République Algérienne Démocratique et Populaire

Minsitère de l'Enseignement et de la Recherche Scientifique

Université de Blida 1
Faculté des Mathématiques et d'Informatique
Département d'Informatique

Mémoire de fin de cycle

En vue de l'obtention du Diplôme de **Master**

Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Sécurité Système Informatique

Thème

Spécification Et Vérification Formelle De La Sécurité D'une Architecture à Base De Réseau Sans Fil Orientée NoC

Réalisé par :

Taibouni Sabrina

Encadré par :

M. Daoud Hayat

Année universitaire 2020/2021

REMERCIEMENTS

Nos premiers remerciements s'adressent à Dieu le tout puissant qui par sa bonté et sa miséricorde nous a permis d'avoir le courage,la foi et la volonté de mener à bien ce travail.

Je veux remercier mes parents pour leur encouragement
Au terme de cette étude, nous souhaitons remercier chaleureusement notre encadreur de mémoire M. daoud hayat pour leur suivi et ses encouragements, ses orientations et ses précieux conseils.

D'une façon général on dit merci à tous qui a aidé d'une maniéré morale et matérielle .
Nous remercions également tous les professeur qui ont contribué de près ou de loin à notre formation universitaire, sans oublier tout personne qui nous a aidés à mener à terme notre projet.
Enfin, je tient également a remercier toutes les personnes qui ont participé de près ou de loin a la réalisation de ce travail.

Merci à tous

DÉDICACES

Je dédie cet humble travail à mon père et à ma mère bien-aimée, que Dieu la préserve, qui m'a aidé et soutenu..
sans oublier mes frères et tout les membre de famille.
J'adresse ma reconnaissance, ma gratitude à mon professeur encadrant M.daoud hayat de m'avoir fait bénéficier de ses compétences, ses qualités humaines et de sa disponibilité pour la réalisation de ce mémoire.
Nous tenons aussi a exprimer nos sincères remerciements a tous les professeurs qui nous ont enseigné et qui par leur compétences j'ai soutenu dans la poursuite de mes études.
Aussi nous remercions nos famille et nos amis qui par leur prière et leur encouragements on a pu surmonter tous les obstacles.
A toutes les personnes qui nous ont apportés de l'aide.

T. Sabrina

RÉSUMÉ

Les systèmes embarques ont un fort impact dans la vie quotidienne par exemple ,les systèmes comme les téléviseurs,les téléphone cellutaires,cartes de crédits sont utilisée pour les personnes,tandis que d'autres systèmes ,comme les réseaux,les télécommunication,la distribution et intégrés dispositifs,les super ordinateurs sont utilisée par des organisation telles que les entreprise,les gouvernements,les nations....Plusieurs pays,basant sur système MPSOc à base de NoC,Ces systèmes doivent être fiable,et leurs spécifications et la conception doivent être claire ,compréhensibles et doivent des règles précises et ils doivent éviter les défauts, les échecs et s'ils ne peuvent pas ,ils devraient au moins être tolérant aux pannes et de sécurité ,par conséquent en raison de ces exigences,"Vérification formelle " peut être utile pour obtenir une assurance et la garantie de leur exactitude par rapport aux questions de sureté et de sécurité.Event-B est définie comme une méthode de développement formelle mettant en valeur aussi bien les aspects statiques que dynamiques d'un système Tirant comme un cas d'étude le système multi-nœud sans fil qui représente un système distribué intégrant MPSoC en réseau sur puce (NOC) permet de communique grâce à la communication ZigBee.Dans ce travail nous proposons de spécifier et prouver formellement ce type de système en instanciant un modèle basé sur graphe colorés,le langage de description utilisé est Event-B qui s'appuie sur l'outils Rodin.Cette description permet d'amélioré la spécification en vue d'augmenter les performances d'une solution de fiabilité auto-organisée basé sur les MPSoC sans fil multi- nœud

Mots-Clés : Systèmes Embarqués,Méthode formelles ,B-Event ,NoC ,Réseau sans fil ,Algorithme de coloration de graphes.

ABSTRACT

Embedded systems have a strong impact in daily life for example, systems like televisions, cell phones, credit cards are used for people, while other systems, like networks, telecommunications, distribution and integrated devices, supercomputers are used by organizations such as business, governments, nations .... Several countries, based on NoC based MPSOc system, These systems must be reliable, and their specifications and design must be clear, understandable and must have precise rules and they must avoid faults, failures and if they cannot, they should at least be fault tolerant and safe, therefore due to these requirements, ”Formal verification” can be useful in obtaining assurance and assurance of their accuracy in relation to safety and security matters. Event-B is defined as a method of development fo rmelle highlighting both the static and dynamic aspects of a system Pulling as a case study the wireless mufti-node system which represents a distributed system integrating MPSoC in network on chip (NOC) allows to communicate thanks to the communication ZigBee. In this work we propose to specify and formally prove this type of system by instantiating a model based on colored graphs, the description language used is Event-B which is based on the Rodin tool. the specification to increase the performance of a self-organized reliability solution based on MPSoC wireless mufti- node

textbfKeywords :Embedded Systems, Formal method, B-Event, NoC, Wireless network, Graph coloring algorithm.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABRÉVIATIONS

Abréviations	signification
SoC	System on Chip
NoC	Network on Chip
MPSoC	Multiprocess System on Chip
FPGA	Field Programmable Gate Arrays
ZGS	Zone Globalement Sure
ET	Éléments de Test
SDF	Sur De Fonction

INTRODUCTION GÉNÉRALE

Les systèmes embarqués prennent une place de plus en plus importante dans notre société, ils servent à contrôler, réguler des dispositifs électroniques grâce à des capteurs, embarqués dans des robots, des véhicules spatiaux, etc. Ces systèmes embarqués sont souvent utilisés par le public dans la vie de tous les jours sans même qu'on ne s'en rende compte, par exemple dans les systèmes de freinage d'une voiture, le contrôle de vol d'un avion, automobile, géo positionnement par satellite, lecteurs multimédia, téléphonie portable, etc.
Les progrès technologiques des systèmes embarqués (SE) ont permis d'intégrer sur une même puce plus de fonctionnalités ce qui a conduit à la définition d'un nouveau paradigme de systèmes embarqués : les systèmes sur puce (SOC : Système on Chip). Les SoCs sont élaborées par l'intégration de différents composants hétérogènes d'un système électronique entier sur une même puce : des processeurs, de la mémoire, circuit RF, des interfaces analogiques/digitales, etc.
Les méthodes formelles ont la capacité à produire des systèmes critiques pour de grands projets industriels, et cela par une création d'un modèle mathématique initial qui peut être raffiné de manière formelle en plusieurs étapes jusqu'à ce que le raffinement final contienne assez de détails pour une implémentation.Les systèmes embarques sont base sur diffèrent architecture, dans notre projet on a choisis une architecture multi-nœud auto-organise base sur réseau sans fil orienté Noc Un système embarqué est vulnérable à des attaques au niveau physique contre ses composants (processeurs, mémoires, bus de communication, etc.) car dans de nombreux cas, l'adversaire (qui peut même parfois être l'utilisateur légitime) peut avoir un accès physique direct au système. Par exemple, l'utilisateur d'une set-top box peut vouloir récupérer les flux vidéos diffusés, un pirate peut vouloir récupérer les secrets contenus dans une carte à puce ou un ennemi peut vouloir récupérer le programme de

