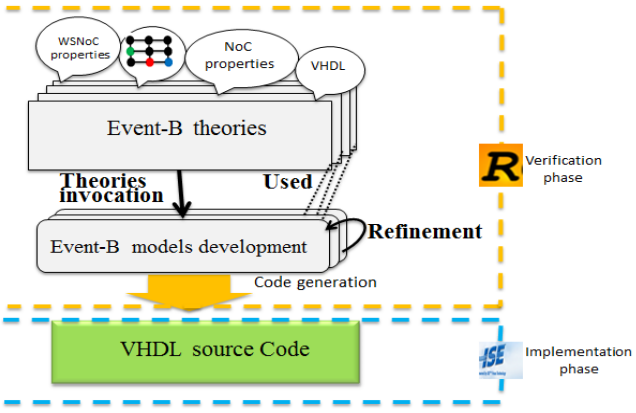


Figure 1. The theory-based modelling of WSNoC system in Event-B

This work proposes to validate the wireless sensors network using the graph theory to handle this kind of distributed system (Figure1). Since this network is composed of set of NoC-based switches, we introduce the NoC theory but the failure may appear inside the wireless network so it is clear to propose a new strategy for recovering the set of faulty nodes in form of theory and to handle and manage the complexity of the multi-failure of sensors using the colored graph theory, the final step is to ensure that all the properties for this embedded system will applied correctly in the application environment by combining these previous theories with the VHDL.

4. THE NOC THEORY DEFINITION IN EVENT-B

This section presents the formal development of the NoC Architecture. (Figure 2) Knowing that:

- The NoC architecture is usually a Mesh topology: boundary switches are connected to two or three neighbors, or even to four neighbors.

- The role of a switch is to pass data packets between elements (routers) of a NoC architecture, each incoming packet is stored in an input register, the packets direction's (whether N, E, S or W) policy is based on the rules of right priority policy.

- The Output Logic is made up of a semi crossbar, an out- put buffer and a finite state machine; those components are for routing the incoming packets according to the state of neighbors, storing packets if the output direction is occupy, even informing neighbors about the state of switch.

- The XY routing algorithm defines by 2D coordinates: (xs ,ys) for the source (s) and (xd ,yd) for the destination (d) of a packet P:

- The packet (p) travels first along x dimension, until xs =xd. Then, the packet (p) travels along y dimension, until ys =yd.

- If the packet (p) encounters elements unable to transmit data in x dimension, the routing temporary switches to y dimension.

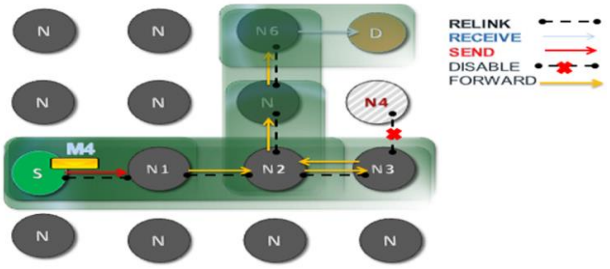


Figure 2. The NoC theory in Event-B

The NoC net is a set of nodes that could have so many roles like being a source *src* of packet transition or it destination *rcv* or an intermediate *dst* during the phase transition even nodes also could have no role *nop*.

Datatype *role* **constructors** *nop, src, rcv, dst*	**Operator** node **prefix** **args** *r:* role, nod:ℙ(S) **condition** *nd∈nod* **definition** nd

The NoC is introduced as a network (a graph) between the sources and destinations of packets. This graph is non-empty, non-transitive and symmetrical.

thm : ∀net,T·net∈ℙ(S×S) ∧ T∈ℙ(S) *any∈role*
∧ *nodes∈node(any,T)*
⇒
sym(net,nodes) ∧cls(net)
∧irreflexive(net,nodes) ∧ card(net)≠0

This graph could be represented like a grid with a very specific size *netsize*.

operator *in_tmax* **definition** 1000// for example the net at maximum had 1000 nodes	**operator** *netsize* **definition** 1..int_max

Every operation inside the NoC net is related to the input port and output port (represented by *iop*) of the nodes represented by *portsNoc*and the buffers (represented by *buffersNoc*for the buffers and *buffchnlNoct*for the buffers that used credit to send data) to store the data transmitted inside the net.

When it is about the set of process that every NoC network used it needs to explain firstly the set of following operators:

Sent: to represent the source of transmitted data.

Availableplace: to represent the maximum of free buffers inside the node.

Every node in the network net could do on of the following events:

A. *Data sending*

When a source sends a packet (m), the packet is put in the network in an input port (ip) of the switch (s).

$$snt = snt \cup \{s \mapsto m\} \land destin = destin \cup (d \times \{m\})$$
$$\land bufferin = bufferin \cup \{ip \mapsto m\}$$

B. *Credit-data receiving*

A credit packet (crd) is received from an output port of switch (op), if there is an available place (places) in the buffer.

$$buffcrdchn = buffcrdchn \setminus \{ch \mapsto bfc\}$$
$$\land sizeplaces(out,op,places) = sizeplaces(out,op,places) + 1$$

datatype *iop* **constructors** *in ,out*	**operator**portsNoc **prefix** **args***pin:*iop, r:ℙ(S×S) **definition**r	**operator**buffersNoc **prefix** **args***pin:*iop r:ℙ(S×S) m: ℙ(T) **condition** *anypin∈iop*∧ *ports∈* portsNoc(anypin,*r*) **definition**ports↦*m*	**operator**buffchnlNoc **prefix** **args***pin:*iop r:ℙ(S×S) m: ℙ(T) crd: ℙ(R) **condition** *anypin∈iop* ∧ *buff∈* buffersNoc(anypin,*r*,*m*) **definition**buff↦*crd*	**operator**coord **prefix** **args**xy: ℕ↔ ℕ **condition** *xy*∈netsize→netsize **definition**xy
operator*sent* **prefix** **args**a:ℙ(S) b:ℙ(S) *m:*ℙ(T) **condition***any∈role* ∧nd ∈ℙ(S) ∧ a ∈node(*src,nd*) ∧ b ∈node(*any,nd*) **definition***a ↦m*	**operator***receive* **prefix** **args**a:ℙ(S) b:ℙ(S) *m:*ℙ(T) **condition***any∈role* ∧ nd ∈ℙ(S) ∧a ∈node(*any,nd*) ∧ b ∈node(rcv,*nd*) **definition***b ↦m*	**operator***chanel* **prefix** **args**a:ℙ(S) b:ℙ(S) *m:*ℙ(T) **condition***any∈role* ∧ nd ∈ℙ(S) ∧a ∈node(*any,nd*) ∧ b ∈node(*any,nd*) **definition***a↦b*	**operator**Position **prefix** **args**nod:ℙ(S) **condition***any∈role* ∧ *net∈node(any, nod)* ∧ c*ls(nod× nod)* ∧*pos*∈ℕ↔ ℕ ∧xy∈coord(pos) **definition** {n,f·n∈net∧f∈n→xy|f}	**operator***availableplce* **args**r:ℙ(S×S) **condition** *any∈* ℙ(S×S) *∧any=*portsNoc(*out,r*) **definition**1..*max_place*
operator*forward* **prefix** **args**a:ℙ(S) b:ℙ(S) *m:*ℙ(T) **ondition***any∈role* ∧ nd ∈ℙ(S) ∧a ∈node(*dst,nd*) ∧ b ∈node(*any,nd*) **definition***b ↦m*	**operator***store* **prefix** **args**a:ℙ(S) *m:*ℙ(T) **condition***any∈role* ∧ nd ∈ℙ(S) ∧a ∈node(*any,nd*) **definition***a↦m*	**operator***max_places* **definition**3		**operator***sizeplaces* **prefix** **args**r:ℙ(S×S) , *pin:*iop *nbrfree :*ℕ1 **condition***port∈* ℙ(S×S)*∧port=*portsNoc (*out,r*) **definition***nbfree*

Receive: to represent the destination of transmitted data.

Forward: to represent the node that will transfer data for its destination.

Chanel: to represent the couple of nodes during the transmission of data.

Store: to represent the node that store transmitted data and the stored data.

Sizeplace: to represent the number of free places for the out port to transmit data.

Coord: to represent integer coordinate.

Position: to represent the coordinates of two nodes in a closure.

C. *Data recieving*

A packet is received by its destination, if the packet has reached the destination.

$$rcvd = rcvd \cup \{d \mapsto m\}$$
$$\land inputbuffer = inputbuffer \setminus \{ip \mapsto m\}$$
$$\land buffcrdchn = buffcrdchn \cup \{portsNoc\sim(in,ip) \mapsto bfc\}$$

D. Data forward

In the network, a packet (m) transits from a node (x) to another node (y), until it reached its destination (d). This passage is from the output logic (op) of switch to a channel (swithchcontrol).

switchcontrol = switchcontrol \ {x ↦ m}
∧ outputbuffer ≔ outputbuffer ∪ {op ↦ m}

E. Switch control

Models the passage of a packet (p), from an input port (ip) of a switch (x), to an output port (op) leading to a switch (y) if the switch (x) is not busy anymore, it sends a release signal to the previous switch linked to the input port.

inputbuffer =inputbuffer \ {ip ↦ m}
∧ outputbuffer =outputbuffer ∪ {op ↦ m}
∧ availableplc(op) = availableplc(op) − 1
∧ *buffcrdchn = buffcrdchn ∪ {portsNoc~(in,ip) ↦ bfc}*

When every node in the NoC net has four directions the Switch can control four cases to route incoming and out coming data these cases are explained as follow:

- Switch control left models Case 1: a packet (m) is transmitted, from an input port of a switch (x), to an output port, leading to a neighbour (y). This event is triggered if the x-coordinate of the destination (d) (of the packet(m)) is inferior to the x-coordinate of the current node (x).
- Switch control right models Case 2: a packet (m) is transmitted, from an input port of a switch (x), to an output port, leading to a neighbour (y). This event is triggered if the x-coordinate of the destination (d) (of the packet(m)) is superior to the x-coordinate of the current node (x).
- Switch control up models Case 3: a packet (m) is transmitted, from an input port of a switch (x), to an output port, leading to a neighbour (y). This event is triggered if the y-coordinate of the destination (d) (of the packet(m)) is superior to the y-coordinate of the current node (x), and either,

if the x-coordinate of the destination (d) is equal to the x-coordinate of the current node (x), or if the packet (m) cannot transit along the x-axis.

- Switch control down models Case 4: a packet (m) is transmitted, from an input port of a switch (x), to an output port, leading to a neighbour (y). This event is triggered if the y-coordinate of the destination (d) (of the packet(m)) is inferior to the y-coordinate of the current node (x), and either, if the x-coordinate of the destination (d) is equal to the x-coordinate of the current node (x), or if the packet (m) cannot transit along the x-axis.

The same predicates as in swtich control
..... But In plus we add :
Case 1 : Switch control Left
prj1(position(x)) > prj1(position (d))
∧prj1(position (y)) = prj1(position (x))−1
∧prj2(position (y)) = prj2(position (x))
Case 2 : Switch control Right
prj1(position(x)) < prj1(position (d))
∧prj1(position (y)) = prj1(position (x))+1
∧prj2(position (y)) = prj2(position (x))
Case 3 : Switch control up
(((prj1(position(x)) = prj1(position(d)))
∨ (prj1(position(x)) > prj1(position(d))
∧ x ↦ position~(prj1(position(x))−1 ↦prj2(position(x))) ∉ gr
)
∨(prj1(position(x)) < prj1(position(d))
∧ x ↦ position~(prj1(position(x))+1
↦ prj2(position(x))) ∉ gr))
∧ prj2(position(x)) < prj2(position(d))
∧ prj1(position(y)) = prj1(position(x))
∧prj2(position(y)) = prj2(position(x))+1
Case 4 : Switch control down
(((prj1(position(x)) = prj1(position(d)))
∨ (prj1(position(x)) > prj1(position(d))
∧ x ↦ position~(prj1(position(x))−1 ↦prj2(position(x))) ∉ gr
)
∨(prj1(position(x)) < prj1(position(d))
∧ x ↦ position~(prj1(position(x))+1
↦ prj2(position(x))) ∉ gr))
∧ prj2(position(x)) < prj2(position(d))
∧ prj1(position(y)) = prj1(position(x))
∧prj2(position(y)) = prj2(position(x))-1
The same actions as in swtich control

F. Data passage from a buffer to a chanel

The transition of a packet (m) from an output port (op), to a channel (ch) leading to a target switch (n).

chan ≔ chan ∪ {ch ↦ m}
∧ *outputbuffer =outputbuffer \ {op ↦ m}*

G. Data passage from a chanel to a node

To transfer of a packet (m) from a channel (ch) to a connected switch (n).This transition of a packet (m) from a channel (ch) to an input port (ip) of a target switch (n).

chan = chan \ {ch ↦ m}
∧ *inputbuffer =inputbuffer ∪{ip ↦ m}*

H. Disabling a link of a node

A disabled node (nd) is not allowed to communicate with its neighbours (failure, etc.) so this lead to remove its link (lnbg) form the closure of the network (gr).