vol d'un drone militaire qu'il a abattu. Ces attaques remettent en cause la sécurité (confidentialité, intégrité, authentification, disponibilité, etc.) des applications exécutées et des données manipulées par le système embarqué ces systèmes peuvent être endommagées par des menaces de sécurité qui rend le système en état critique tel que :les Terroristes cause la destruction ,les Ennemis causer des désagréments , Espions pour avoir des informations ,Concurrents industriels prise de part de marché ,Pirates vol d'information , Hackers défi ...Afin de garantir la forte sécurisation contre ces menaces et assurer le bon fonctionnement de ces systèmes on a proposé une vérification formelle par la méthode évent B Pour ce faire on a organisé notre travail est comme suite

Le premier chapitre est une introduction générale à ce domaine de système embarques,Après en avoir posé les grands principes avec des généralités Sur les systèmes sur puce SoC les réseaux sur puce NoC, et leurs composants et caractéristiques

Le deuxième chapitre présente les techniques de vérification par une vue globale sur les méthodes formelles qui prouvent la sûreté de fonctionnement des systèmes architecturaux. Plus précisément explique comment est procédé une vérification formelle avec la méthode B évènementiel (Event-B). Ainsi que les avantages qu'une telle méthode apporte par rapport au domaine de conception. L'adaptation à la modélisation des systèmes par raffinements (correction par construction) au travers d'une description avec l'outil RODIN associé à la méthode B.

Le troisième chapitre représente une architecture mufti-nœud auto-organise propose par le docteur Mikael HEIL, ce chapitre détail le mécanisme délocalisé de tests base sur le système auto-organise propose et permettant d'assurer la sureté de fonction des structure de communication sur puce mettent en œuvre l'interaction mutuelles de l'ensemble des entités constituants le réseau de nœud auto-organise re configurable

Le dernier chapitre fait l'objet de l'injection de la théorie des graphes colorés en Event-B et présente la spécification et la vérification de projet et prouver comment la méthode évent B garantir les grands aspects de sécurité dans notre architecture propose

Ce document se termine par une conclusion générale résumant l'ensemble de ce travail.

CHAPITRE 1

GÉNÉRALITÉ SUR LE SYSTÈME EMBARQUÉ

1.1 Introduction

Un système embarqué est un système électronique et informatique autonome ne possédant pas des entrées/sorties standards comme un clavier ou un écran d'ordinateur. Il est piloté dans la majorité des cas par un logiciel, qui est complètement intégré au système qu'il contrôle. On peut aussi définir un système embarqué comme un système électronique soumis à diverses contraintes. Les concepteurs des systèmes électroniques sont aujourd'hui confrontés à la complexité croissante des algorithmes mis en œuvre et à la variété des cibles potentielles FPGAs et/ou DSPs. Actuellement, il n'est pas rare que ces systèmes intègrent plusieurs douzaines voire des centaines de processeurs. A l'origine, ce sont des systèmes matériels et logiciels intégrés dans des avions militaires ou des missiles. Ensuite dans le civil : avions, voitures, machine à laver...! La machine et le logiciel sont intimement liés et noyés dans le matériel et ne sont pas aussi facilement discernables comme dans un environnement de travail classique de type PC. On mentionne les contraintes physiques fortes : dimensions, poids, taille, autonomie, consommation,fiabilité, contraintes temporelles (temps réels).

À l'énoncé de la place que prennent les systèmes embarqués sur notre vie, il est facile de comprendre à quel point leur sécurité est critique. Sont-ils pour autant protégés de toutes menaces ? Bien au contraire, excepté dans une certaine mesure les systèmes dont la sécurité est une contrainte par définition, par exemple les domaines d'applications militaires et bancaires, les systèmes embarqués sont très fortement menacés, non seulement par des attaques logicielles et sur les réseaux (comme tous systèmes informatiques), mais aussi par des at-

taques physiques ciblant leurs parties électroniques et micro électroniques. ils doivent avoir conscience des menaces qui peuvent peser sur les systèmes embarqués qu'ils développent et avoir une idée des possibilités de protection, c'est ce que nous propose dans ce chapitre

1.2 Système Embarqué

1.2.1 Définition1

C'est un système électronique et informatique autonome, qui est dédié à une tâche bien précise Un ensemble de logiciels et de matériels conçu pour une application spécifique contrairement à un système qui peut effectuer toutes sortes de tâches tel qu'un ordinateur de bureau.

1.2.2 Définition2

" Embedded système " tout système conçu pour résoudre un problème ou une tache spécifique mais n'est pas un ordinateur d'usage général.
•Utilisent généralement un microprocesseur combiné avec d'autres matériels et logiciel pour résoudre un problème de calcul spécifique

1.2.3 L'utilisation de système embarqué

La plus grande partie des systèmes informatiques utilisés de nos jours sont des systèmes embarqué

Figure 1.1 – Les Domaines D'utilisation De Système Embarque

D'autre utilisation de système embarque :

Domaine d'utilisation	Application majeurs
Espace	Compression de donnée exécution.
Médecine	Diagnostic par image(ultrason...) - Analyse électrocardiogramme
Commerce	Compression d'image et du son pour la présentation multimédia,Effets spéciaux dans le cinéma,la vidéo conférence
La Téléphone	Compression de la voix et de donné Réduction de l'écho signal multiplexing filtrage
Militaire	Radar Sonar Guidage de cible Sécurisé la communication
Industrie	Prospection d'eau et de pétrole,Traitement et contrôle Test CAD et outils de conception
Science	Modélisation et simulation , Acquisition de données

1.2.4 Caractéristiques et contraintes

Fonctionnement en Temps Réel :

La validité d'un résultat (et sa pertinence) dépend du moment où il est délivré.

Faible consommation :

Batterie de 8 heures et plus (PC portable : 2 heures)

Fonctionnement critique pour la sécurité des personnes. Sûreté :

Le système doit toujours fonctionner correctement.[1]

1.3 SoC

1.3.1 Définition

Un " **système sur une puce** ",souvent désigné dans la littérature scientifique par le terme anglais " **système on a chip** " (où son abréviation **SoC**), est un système complet embarqué sur une seule puce (" circuit intégré "),pouvant comprendre de la mémoire, un ou plusieurs microprocesseurs, des périphériques ,interface, ou tout autre composant nécessaire à la réalisation de la fonction attendue.

1.3.2 L'Evolution de SoC

Aujourd'hui, ,interconnexion au sein ,un composant électronique complexe (SoC : système On Chip) est un problème important en termes de coût du silicium, de sa performance, de sa consommation.
Les architectures à base de bus sont clairement insuffisantes.
Avant les MPSoCs ont été fondés sur les bus, et maintenant, ils sont basé sur les NoC.
Les systèmes de type NoC (Network on Chip), véritables protocoles de réseau sur puce, sont à la fois les plus prometteuses.

1.4 NoC

Créée en début 2003,la **star-up française Arteris** propose la notion de réseau à l'intérieur de la puce, qu'elle appelle NOC, Réseau sur une puce (NoC) est relativement une nouvelle approche visant à intégrer les circuits sur une plate-forme SoC.[2]

1.4.1 Définition

Network-on-Chip ou Network-on-a-Chip (NoC or NOC) ou en français réseau sur une puce est une technique de conception du système de communication entre les cœurs sur les Système on Chip (SoC). Le NoC applique les théories et méthodes de réseau aux communications à l'intérieur d'une puce et permet ainsi l'amélioration des performances par rapport aux interconnexions de bus et commutateur matriciel conventionnelles. Le NoC améliore la scalabilité des SoCs et assurer la communication entre les éléments de SoC , et il gère dynamiquement les différent flux de donnes.

1.4.2 Les Composants d'un NoC

Un réseau sur puce est constitué de trois type de composant qui sont :
Les interfaces réseau se sont les éléments qui permettent d'accéder au réseau sur puce
Les nœuds de routage sont charger de diriger les donnes qui ils reçoivent vers leur destination
Les liens de communication assurent le transfert de donnes entre deux éléments de réseau sur puce.[3]

1.4.3 Les Caractéristiques des architectures basées sur les NoCs

L'objectif principal d'un réseau sur puce est de fournir un système permettant d'échanger des données entre différentes ressources de traitement (ou cœur d'IP). Le réseau est composé des nœuds (aussi appelés Switchs ou routeurs). Les données transitent entre les nœuds grâce à des liens (ou canaux) de communication point à point. L'ensemble formé par les nœuds et les liens constitue le système de communication. Chaque nœud comprend un ensemble de ports de communications. Ces ports connectés aux liens permettent de relier les nœuds entre eux. Suivant les architectures

Les ressources sont connectées directement aux nœuds ou par l'intermédiaire de liens. Les nœuds prennent les décisions de routage, ainsi que celles d'arbitrage.[4]

1.4.4 Les différentes topologies des NoCs

La topologie d'un réseau définit comment les routeurs sont inter connectés entre eux en utilisant les liens de réseau. Elle spécifie l'organisation physique du réseau et elle est donc souvent modélisée par un graphe. Comme pour les réseaux informatiques, de nombreuses topologies sont envisageables pour construire un NoC [5]. (La figure 5) Introduit sélectivement les topologies les plus couramment utilisées dans la conception des NoCs.

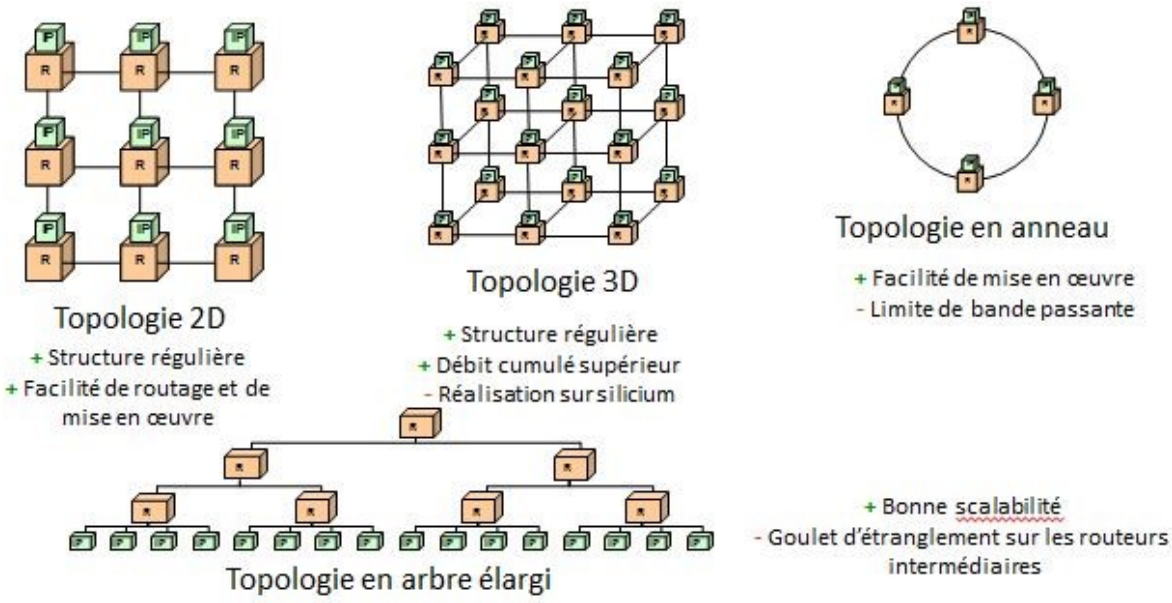


Figure 1.2 – Différentes topologies des NoCs.

1.4.5 Les Techniques de testabilité des NoCs

Concernant le système micro électronique embarqué du réseau géré, le SOCs devient plus sensible aux phénomènes produisant des défauts, et le comportement de son support de communication centrale peut être affecté par des différents facteurs (comme électro-migration), et ceci peut mener à un échec du système sur puce entier (SOC).
Il existe deux familles des techniques qui sont généralement employés dans les NoCs : (Technique en ligne Bist et hors ligne Dft)
La technique en ligne Bist :(Built-In-Self-Test)
Les techniques de BIST comprennent des circuits de test à l'intérieur du NoC et exiger plus de ressources matérielles, mais moins de temps à effectuer.
La technique hors ligne Dft :(Design-For-Testabilty)
Les techniques DfT nécessitent un module externe de test qui s'injectent des vecteurs et des ressources matérielles sans coût, mais nécessite plus de temps pour être réalisée.[6]

1.5 FPGA

1.5.1 Historique Et Terminologie

Très tôt dans l'histoire de la micro-électronique, plusieurs dispositifs ont été proposés pour rendre les circuits intégrés plus flexibles par programmation ou configuration. Deux niveaux de programmation ont été proposés : une programmation directe et totale par l'utilisateur et une programmation partielle en fonderie.
Les premiers à apparaître furent les PLA (Programmable Logic Arrays), des dispositifs très simples et totalement programmables par l'utilisateur. L'apparition de ces circuits date du début des années 1970. Le « plan ET » est composé de deux réseaux de fils qui se croisent (sans contact avant configuration) : celui des fils des entrées xi (et leurs complémentaires), et celui des fils en entrée des six portes ET. Le « plan OU » est composé de deux réseaux de fils : celui des produits pi et celui des entrées des deux portes OU.