$$gr = cls(gr \setminus (lnbg \cup lnbg^{\sim}))$$

I. Relink a node

The reconfiguration of the network is process where Disabled nodes are re-enabled: the links between them and their neighbours are restored, therefore allowing communications and packets transfers so this is meaning to add the link (lnbg) of the node (nd) to the closure of the network (gr).

$$gr = cls(gr \cup (lnbg \cup lnbg^{\sim})) \\ \wedge \\ sizeplaces(out,op,places) = sizeplaces(out,(lnbg \cup lnbg^{\sim}) \lhd op, availableplc((lnbg \cup lnbg^{\sim}) \lhd op))$$

5. THE WSNOC-BASED NETWORK IN EVENT-B

In the purpose of improving our approach for the implementation of systems using a formal verification, it is restricted in the creation of a very particular NoC system inspired from the model proposed [31] for a self-reconfigurable multi-node network, this net is composed of a set of self-organized wireless nodes. Each node is independent. This allows maintenance and operational reliability of the system in the case of failure. this network system is a system composed of several nodes which apply XY algorithm inside as described previously and communicate with each other through a communication channel, which can be a network, shared variables, messages, this allows us to consider our system like a distributed system which is often seen as a graph [32], where the vertices are the nodes and the edges, direct communication links between them.

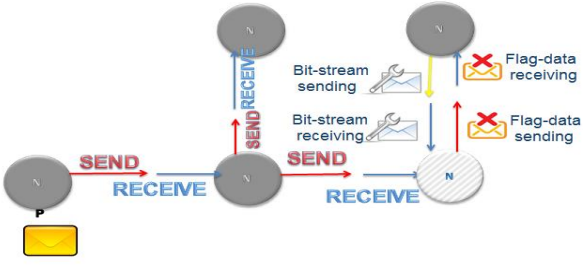


Figure 3. The Wireless network of NoC theory in Event-B

During the modelling for this particular network (Figure3) used NoC, it must introduced the way of reconfiguration for any faulty node and for this new constraint it adds a new extension for the NoC theory , this new theory Wireless-NoC contains:

The new data-type "state" is used to represent the state of every node.

The next operators are used to clarify more the new theory that respects all the properties of reliabilities:

WNocStat: to create a set of nodes with its state:

ocpy: mentions that the node is occupied by sending or receiving data.

free: mentions that node is not occupy.

nodeState: this operator presents the state of specific node "nod".

packet: to represent the new structure of data which contains a flag "flg" to specify the type of data in the case of node failure this flg will have the value 1.

Datatype *state* **constructors** *ocpy* *free*	**operator**WSnode **prefix** **args***s:* state , nod:ℙ(S) **condition***any∈role* ∧ *net∈node(any, nod)* **definition** net
operator packet **prefix** **args***flg:* ℕ data:ℙ(T) **condition***flg∈0..1* **definition** data	
operatornodeState **prefix** **args** nod:ℙ(S) **condition***a∈role* ∧ *net∈node(a, nod)* ∧ *s∈state* **definition** s	

So it can present this particular NoC using Wirless-NoC theory definition like in follow:

A. Data sending

When a source *Srcnod* sends a packet (msg), the packet must have a value of 0 for the *flg* argument and the states of nodes *Srcnod*, *Desnod* must be changed with the value *ocpy*.

B. Data recieving

A packet is received by its destination, if the packet has reached the destination (node with a role equal to *rcv* **Case 3** in predicate).This packet must have a value of 0 for the *flg* argument and the states of nodes *Srcnod*, *Desnod* must be changed with the value free.

C. Data forward

In the network, a packet (*msg*) transits from a node (*Srcnod*) to another node *Desnod*(node with a role equal to *dst* **Case 2** in predicate).This packet must have a value of 0 for the *flg* argument and the state of the node *Srcnod*

must be changed with the value free when the node *Desnod* will have the value *ocpy*.

Case 1: SEND
Send=send∪{Srcnod ↦msg} ∧ srcstate= ocpy
∧ Desstate= ocpy
Case 2: FORWARD
rcvd= rcvd ∪{Desnod↦msg} ∧ srcstate= free
∧ Desstate= ocpy
Case 3: RECIEVE
rcvd= rcvd ∪{Desnod↦msg} ∧ srcstate= free
∧ Desstate= free

In the optimal case those last events could represent the communication between different nodes otherwise some nodes could be in failure state so they must inform the other nodes by sending a *flagData* with *flg*=1, (see Flag data sending).

D. Flag-data sending

In case of a node (Fnod) Failure which can't receive a packet (msg), the packet FlagData with a value of 1 for the flg argument is sent if the faulty node Fnode still be occupied (value ocpy).

When the flag data is received (see Flag-data receiving), one of the healthy nodes take the mission of reconfiguration (see Config-data receiving) for the faulty nodes after checking these following rules:
Every node is not a faulty node, has the material resources to implement IP node, has the Bitstream packet of configuration IP and do not be occupied by a priority.

∀*send, Fnod, FlagData*•
FlagData∈packet(1,ℙ(T))
∧ srcstate=Statenod (Srcnod)
∧ failstate= Statenode(Fnod))
⇒
Send=send∪{Fnod ↦FlagData}
∧ srcstate= ocpy ∧ Failstate= ocpy

E. Flag-data recieving

A packet (*FlagData*) is received by switch (*Srcnod*), if it is sending from a faulty node (*Fnod*).

rcvd = rcvd ∪{Srcnod ↦FlagData}
∧ srcstate=ocpy ∧ Failstate=ocpy

F. Config-data sending

In the case of a node (Src*nod*) can check the rules of reconfiguration ability (described before) it send a configuration data (*Bitstream)* to the faulty node (*Fnod)*.

∀ *Bitetream,FlagData*•
Bitsream∈packet(0,ℙ(T)) ∧ FlagData∈ packet(1,ℙ(T))
⇒
Send=send∪{Srcnod↦Bitstream}
∧ srcstate=free ∧ Failstate=ocpy

G. Config-data recieving

A packet (*FlagData*) is received by switch (*Srcnod*), if it is sending from a faulty node (*Fnod*).

rcvd = rcvd \ {Fnod↦Bitstream }
∧ srcstate=free∧ srcstate=free

6. THE COLORED GRAPHS THEORY IN EVENT-B

In the reason of managing nodes failures and the way of the self-recovering, the graph theory help to specify the Wireless reconfigurable WSNoC (Figure4) using algorithms of colored graph principals [33] that include a part from Closure theory [16, 17], the *closure* theory as in follow composed of operator *cls* and have so many properties of graph such as composition of sub-closures an transitivity:

THEORY closure
OPERATORS •cls : cls(r : ℙ(S×S))
direct definition
cls(r : ℙ(S×S)) ≙ fix(λs·s∈ℙ(S×S) | r ∪ (s;r))
THEOREMS
∀r·r∈ℙ(S×S) ⇒ cls(r) = r ∪ (cls(r);r)
∀r·r∈ℙ(S×S) ⇒ r ⊆ cls(r)
∀r·r∈ℙ(S×S) ⇒ cls(r);r ⊆ cls(r)
∀r,x·r∈ℙ(S×S) ∧ r[x]⊆x ⇒ cls(r)[x]⊆x ,
∀r,s·r∈ℙ(S×S) ∧ s∈ℙ(S×S) ∧ r⊆s ∧ s;r⊆s ⇒ cls(r)⊆s
∀r·r∈ℙ(S×S) ⇒ cls(r);cls(r)⊆cls(r)

- The colouring graph algorithms [32, 33] can be used to control a set of nodes: There may be two nodes that have the same job but two adjacent nodes cannot even fix the failed node at the same time [31].

- Math extension is a standard library provides the closure [16, 17] as a theory which is almost similar to our Wireless NoC architecture but need to add the colouring rules in the context to cover all courant proprieties of our Graph Theory.

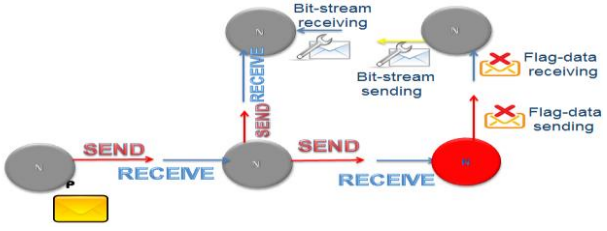


Figure 4. The colored graph theory application onto WSNoC system in Event-B

During the refinement events which are for the Application variables represented by a graph theory and their Execution environment variables must be handled in the same model by introducing the VHDL theory that could be used in every event Model as new VHDL variables represented with Event-B. So the coloration will cover all possible state for the events that nodes can do.

Datatype *Colors* **constructors** *Yellow,Red,* *Green,Bleu ,none*	**Operator** Colorenode **prefix** **args** *c:*Colors , nod:ℙ(S) **definition** nod

So any node that cannot send and receive data can be labelled with *Blue* ,*Yellow*, *Green* , *Red* and uncoloured even managed the multi-failure of nodes that overpass more than 4 nodes by counting it as node in waiting list to be in the next colored due to the available colour.

∀ *clr,count*• clr∈ COLORS ∧ Count∈ℕ
Case 1 :colored_faulty
∧*count* ≤ 4
Case 2 :faulty_wait_to_be_colored
∧*count* > 4
Case 1 :colored_faulty
colored= colored∪{Fnod}
∧ colored(Desnod)=clr∧ count=count+1
∧ has_colored=has_colored∪{Fnod}
Case 2 :faulty_wait_to_be_colored
wait_to_be_colored= wait_to_be_colored∪{Fnod}
∧ count=count+1
∧ colored(Desnod)=clr

A. *Flag-data sending*

In case of a node (Fnod) Failure which can't receive a packet it must colore the node that sends a FlagData with red.

Send=send∪{Fnod ↦FlagData} ∧ colored(Fnod)=clr
∧ has_colored:=has_colored∪{Fnod }

B. *Config-data sending*

Bitstream-packet (Bitstream) is received by a faulty switch (Fnod), if it is sending from a reorganizer IP (Rfcgnod) in the buffer, then for that reason both nodes Fnod and Rcfgnod must be uncolored.

rcvd = rcvd \ {Fnod↦Bitstream }∧colored(Fnod)=none
∧has_colored:=has_colored\{Fnod }

7. THE VHDL THEORY IN EVENT-B

The Theory *VHDL_th* used during this Self-Organized network Modelling is composed of an Entity and a set of architectures that contains a set of variables, the entity contains ports in direction in out or in_out, from that two operators are created (*arch_decl* and *ports_decl*) when one had a data type parameter (*pio*), in this VHDL case study, we had *std_logic* and the *std_logic_vectors* signals, the last one must have new theory could be used in the *VHDL_th*, it is a similar structure of array presented in[34].

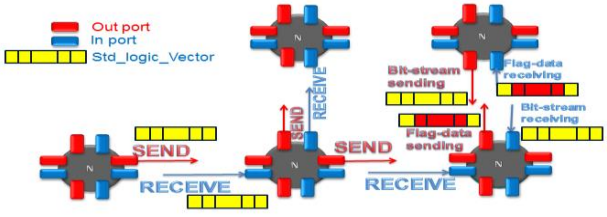


Figure 5. The VHDL theory application onto WSNoC system in Event-B

The variable in VHDL must have a type and in this structure for example the integer type is represented like an operator and ensure that couldn't be over *int_max*=2^{32} and every operation that use that kind of variable must never overflow that value so the Proof obligation will ensure the non-overflow error for integer variables in the VHDL code.

THEORY VHDL_th
DATATYPES
pio≙I n , out, inout
OPERATORS
•arch_decl :
arch_decl(s:ℙ(T))
direct definition
arch_decl(s : ℙ(T)) ≙ s
•port_decl : port_decl(p:pio,s: ℙ(T))
direct definition port_decl(p:pio,s : ℙ(T)) ≙ s

•std_logic : std_logic
direct definition
std_logicl ≙ 0··1
•vector:
vector(s: ℙ(T))
direct definition
vector(s : ℙ(T)) ≙
{n,f·n∈ℕ∧f∈0··(n−1)→s|f}

•std_logic_vector:
vector(length: ℕ,s: ℙ(T))
well-definedness condition
direct definition
std_logic_vector(length: ℕ,s : ℙ(T)) ≙
{v|v∈vector(s)∧card(s)=length}

During the modelling of our VHDL theory that it will introduce into the Event-B model the set of operations created in our new theories and taking some cases like VHDL operations (+,*,-, /), or the assignment of variables in VHDL, also some proofs had to be discharged during this WSNoC modelling will be a good start for the code generation after ensuring the well definition of all the system.

8. THEORIES DEPLOYEMENT AND VHDL CODE GENERATION

In the main of verifying the well-function of this kind of wireless network three theories are proposed to be used as basic to model the WSNoC network. These models were created by combining those theories (Well explained in the three previously sections): the colored graph theory, The NoC theory, theWSNoC theory.
Then the present section leads us to validate also this behaviour is described in VHDL structure using VHDL theory already introduced before.