1.5.2 Définition

L'acronyme FPGA (Field Programmable Gate Arrays) désigne un circuit intégré composé d'un réseau de cellules programmables chaque cellule est capable de réaliser une fonction,choisie parmi plusieurs possibles .les interconnexions sont également

programmables[7]
Lorsque un système doit être implémenté dans un FPGA un fichier de configuration (bitstream) est charge ce fichier de configuration indique toutes les interconnexion a réaliser dans FPGA ainsi le contenu et le registre utilisés

1.5.3 L'intérêt de FPGA

Il est complété généralement d'un circuit FPGA jouant le rôle de coprocesseur afin de proposer des accélérations matérielles au processeur. Et augmenter la performance et la fiabilité de système
Afin d'augmenter les performance des FPGA un certain nombre de connexion interne et de blocs matériels (bloc RAM . . .) sont intègres et présent de manier permanente au sein de FPGA , parmi ces blocs , on trouve le composant réalisant la configuration dynamique et partielle , ce module est appelé Infernal Configuration Accessit Port (ICAP) et permet d'accéder a la mémoire de configuration du FPGA contenant le fichier de configuration (bitstream) .[8]
Les FPGA peuvent offrir un certain nombre d'avantages par rapport à une technologie ASIC à fonction fixe telle que les cellules standard. Normalement, la fabrication des ASIC prend des mois et leur coût s'élèvera à des milliers de dollars pour obtenir l'appareil. Mais, les FPGA sont fabriqués en moins d'une seconde, le coût sera de quelques dollars à mille dollars. La nature flexible du FPGA a un coût important en termes de surface, de consommation d'énergie et de retard. Par rapport à un ASIC à cellule standard, un FPGA nécessite 20 à 35 fois plus de surface, et les performances de vitesse seront 3 à 4 fois plus lentes que l'ASIC. Cet article décrit les bases du FPGA et le module d'architecture FPGA qui comprend un bloc d'E/S, des blocs logiques et une matrice de commutation. Les FPGA font partie des nouveaux domaines de tendance du VLSI. Par conséquent, ceux-ci sont utilisés dans Projets basés sur VLSI pour les étudiants en génie électronique.

Figure 1.3 – Circuit logique programmable

1.5.4 Les Types de FPGA basés sur les applications

Les matrices de portes programmables sur site sont classées en trois types basés sur des applications telles que les FPGA bas de gamme, les FPGA milieu de gamme et les FPGA haut de gamme.

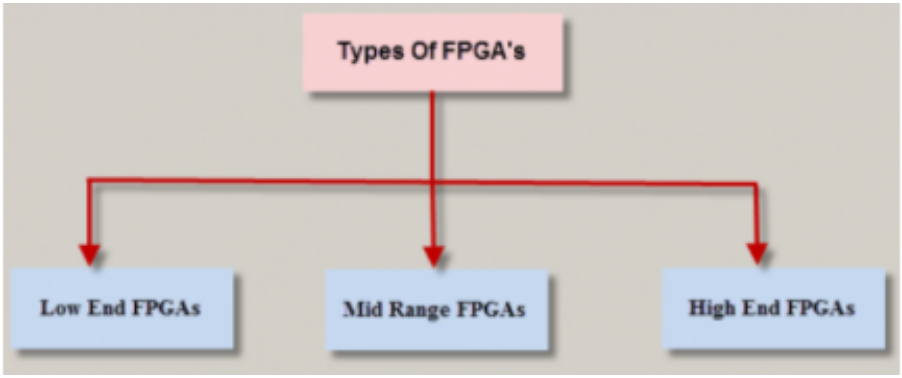


Figure 1.4 – Types de FPGA

FPGA bas de gamme :

Ces types de FPGA sont conçus pour une faible consommation d'énergie, une faible densité logique et une faible complexité par puce. Des exemples de FPGA bas de gamme sont la famille Cyclone d'Altera, la famille Spartan de Xilinx, la famille fusion de Microsemi et le Mach XO/ICE40 de Lattice semi-conducteur.

FPGA milieu de gamme :

Ces types de FPGA sont la solution optimale entre les FPGA bas de gamme et haut de gamme et ils sont développés comme un équilibre entre les performances et le coût. Des exemples de FPGA de milieu de gamme sont Arria d'Altera, les séries Artix-7/Kintex-7 de Xlinix, IGL002 de Microsemi et les séries ECP3 et ECP5 de Lattice Semi-conductrice.

FPGA haut de gamme :

Ces types de FPGA sont développés pour une densité logique et des performances élevées. Des exemples de FPGA haut de gamme sont la famille Stratix d'Altera, la famille Virtex de Xilinx, la famille Speedster 22i d'Achronix et la famille ProASIC3 de Microsemi.

1.5.5 Les Applications de FPGA

Les FPGA ont connu une croissance rapide au cours de la dernière décennie, car ils sont utiles pour un large éventail d'applications. L'application spécifique d'un FPGA comprend le traitement du signal numérique, la bio informatique, les contrôleurs de dispositifs, la radio définie par logiciel, la logique aléatoire, le prototypage ASIC, l'imagerie médicale, l'émulation de matériel informatique, l'intégration de plusieurs SPLD, la reconnaissance vocale , la cryptographie, le filtrage et l'encodage de communication et bien d'autres.
Habituellement, les FPGA sont réservés à des applications verticales particulières où le volume de production est faible. Pour ces applications à faible volume, les principales entreprises paient en coûts de matériel par unité. Aujourd'hui, la nouvelle dynamique de performance et le nouveau coût ont étendu la gamme d'applications viables.

Figure 1.5 – Applications du FPGA

Certaines applications FPGA plus courantes sont : aérospatiale et défense, électronique médicale, prototypage ASIC, audio, automobile, diffusion, électronique grand public, systèmes monétaires distribués, centre de données, calcul haute performance, industriel, médical, instruments scientifiques, systèmes de sécurité , vidéo et image Traitement, communications filaires, communications sans fil .

1.6 Sécurité des systèmes embarqués

Les réseaux et systèmes informatiques prennent un rôle et une place chaque jour plus importants dans tous les aspects des activités humaines. Les situations multiples dans lesquelles il est indispensable de savoir définir et garantir leur sécurité concernent aussi bien les activités professionnelles qu'associatives ou personnelles. La sécurité se décline alors de nombreuses manières, par exemple dans le cadre des transactions électroniques mais également dans la protection des données, des informations, des personnes et des biens. Dans ce contexte, la sécurité comprend en particulier celle des systèmes, des logiciels, des protocoles, des architectures globales,des composants matériels, des réseaux tan filaires ou optiques que radios, des équipements d'extrémités, des moyens de stockage de l'information.

1.6.1 Les menaces de sécurités

•Terroristes (but : destruction)
•Ennemis (but : causer des désagréments)
•Espions, services de renseignement (gouvernementaux, industriels) (but : informations)
•Concurrents industriels (but : prise de part de marché)
•Pirates (but : vol)
•Hackers (but : défi, jeux).