After modelling this system by combining these theories it obtains discharged interactive proofs so it can clearly say the strategy of modeling is enhanced.

TABLE I. STATISTIC PROOFS FOR WSNOC MODELLING WITH THEORIES

Element Name	Total	Auto	Manual	Reviewed	Undischarged
Color_WNoc	61	47	14	0	0
ModC001	17	15	2	0	0
ModC002	10	10	0	0	0
ModM001	17	8	9	0	0
ModM002	7	5	2	0	0
ModM003	5	5	0	0	0
ModM004	5	4	1	0	0

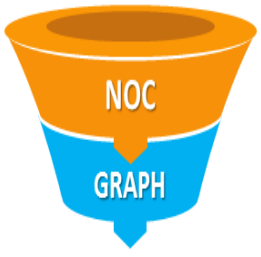
To check this system starting with the graph theory that represent the distributed system of NoC and then we collect it to the set of colored graphs properties to validate the management of the reconfiguration strategy for NoC wireless sensor network and finalizing our work by including VHDL theory to get a formal validation of VHDL behavior of our system.

Beginning always with the NoC theory the node can send, receive and forward packets some POs in this theory are discharged manually (Figure 6) but the most of rules related for the NoC system are convenant for the Rodin prover and by the result the NoC properties could be applied.

Element Name	Total	Auto	Manual	Reviewed	Undischarged
dftM0	**17**	**15**	**2**	**0**	**0**
INITIALISATI...	5	5	0	0	0
inv1	0	0	0	0	0
inv2	0	0	0	0	0
inv3	3	3	0	0	0
inv4	3	3	0	0	0
inv5	3	3	0	0	0
inv6	2	1	1	0	0
inv7	2	1	1	0	0
SEND	4	3	1	0	0
RECIEVE	5	4	1	0	0
FORWARD	1	1	0	0	0
inv8	0	0	0	0	0
inv9	0	0	0	0	0
inv10	0	0	0	0	0

Figure 6. The NoC theory application onto WSNoC system in Event-B

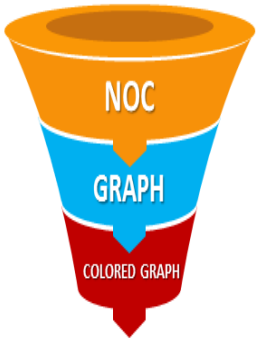
According to the NoC theory a node can process inside a network of NoC-based switches applying all the graph properties using the closure theory, in this level the number of manual discharged POs are increasing due to the complexity of network properties combined with the NoC properties (Figure 7.).



Element Name	Total	Auto	Manual	Reviewed	Undischarged
dftM01	**10**	**10**	**0**	**0**	**0**
INITIALISATI...	6	6	0	0	0
SEND	0	0	0	0	0
RECIEVE	0	0	0	0	0
COLORE_FAU...	3	3	0	0	0
FORWARD	0	0	0	0	0
inv1	0	0	0	0	0
inv2	0	0	0	0	0
inv3	2	2	0	0	0
inv4	1	1	0	0	0
inv5	2	2	0	0	0
inv6	0	0	0	0	0
inv7	2	2	0	0	0
inv8	1	1	0	0	0
inv10	1	1	0	0	0

Figure 7. The graph theory and the NoC theory application onto WSNoC system in Event-B

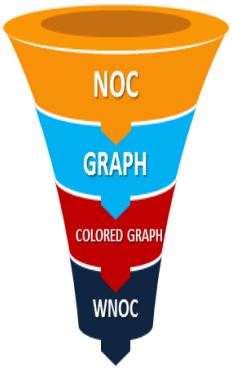
The NoC-based network may become into the failure state, so the coloration theory (extended from Closure theory) manage the failure zone (a number of faulty nodes) and colored it by the four available colors or adding to a pending list to be colored, here the number of manual Discharged POs still big according to the new strategy of handling a failure (Figure 8.).



Element Name	Total	Auto	Manual	Reviewed	Undischarged
dftM02	**17**	**8**	**9**	**0**	**0**
INITIALISATI...	3	2	1	0	0
SEND	0	0	0	0	0
RECIEVE	0	0	0	0	0
COLORE_FAU...	1	0	1	0	0
inv1	0	0	0	0	0
inv2	0	0	0	0	0
RECONFIG_N...	3	2	1	0	0
inv3	0	0	0	0	0
inv4	3	1	2	0	0
inv5	1	1	0	0	0
RECEIVING_BI...	5	2	3	0	0
inv6	1	1	0	0	0
MANAGE_CO...	3	0	3	0	0
FORWARD	0	0	0	0	0

Figure 8. The colored graph theory and the NoC theory application onto WSNoC system in Event-B

The WNoC theory express the emission of the reconfiguration data to reconfig_node and receiving bistream and forward packets till a faulty node returns to the initial state and the system with its all nodes are correct, it still have to color the failed nodes that are waiting to be recovered, the POs are all proved automatically which means that the coloring rules and NoC properties that are introduced previously made the system more suitable (Figure 9.).



Element Name	Total	Auto	Manual	Reviewed	Undischarged
dftM03	7	5	2	0	0
INITIALISATI...	2	2	0	0	0
SEND	0	0	0	0	0
RECIEVE	0	0	0	0	0
COLORE_FAU...	1	1	0	0	0
RECONFIG_N...	0	0	0	0	0
FORWARD	0	0	0	0	0
CORRECTED_...	3	1	2	0	0
MANAGE_CO...	1	1	0	0	0
RECEIVING_BI...	0	0	0	0	0
inv1	4	4	0	0	0
inv2	0	0	0	0	0

Figure 9. The WNoC theory application onto WSNoC system in Event-B

After checking all the functionalities of the WSNoC net

$\forall$ *Datatoroute,bufferin, BitstreamV,Fail_data, max_buff,*outportbus, inportbus,*occ_rqst_vect_out* • *Max_buff* $\in$ Vhdl_int $\wedge$ *max_buff=20*
$\wedge$ *Datatoroute* $\in$ archdecl(std_logic_vector(max_buff))
$\wedge$ *BitstreamV* $\in$ archdecl(std_logic_vector(max_buff))
$\wedge$ *Fail_data* $\in$ archdecl(std_logic_vector(max_buff))
$\wedge$ *bufferin* $\in$ archdecl(std_logic_vector(max_buff))
$\wedge$ *inportbus* $\in$ portdecl(in,std_logic_vector(max_buff))
$\wedge$ *outportbus* $\in$ portdecl(out,std_logic_vector(max_buff))
$\wedge$ *Occ_rqst_vect_out* $\in$ portdecl(out,std_logic_vector(4))
) $\Rightarrow$

by three grand theories to represent the properties of reconfiguration strategy, Network properties and the WSNoC architecture it last now to interpret this properties onto VHDL behaviour using the VHDL theory.

Case 1: SEND
Datatoroute:=Buffer_in
Datatoroute:=UpdateVect(datatoroute,9,0)
outportbus:= datatoroute
Case 2: FORWARD
Buffer_in:=inportbus
Datatoroute:=Buffer_in
Datatoroute:=UpdateVect(datatoroute,9,0)
outportbus:= datatoroute
Case 3: RECIEVE
Buffer_in:=inportbus

As it explains in the previous sections every events that represent an action for the WNoC network must be enriched with VHDL behaviour as follow:
In the ordinary case the data *datatoroute* must be sent from the input buffer *buffer_in* to the out port bus *outportsbus* this data must be transmitted from a node to another until it reach its destination and stored this received data from the input data bus *inportbus* into the buffer *buffer_in.*

Whereas nodes could have some fails during this process of exchanging data so it claimed that it had already a fail state in this Wireless sensors Network of NoC system and it handled this case by making a kind of reconfiguration protocol starting by adding a flag bit in the structure of every exchanging data.

Adress	FLAG	DATA
8bits	1bit	11bits

Figure 10. Structure of the transmitted data inside the WSNoC network

If this bit equals to 1 the data considered as a *Fail_Data* after that the node receive this data and considers its neighbours a faulty node, also this faulty node should change the value of the *occ_rqst_ out* to '1111' to do not receive any other data except the *bitstream* data it has sent by a node of reconfiguration just to make the faulty node re-change the value of *occ_rqst_vect* to '0000'.

$\forall$ *Datatoroute,bufferin, BitstreamV,Fail_data, max_buff,*outportbus, inportbus,*occ_rqst_vect_out* • *Max_buff* $\in$ Vhdl_int $\wedge$ *max_buff=20*
$\wedge$ *Datatoroute* $\in$ archdecl(std_logic_vector(max_buff))
$\wedge$ *BitstreamV* $\in$ archdecl(std_logic_vector(max_buff))
$\wedge$ *Fail_data* $\in$ archdecl(std_logic_vector(max_buff))
$\wedge$ *bufferin* $\in$ archdecl(std_logic_vector(max_buff))
$\wedge$ *inportbus* $\in$ portdecl(in,std_logic_vector(max_buff))
$\wedge$ *outportbus* $\in$ portdecl(out,std_logic_vector(max_buff))
$\wedge$ *Occ_rqst_vect_out* $\in$ portdecl(out,std_logic_vector(4))
) $\Rightarrow$

Case 4: FAILDATA SENDING
Datatoroute:=Fail_data
Datatoroute:=UpdateVect(datatoroute,9,1)
outportbus:= datatoroute

Vali(outportbus,9)=1
Case 5: FAILDATA RECEIVING
Buffer_in:=outportbus

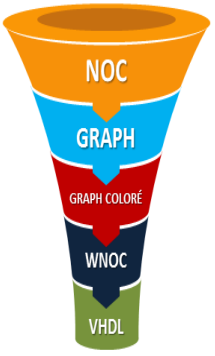
Case 6: DISABLE
Occ_rqst_vect_out= update(occ_rqst_vect_out,0,1)
Occ_rqst_vect_out = update(occ_rqst_vect_out,1,1)
Occ_rqst_vect_out = update(occ_rqst_vect_out,2,1)
Occ_rqst_vect_out = update(occ_rqst_vect_out,3,1)

Vali(BitstreamV,9)=0
Case 7: BITSTREAM SENDING
Datatoroute:=BitstreamV
Datatoroute:=UpdateVect(datatoroute,9,0)
outportbus:= datatoroute

Vali(BitstreamV,9)=0
Case 8: BITSTREAM RECEIVING
Buffer_in:= BitstreamV

Case 9: RELINK
Occ_rqst_vect_out = update(occ_rqst_vect_out,0,0)
Occ_rqst_vect_out = update(occ_rqst_vec_out t,1,0)
Occ_rqst_vect_out = update(occ_rqst_vec_out t,2,0)

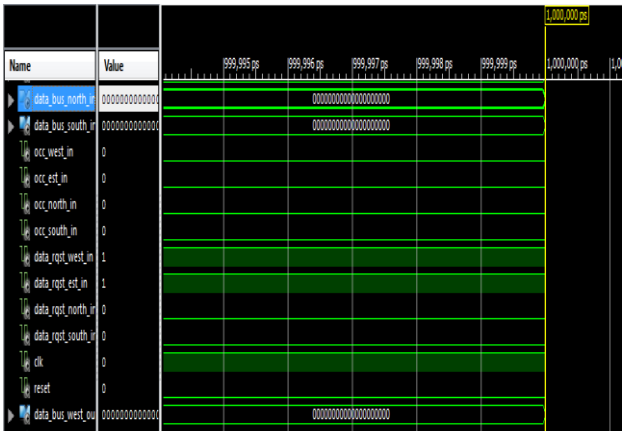
After combining all theories to express the properties of this system, VHDL variables are introduced to represented the VHDL behavior of this system and it's clear that all POs are automatically discharged which means that the systems is ready to the code generation step (Figure 9.).



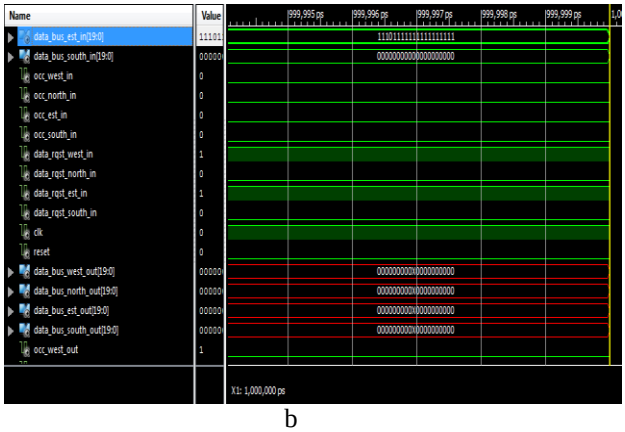
Element Name	Total	Auto	Manual	Reviewed	Undischarged
dftM04	**5**	**5**	**0**	**0**	**0**
INITIALISATI...	1	1	0	0	0
SEND	0	0	0	0	0
RECIEVE	0	0	0	0	0
COLORE_FAU...	1	1	0	0	0
RECONFIG_N...	0	0	0	0	0
FORWARD	0	0	0	0	0
CORRECTED_...	1	1	0	0	0
MANAGE_CO...	1	1	0	0	0
RECEIVING_BI...	0	0	0	0	0
inv1	5	5	0	0	0
COLORE_WAI...	1	1	0	0	0