1.6.2 Les Objectifs des attaques

Accéder aux données privées et secrètes pour accéder à un niveau supérieur d'information(exemple : clé de cryptage pour décoder un message crypté)
•Accéder aux données privées pour les modifier ou les détruire
•Copier les données de conception pour reconstruire ou améliorer un système
•Prendre en main un système pour le détourner de sa fonction ou pour le détruire.

1.6.3 Classement des attaques

Il y'a des différentes manières pour classer ces attaques, et on peut les classés en deux catégories

Les attaques logicielles

Infections informatiques, codes malveillants :
Programmes simples qui exécutent un programme en tâche de fond (dans le cas de la bombe logique le départ du programme est retardé). Ils ne peuvent pas se reproduire, ils servent souvent à ouvrir une brèche cachée dans le système pour y faire pénétrer un pirate, par exemple : Cheval de Troie et Bombe Logique.
Programmes auto reproducteurs par exemple : Un ver est un virus très virulent qui se diffuse de façon planétaire.
Vulnérabilités logicielles :
La plus part des systèmes embarqués ont une zone mémoire extérieure au SoC principal. Des données et des instructions sont échangés entre le processeur et la mémoire sur un ou plusieurs bus(adresse, instruction, donnée)Menaces (logicielle ou matérielle) : Lecture non autorisée de données, Injection de code,Modification des données.

Les attaques matérielles

Vulnérabilités matérielles : Plusieurs types de menaces peut être considérés en fonction de l'environnement et de la possibilité d'accès ou non à l'intérieure du système
Attaques physiques :
•Découpage du circuit intégré
•Attaque chimique de la puce

1.6.4 Les Contres Mesures

Symptôme Gratuit (Asymptote Free) : ne fournir aucune information par fuite pour éviter les attaques Consciente de sécurité (Security Aware) être toujours conscient de son état et notamment de sa vulnérabilité dans le but d'être réactif.
Conscient d'activité (Activity Aware) : Analyser son environnement pour détecter une activité irrégulière en embarquant des capteurs et des systèmes de surveillance.
1 Système agile (Agile Système) : être flexible pour pouvoir rapidement répondre à une attaque ou alors anticiper l'attaque.

Mise à jour du logiciel et de matériel (Software or Hardware update) : Remettre à jour facilement les mécanismes de sécurité en fonction de l'évolution des attaques.
Résistance contre les erreurs : être résistant aux sabotages et attaques physiques.
Efficacité (Efficient) : Rester performant :ressources limités des systèmes embarqués plus contraintes fortes (consommation de puissance et d'énergie, débit, latence, surface)

1.7 Conclusion

La sécurité des systèmes et des données est un problème critique pour nos sociétés développées a cause des : Coûts de plus en plus important, le nombre d'attaques logicielles explose, de plus en plus de systèmes embarqués mobiles et communicants, menaces fortes des attaques logiciel. Et pour aborder la sécurité des systèmes et des données qui subissent des attaques sont soft ou hard impliquant des protections soft ou hard, il faut envisager des solutions à tous les niveaux, du système au transistor, définir des politiques de sécurité (sécurisé suffisante au bon moment) , définir des méthodes de vérification formelle assurant les performances des systèmes, ne rien négliger (aspect humain, légal, logicielle, matérielle, extérieur, intérieur).
Dans notre projet on vas s'intéresse aux méthode formelle précisément la méthode évent B et comment elle est efficace a la vérification de sécurité et le bon fonctionnement des systèmes embarques.

CHAPITRE 2

LES MÉTHODES FORMELLES

2.1 Introduction

Les principes qui définissent une méthode formelle appelée Event-B. Ses caractéristiques en font un outil fondamental, qui amène une certaine rigueur dans le processus de conception informatique. Cette rigueur, jointe à des concepts de production qui fait leur succès dans l'industrie. Il est à préciser que l'utilisation de cette méthode est imposée par les pouvoirs publics afin de vérifier l'élaboration des systèmes informatiques en relation avec des êtres humains, en particulier pour la conception des logiciels critiques, logiciels dont tout dysfonctionnement entraînerait des conséquences inacceptables.
Ce chapitre est organisé comme suite :la première partie illustre l'utilisation des méthodes formelles pour vérifier les systèmes critiques, la deuxième partie détaille la méthode Event-B. La troisième partie présente la plateforme Rodin et ses plugins et pour la dernière on a expliqué le prouver dans l'outil Rodin.

2.2 Utilisation des méthodes formelles pour la vérification

Les méthodes formelles sont de plus en plus imposées parmi les éléments majeurs des normes de qualité pour les «systèmes embarqués ». Elles ont pour objectif de démontrer par des techniques mathématiquement fondées et vérifiées ; par ordinateur, que ces systèmes répondent logiquement à leurs spécifications.
En effet, entre la fin des années 60 aux années 80, la méthode de vérification formelle la plus utilisée était basée sur la logique de Floyd-Hoare. Cette logique définit un ensemble

d'axiomes et de règles d'inférences pour l'ensemble des instructions de base d'un langage de programmation. En utilisant ces axiomes et ces règles d'inférences dans un système déductif formel, il est possible de produire manuellement des preuves de propriétés d'un programme. Cette approche nécessite la construction manuelle des preuves beaucoup plus longues et complexes que le programme initial.[9]

Ces méthodes formelles s'articulent autour d'un formalisme de spécification ayant des bases mathématiques fixant leur sémantique[10] . Différentes sortes de formalismes apparaissent dans l'utilisation des méthodes formelles :

•Le formalisme basé sur la logique.

•Le formalisme basé sur le comportement.

La base mathématique des formalismes permet aux outils de procéder à des analyses et de produire des résultats garantissant ou invalidant le respect de contraintes.

Les méthodes formelles sont utilisées selon le moment où l'on souhaite les intégrées, pour s'adaptée au contexte de vérification ou de validation.

Le choix d'une méthode formelle pour analyser un aspect du système concerné dépend du type de propriétés considéré. Dans notre cas on a choisi à travailler avec la méthode Event B. L'outil qu'on a choisi (Rodin), nous fournit des performances pour le type de propriété et supporte le type d'analyse que l'on souhaite effectue.

2.2.1 La méthode z

La notation Z est un langage de programmation formelle qui se distingue des méthodes semi-formelles et informelles. Autrement dit, Z offre un descriptif précis de l'usage d'un logiciel ou d'une application pour éviter les erreurs de développement. Ce qui est généralement le cas quand on emploie un langage standard. Elle est apparue au cours des années 1980, conçue par Jean-Raymond Abrial, informaticien français.