Figure 11. The VHDL theory application onto WSNoC system in Event-B

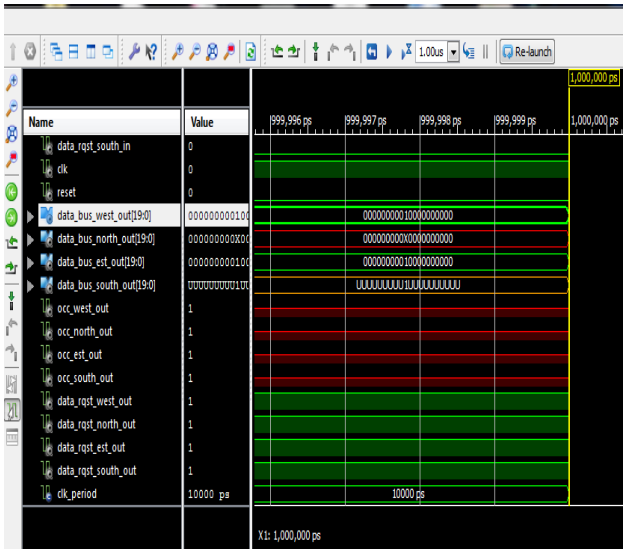
After checking formally about all the properties of the wireless sensors network of NoC system and when this phase is certainly validated the formal interpretation based on theories is just made to generate VHDL code representing all architectural properties and the result of realized code after the validation of wireless sensors network NoC, however we did a little demonstration where the implementation of the reconfiguration strategy will be presented by the creation of a node in a state without any failure Faulty-node and sending and receives data with the value of the flag bit is 0 (Figure12 case of a), another node type is the one that contains a software failure so it sends a fail_Data where the Flag bit is 1 to its neighbours(Figure12 case of b) waiting for any pending of a response to a correct the failure of this faulty-node(Figure12 case of c) and as you see that the strategy is well checked.



a



b



c

Figure 12. The result of simulation for the Wireless sensors network of NoC after the verification with Event-B theory
a.Sending and receiving data without Failure
b. Sending Flag_data in the case of failure
c. Starting the reconfiguration step

9. CONCLUSION

This paper addresses the key role played by formal modelling and verification in embedded systems design taking systems based on Network-on-chips (NoC) which is considered as the next generation of communication infrastructure in embedded systems. Modelling may be used at all stages of the development process from requirements analysis to system acceptance testing delivered by Xilinx Environment. In order to manage this system complexity, abstraction and refinement are key methods for structuring the formal modelling effort since they support separation of concerns and layered reasoning. The key features of Event-B are the use of set theory as a modelling notation, the use of refinement to represent systems at different levels and the use of mathematical proof to verify the consistency between refinement levels. Core Rodin supports rich mathematical theories in form of different types (integers, sets, relations, functions …), but many problems require additional mathematical structures (e.g., lists, trees, graphs, reals), these structures could be defined axiomatically (as an Event-B context), but polymorphism is not supported and no direct way to extend the provers of Rodin also no direct support for ensuring soundness of new operator definitions and proof rules , this lead to a new extension of modelling using theory plug-in which allows users to define new mathematical operators and data types, add proof rules to Rodin prover (Rules may be used interactively to be added to automated tactics) and generate soundness POs for new definitions and proof rules.

This formal validation based on theory using the Event-B theory plug-in tool give us a powerful auditing and generic covering for embedded systems. The generation of theories that care about NoC, the WSNoC and the colored graph theory separately allows us to generalize a kind of extending types that can be used and modify as necessary due to the environment where the system will be applied. Among the uses of these theories is the possibility of generating the VHDL behaviour for such a system based on the theory VHDL. However it still have to check the correct passage strategies given during software and hardware failure for nodes in the network and how a node can detect a failure, formal validation based on theory for these points give us a global view possible scenarios for achieving a self-adaptive wireless sensors network of NoC.

ACKNOWLEDGMENT

This research is a part of research project supported by the thematic research agency (ATRST) of research direction (Algerian Ministry of High Education and Scientific Research.).

REFERENCES

[1] Bergeron J.:'Writing Testbenches: Functional Verification of HDLModels', (Kluwer AcademicPublishers, 2003,2nded, vol. 2, Norwell, MA, USA).

[2] Piziali A.: 'Functional Verification Coverage Measurement andAnalysis', (Kluwer AcademicPublishers, 2004 1st ed, Massachusetts, USA).

[3] Sangiovanni-Vincentelli A., Carloni L., Bernardinis F. D., SgroiM.:'Benefits and challenges for platform-based design', IntConf DAC '04: Proceedings of the 41st annual conference on Design automation, New York, NY, USA, 2004, ACM, p. 409–414.

[4] 'FP-7 DEPLOY Integrated Project: Industrial deployment ofadvanced system engineering methods for high productivity anddependability'. ICT-214158. http://deploy-project.eu, accessed 26 August 2015.

[5] Abrial, J-R.:' System and Software Engineering in Modelling in Event-B', (Cambridge University Press, 1st ed , 2010,New York, NY, USA).

[6] Abrial J-R.:' assigning programs to meanings'in The B-book. (Cambridge University Press, 1996, New York, NY, USA).

[7] Butler M., Hallerstede S.: 'The Rodin Formal Modelling Tool'. BCS-FACS Christmas 2007 Meeting - Formal Methods In Industry, London, December 2007.

[8] Abrial J-R., Cansell D.:'Click'n Prove: Interactive Proofs within Set Theory', In Lecture Notes in Computer Science: Theorem Proving in Higher Order Logics, 2003, pp. 1-24.

[9] Castéran P., Filou V., Mosbah M.:'Formal Proofs of Local Computation Systems'.Technical Report 0, LaBRI - University of Bordeaux, 2009.

[10] Sannelis D., WirsingM.: 'A kernel language for algebraic specification and implementation extended abstract',computer sciences faculty, Abstract Software Specifications, Springer , 1983,**158**, pp. 413-427.

[11] Burstall R. M.,Goguen J. A.,'The semantics of clear, a specification language',Abstract Software Specifications,Springer ,Lecture Notes in Computer Science,1980,**86** , pp. 292-332.

[12] Dos Reis G., Järvi J.:' What is Generic Programming?', Intconference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'05), San Diego, California, October, 2005,pp. 1-11.