La notation Z possède une structure en schémas. Chacun d'entre eux correspond à un ensemble de données spécifiques. Celles-ci sont compartimentées pour mieux déterminer les fonctionnalités d'un système ou d'un logiciel. Là encore, cette division empêche les amalgames et d'éventuels problèmes de compatibilité. On appelle cela le principe de modularité. Z permet de prendre en charge deux principes essentiels de la programmation. L'aspect dynamique qui intègre les opérations, la corrélation entre entrées et sorties, ainsi que les changements d'état. Le second point réside dans l'aspect statique. Sur la base d'un schéma dédié, c'est elle qui détermine les états et les relations.[11]

2.2.2 Méthode de développement de vienne VDM

La méthode de développement de Vienne (VDM) est l'une des méthodes formelles orientées modèle les plus anciennes pour le développement de systèmes informatiques et de logiciels. Il consiste en un groupe de langages et d'outils mathématiquement fondés pour exprimer et analyser des modèles de système au cours des premières étapes de la conception, avant que des engagements de mise en œuvre coûteux ne soient pris. La construction et l'analyse du modèle aident à identifier les zones d'incomplétude ou d'ambiguïté dans les spécifications de système informel, et à fournir un certain niveau de confiance qu'une mise en œuvre valide aura des propriétés clés, en particulier celles de sûreté ou de sécurité. VDM a une solide expérience des applications industrielles, dans de nombreux cas par des praticiens qui ne sont pas des spécialistes du formalisme ou de la logique sous-jacents. L'expérience de la méthode suggère que les efforts consacrés à la modélisation et à l'analyse formelles peuvent être récupérés grâce à des coûts de reprise réduits résultant d'erreurs de conception.[12]
Une variante de la méthode VDM se nomme VDM++. Elle a été appliquée à divers domaines où la fiabilité et la sécurité des applications informatiques devaient être optimisées. Il s'agit principalement de l'aéronautique, le domaine spatial, les transports (exemple : trains automatiques, optimisation de la charge des navires). Peter Gorm Larsen, professeur à l'université d'Aarhus (Danemark) a publié plusieurs ouvrages et communications scientifiques relatives à ces applications. Plusieurs de ces expériences ont été cofinancées par la Commission des Communautés Européennes. L'un des projets (Afrodite) a fait l'objet de publications disponibles en ligne sur des sites hébergés par la Commission des Communautés Européennes. Il a consisté, entre autres, à appliquer les fondements de VDM++ à des expérimentations au CERN (Genève) , à l'évitement des tours par les avions (Centre technique de la navigation aérienne de Toulouse), à l'optimisation des charges de navires en .Grèce

2.2.3 La méthode B

Une B est une méthode de spécification formelle qui permet, grâce à un langage adéquat, d'exprimer très rigoureusement les propriétés exigées dans un cahier des charges. Il est alors possible de prouver de manière automatisée que ces propriétés sont non ambiguës, cohérentes et non contradictoires.
Cela nous permet de garantir ensuite par preuve mathématique que ces propriétés sont respectées au fur et à mesure des étapes de conception **Ainsi, cette méthode formelle et la preuve qui lui est associée permettent**

•D'obtenir des spécifications techniques et des cahiers des charges système clairs, structurés, cohérents et sans ambiguïté.

•De développer des logiciels garantis contractuellement sans défauts,Dans des domaines tels que le temps réel, les automatismes industriels, les protocoles de communication, les protocoles cryptographiques, l'informatique embarquée...

La **"méthode formelle B"** évoque traditionnellement l'ensemble comprenant : le langage B, le raffinement, la preuve et les outils associés.

Un développement B débute par l'écriture d'un modèle concret, reprenant tous les aspects du besoin. Les principales données sont manipulées par le système et sont décrites ainsi que les propriétés fondamentales de ces données. Des services assurent les transformations de ces données tout en préservant leurs propriétés. Le modèle B ainsi obtenu constitue une spécification de ce que devra réaliser le système. Le modèle B est ensuite transformé (''raffiné'' dans le vocabulaire B), jusqu'à'à obtenir une implantation logicielle complète du logiciel. Au final, nous aboutissons alors à un modèle concret, prouvé et sans défaut, transcodage dans le langage C ou Ada.La méthode formelle B est donc **"une démarche de construction prouvée (dite correcte), sur la base du langage B, du raffinement et de la preuve".**

Les objectifs de méthode B

Inventée par Jean-Raymond Abrial, la méthode formelle B est avant tout une nouvelle approche permettant de spécifier et concevoir des logiciels tout en s'assurant de leur sûreté ainsi que de leur fiabilité. Ainsi, l'ensemble des processus de spécification, de conception et de codage sont entièrement basés sur la réalisation d'un certain nombre de preuve mathématique. ce n'est qu'après avoir prouvé mathématiquement un modèle qu'il est alors jugé cohérent et sans défaut .[13]

2.2.4 La méthode event B

Event B est une méthode formelle pour la modélisation du système créé par Jean-Raymond Abrial. Il est basé sur une autre méthode formelle B appelé ''classique'' . Un modèle de l'événement B est caractérisé par ses invariants et des propriétés de sécurité. La méthode de l'événement B génère des obligations de preuve pour montrer que le modèle a un comportement correct, et que ses invariants et des propriétés de sécurité ne sont pas respectées.

Structure d'un modèle Event-B

Un modèle Event-B est décomposé en deux parties :le contexte qui contient la partie statique du modèle et la machine qui contient la partie dynamique du modèle. Cette séparation permet d'indiquer à une machine donnée les contextes qu'elle « voit ». Un modèle Event-B peut contenir des contextes seulement, des machines seulement ou les deux. Dans le premier cas, le modèle représente une structure mathématique pure. Le deuxième cas représente quant à elle un modèle non paramétré. Le dernier cas représente un modèle paramétré par les contextes. La figure suivante présente les deux composants avec leurs différentes clauses telles qu'elles sont déclarées dans la plateforme RODIN.[14]

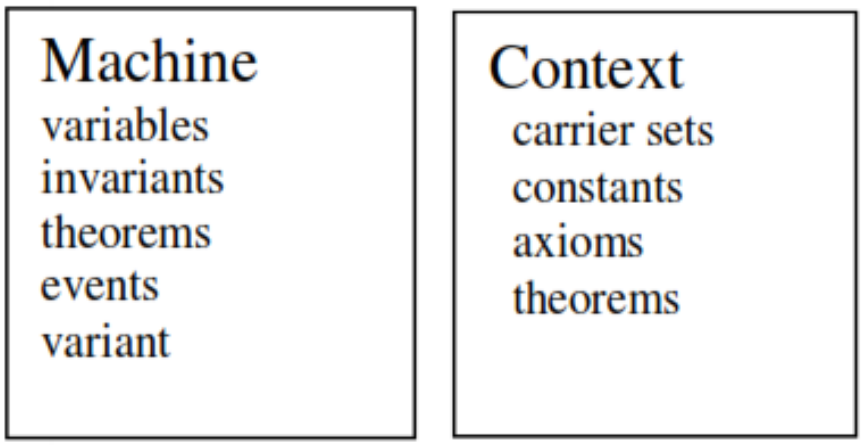


FIGURE 2.1 – Machine et Contexte.

Les modèles peuvent voir des contextes (clause SEES), pour accéder à la théorie sur laquelle un modèle est construit. Un modèle peut être raffiné par un autre modèle. Une différence importante vient du fait qu'un événement peut être raffiné par plusieurs événements (découpage des événements), et inversement, un événement peut en raffiner plusieurs (fusions des événements). Une fusion de deux événements E_1 et E_2 est équivalente au choix borné des deux événements E_1— E_2. Un contexte peut être étendu par un autre contexte. Le raffinement de contexte est simplement une extension des déclarations du premier contexte. Un modèle raffiné M_2 peut voir le raffinement c_2 d'un contextec_1 vu par l'abstraction M_1 du modèle raffiné (voir figure). Les relations de raffinement entre composants sont transitives (et réflexives). Les ensembles et les constantes déclarées dans un contexte sont, des paramètres du modèle. Il est prévu d'avoir une opération d'distanciation des modèles, c'est-à-dire remplacer un contexte abstrait par des ensembles et des constantes d'une théorie plus spécifique.

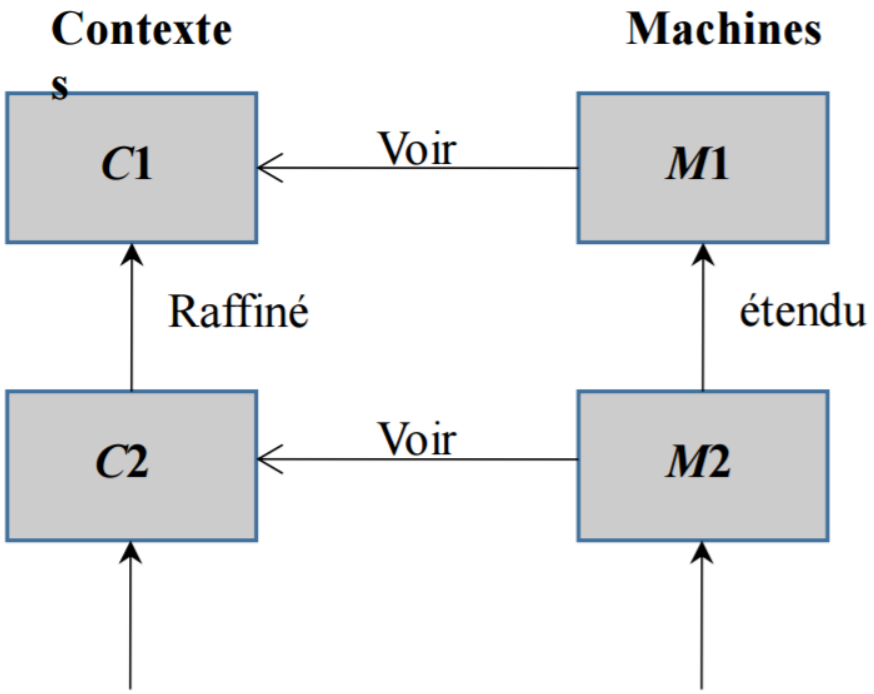


FIGURE 2.2 – Relation entre composants d'un modèle event-B

Modélisation en utilisant la méthodologie de l'Event B

Modèles événement B sont composés de deux éléments : un «contexte» et une «machine».

Le Contexte :

On peut considérer le contexte comme la partie statique du modèle contenant plusieurs

```
CONTEXT ctxtN
   EXTENDS ctxt1, ctxt2, …
   SETS S1, S2, …
   CONSTANTS C1, C2, …
   AXIOMS A1, A2, …
   PROPERTIES P1, P2, …
END
```

FIGURE 2.3 – les clauses possédant un contexte

clauses. La figure.7 décrit les clauses du contexte. Nous distinguons :

•La clause CONTEXT représente le nom du composant qui devrait être unique dans un modèle.

•La clause EXTENDS déclare la liste des contextes qu'étend le contexte décrit (ctxt1, ctxt2...).

• Un contexte peut étendre un autre contexte en rajoutant de nouvelles constantes et de nouvelles propriétés

•La clause SETS définit les ensembles porteurs du modèle (S_1, S_2...). Ces ensembles non vides servent à typer le reste des entités du modèle.

•La clause CONSTANTS contient la liste des constantes utilisées par le modèle (C_1, C_2...).

• La clause AXIOMS définit les propriétés liées aux constantes et notamment leurs types (A_1, A_2...).

•La clause THEOREMS exprime des propriétés qui peuvent être déduites à partir des propriétés présentées dans la clause AXIOMS (P_1, P_2...)

La machine abstraite :

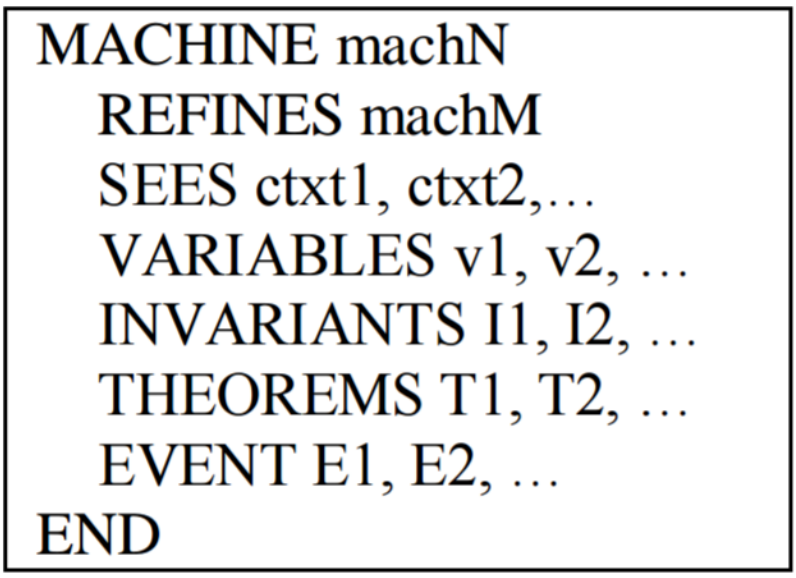


FIGURE 2.4 – les clauses possédant une machine

La machine est la partie dynamique du modèle et elle est constituée de plusieurs clauses :

•La clause MACHINE représente le nom du composant (mahN) qui devrait être unique dans un modèle.

•La clause REFINES déclare le nom de la machine raffinée (machM) par la machine décrite.

•a clause SEES spécifie la liste des contextes « vus » par la machine (ctxt1, ctxt. . .). Dans ce cas, la machine peut utiliser les constantes et les propriétés figurant dans les contextes.

•La clause VARIABLES contient la liste des variables du modèle (V_1, V_2...).

•La clause INVARIANTS définit les propriétés d'invariance du modèle telles que des infor-

mations sur les types des variables et des propriétés de sûreté (I_1, I_2...).

•La clause THEOREMS exprime des propriétés qui peuvent être déduites des propriétés d'invariance de la machine et des propriétés présentes dans les clauses AXIOMS et THEOREMS du contexte vu (T_2, T_2...). En outre, cette clause peut contenir des propriétés que l'on souhaite prouver afin de les employer dans la preuve des invariants du modèle.