[13] Gibbons J., Paterson R., 'Parametric datatype-genericity'. Technical repport ,parametric, Oxford University Computing Laboratory,July 2007,pp.1-9

[14] Hariche A., Belarbi M., Daoud H. :'Based B Extraction of QNoC architecture properties', Int workshop on Mathematic and Computer Sciences, Tiaret , December 2012,MOMAJ issue 01Vol. 02 pp. 8-13

[15] Andriamiarina M. B., Méry D., Singh N. K..'Revisiting Snapshot Algorithms by Refinement-based Techniques', Computer Science and Information Systems, 2012, **11**, (1), pp.251–270.

[16] 'Deploy project Rodin tools Documentation', http://wiki.event-b.org/index.php/Theory_Plug-in accessed 21 july 2015

[17] 'Deploy project Rodin theory plug-in ', http://users.ecs.soton.ac.uk/asf08r/org.eventb.theory.updateSite/ accessed 21 July 2015.

[18] Allan A.;' International Technology Roadmap for Semiconductors', July 2008 technical report, San Francisco, USA, pp.1-23.

[19] Gajski D.D., Kuhn R. H.: 'New VLSI Tools' , IEEE Computer Society Press, 16, (12),1983,pp. 11–14.

[20] Kasuya A., Tesfaye T.: 'Verification methodologies in a TLM-to-RTL designFlow', IntConf DAC '07: Proceedings of the 44th annual conference on Design automation, New York, NY, USA, 2007, ACM, p. 199–204.

[21] Cherif S., RafiqQuadri I., Meftali S., Dekeyser J-L.:'Modeling reconfigurable Systems-on-Chips with UML MARTE profile: an exploratory analysis'. 13th Euromicro Conference on Digital System Design (DSD 2010), September 2010, Lille, France,<inria-00525004>,pp.1-9.

[22] Xing J., Theelen B.D., Langerak R., van de Pol J., Tretmans J., Voeten J.P.M.. :'From POOSL to UPPAAL: Transformation and Quantitative Analysis', IntConf, Proceedings of the 10th International Conference on Application of Concurrency to System Design, Braga, Portugal, June 23-25, 2010, pp.47-56.

[23] Nouri A., Bensalem S., Bozga M. , Delahaye B. , Jegourel C., Legay A. :' Statistical model checking QoS properties of systems with SBIP', International Journal on Software Tools for Technology Transfer ,April 2015, **17**, (2), pp.171-185.

[24] Manfredi S.: ' Reliable and energy-efficient cooperative routing algorithm for wireless monitoring systems', IET Wireless Sensor Systems, June 2012,**2**, (2), pp. 128–135

[25] Tomizawa M. , Yamabayashi Y. , Kawase N. , Kobayashi Y.: ' Self-healing algorithm for logical mesh connection on ring networks', IET Electronics Letters, September 1994,**30**, (19), pp. 1615 – 1616.

[26] Mokhov A.: ' Algebra of switching networks ', IET Computers & Digital Techniques, July 2015,**9**, (4), pp. 197 – 205

[27] Kamali M., Laibinis L.,PetreL., SereK. :' Self-Recovering Sensor-Actor Networks', in Proceedings FOCLASA,Distributed, Parallel, and Cluster Computing (cs.DC); Logic in Computer Science (cs.LO),France, July, 2010,EPTCS 30, 2010, pp. 47-61.

[28] Andriamiarina M.B, Mèry D.: 'Stepwise development of distributed vertex coloring algorithms'. PhD thesis, Nancy University, 2011

[29] M. Andriamiarina, H.Daoud, M.Belarbi, D.Méry and C.Tanougast; 'Formal Specification and Verification of an architeture based wireless network oriented NoC', 2012,MOMAJ Vol. 02 page 1-4 , 2014.

[30] A. Hariche. M. Belarbi. H. Daoud: 'new Operators-Based Approach for the Event-B Refinement: QNoC Case Study'IEEE International Conference on Mechatronics IEEE ICM'2013, beirut ,Lebanon, December. 2013.

[31] M. Heil, C. Tanougast, C. Killian, and A. Dandache, "Self-Organized Reliability Suitable for Wireless Networked MPSoC,"in Proc. of the Int. Conf. on Field Programmable Logic and Applications, Prague, Czech Republic, 2009, pp. 326–331.

[32] Andriamiarina M.B, Mèry D.: 'Stepwise development of distributed vertex coloring algorithms'. PhD thesis, Nancy University, 2011

[33] SwaminathanaV.,Jeyanthib P.: ' On super vertex-magic labeling', Journal of Discrete Mathematical Sciences and Cryptography., 2005, **8**, (2), pp. 217-224

[34] Edmunds A., Butler M., Maamria I., Silva R., Lovell C.:'Event-B Code Generation: Type Extension with Theories' Abstract State Machines, Alloy, B, VDM, and Z, Lecture Notes in Computer Science,2012,**7316**, pp 365-368.

Abdelhamid HARICHE Currently is PHD student of embedded systems and real time in the University of Tiaret in Algeria. Having graduated from the University Ibn khaldoun of Tiaret with a Master degree in Computer Science Engineering in 2012 where the title of the theme was :' Design and implementation of an environmental validation for the implementation of embedded applications'. He received his major on computer science in 2010 from the University of Ibn khaldoun where the title of the graduation project was''Identification of persons by their signatures'. He is a member of the LIM research laboratory that focused on many fields that applies mathematics and computer sciences: applied formal methods, Parallel Computing Design and Validation, Verification of Embedded and Real Time Systems.

Mostefa BELARBI is lecturer at Computer Science Department Faculty of Mathematics and Computer Science of University of Tiaret – Algeria. He received the diploma of Doctor in Computer Science at INSA Lyon-France in December 2003. Title of the thesis is : temporal validation of real-time multitasking applications based on communicating timed automata. He received the diploma of Master in Software Engineering at university of Sciences and Technology of Oran- Algeria in November 1997 (with highest honours), the title of thesis: An algebraic approach for program construction. He prepared the diploma of Engineer of state in computer science at University of Senia (Oran-Algeria) in September 1988. He is Member of LIM (Computer science and mathematics) Laboratory – University Ibn Khaldoun of Tiaret. Its Domains of research : applied formal methods, Parallel Computing Design and Validation, Verification of Embedded and Real Time Systems.

Abdallah Chouarfia was born in Algeria, graduated with Engineer Degree in Computer Science from National Institute of Computer Science, Algiers, in 1981. He received the PhD degree in Computer Science from the Paul Sabatier University in Toulouse, France in 1983. He was a Senior Lecturer in the Computer Science Department at Science and Technology University, Oran Algeria from 1987-2015. He is currently a Professor at Science and Technology University, Oran Algeria. His research interests include software engineering, information systems interoperability, networking, web engineering, pervasive systems.