•La clause VARIANT définit l'expression du variant du modèle.

•La clause EVENTS contient la liste des événements qui opèrent une ou plusieurs substitutions sur la valeur des variables (E_1,E_2...). Parmi ces événements, l'événement INITIALISATION donne une valeur initiale aux variables.

Les événements

Un événement Event-B correspond à un changement d'état dénotant une transition dans le système modélisé. Un événement est essentiellement composé d'une garde (la clause WHEN) qui définit les conditions nécessaires au déclenchement de l'évènement et d'une action (la clause THEN) qui définit l'évolution des variables d'états. Notons que plusieurs gardes d'événements peuvent être vraies en même temps. Néanmoins, un seul événement peut se déclencher et le choix de cet événement est non déterministe. Un évènement peut posséder des paramètres (des variables locales) définis dans la clause ANY.[15]

Notion du raffinement en Event B

Le raffinement en Event-B consiste à développer le système de manière incrément ale en partant d'un modèle abstrait qui constitue une spécification du système. A chaque étape de raffinement, des détails du système sont rajoutés graduellement dans un modèle concret qui doit préserver la fonctionnalité et les propriétés des modèles plus abstraits. Ces détails rajoutés apparaissent dans l'état du système en ajoutant des variables, et dans le comportement en détaillant les événements de l'abstraction ou en ajoutant de nouveaux événements. Notons que les nouveaux événements raffinent un événement particulier de l'abstraction qui est l'évènement vide (appelé skip). Des obligations de preuve sont générées à chaque étape de raffinement afin d'assurer la correction du raffinement .[15]

2.2.5 Les Principal Différent Entre Event B et B Classique

•a notion d'opération est remplacée par la notion d'événement. Un événement a une garde (sans pré condition), et peut être déclenché si la garde est vraie.

•le remplacement des notions de structuration de machines (USES, SEES, INCLUDES, IMPORTS, etc.) par la seule notion de contexte, qui regroupe les constantes et les ensembles de bases, ainsi que des axiomes associés.
•la disparition de certains types de données et leurs opérateurs associés, notamment les séquences (seq), les structures (struct) et les arbres (btree).
•la disparition de certaines substitutions. Chaque événement en Event-B a en fait une forme très simple. La forme la plus compliquée peut-être exprimée par un seul ANY contenant des assignations (déterministes et non déterministes) parallèles. Les variables du ANY sont les « paramètres » de l'événement. Il existe deux formes simplifiées, une pour des événements sans paramètre et une pour des événements sans paramètre et sans garde.
une notion adaptée du raffinement, permettant l'introduction de nouveaux événements pour séparer un événement en plusieurs événements dans la machine raffinée, et inversement, de fusionner plusieurs événements.

2.3 La Plateforme Rodin

2.3.1 Description de la plateforme

Un grand atout d'Event-B est la plate-forme RODIN[16] , qui est basée sur Eclipse. RODIN stocke les modèles dans une base de données et fournit des nouveaux prouveurs puissants, qui peuvent être manipulés à l'aide d'une interface graphique. Un autre aspect intéressant est la possibilité d'étendre RODIN à l'aide de «plug-ins» permettant notamment de :
•Prouver les modèles en utilisant les prouveurs de l'Atelier B, aussi le propre prouveur de RODIN, appelé NewPP.
•Vérifier les modèles par les techniques du model checking en utilisant ProB.
•Animer les modèles en utilisant AnimB.
L'intérêt de RODIN est que c'est une plateforme extensible. D'ailleurs, elle est maintenant en cours d'amélioration et d'extension par d'autres «plug-ins». [16]

2.4 Le prouver dans l'outil Rodin

2.4.1 Définition

La preuve sert à trouver des erreurs,ce n'est pas programmer,la preuve peut être partiellement automatique,donc un prouver ne sait pas démontrer du'une obligation est fausse,il

a une base de règles qui est l'ensemble des connaissance mathématiques à utiliser,donc sa preuve désigne la base et les mécanisme (regroupé en niveaux :les force Rapide 0,1,2,3) les commandes de preuve contrôlent un prouve en automatique comme en interactif c à d de prouver une obligation de preuve par une démonstration automatique ou interactive pour qu'un peut dire que cette dernière à l'état :" Proved " ou " Unproved " Dans les cas où on veut orienter la preuve sans ajout de connaissance est une preuve non validée et cela sert que la démonstration manuelle peut n'employer que des règles validées et pas des règles manuelles.

Afin de garantir la correction de notre modèle ,il est indisponible de le prouver pour parvenir le générateur d'obligations de preuve génère automatiquement des obligations de preuve une obligation de preuve définit ce que doit être prouver pour un modèle.

2.4.2 Les obligations de preuves

Définition

Une obligation de preuve est une formule mathématique à démontrer afin d'assurer qu'un composant B est correct.

La théorie B indique quelles sont les obligations de preuve à d´démontrer pour assurer la correction d'un composant B donné. Dans cette optique, les obligations de preuve sont une aide au processus de Vérification. Les règles des obligations de preuve sont au nombre de onze, et pour chacune le générateur d'obligations de preuve génère une forme spécifique .

Les obligations de preuve produites par le générateur d'obligations de preuve qui transforme les formules théoriques en formules uniformes (plus nombreuses et plus simples) propres à être utilisées efficacement par le prouveur de l'atelier B.

Forme générale des obligations de preuve

Les obligations de preuve B sont auto-suffisantes, c'est-à-dire qu'aucune information implicite ne doit être utilisée pour leur démonstration. Toutes les obligations de preuve ont la structure suivante : $H \Rightarrow P$Où P et H sont des prédicats. Cette formule signifie qu'il faut démontrer le but P sous l'hypothèse H, H étant généralement une conjonction de prédicats. En B, le prédicat P et certaines hypothèses de H sont construits par application d'une (ou plusieurs) substitution(s) à un prédicat. B étant un langage mathématique, les substitutions et les prédicats considères sont directement extraits de la source B.

Les règles des obligations de preuve sont au nombre de onze, et pour chacune le générateur

d'obligations de preuve génère une forme spécifique .[17]

2.4.3 Génération d'obligations de preuve au sein de la plateforme Rodin

Il est extrêmement important de savoir comment Rodin construit et essaye de résoudre les obligations de preuve associées à des machines abstraites et leurs raffinements. Cela nous permet de :

•Concevoir des machines et des invariants correctes.

•Détecter les erreurs de conception.

•En cas d'une preuve non-automatique, il permet d'interagir avec l'outil.Les Obligations de Preuve sont générées automatiquement par un outil de la plate-forme Rodin appelé le générateur d'obligations de preuve.

•Cet outil accomplit une vérification statique y compris une analyse lexicale, une analyse syntaxique et une vérification de type des contextes et des machines.

•Il décide alors de ce qui doit être prouvé.

•Les résultats sont des séquents différents, qui sont transmises aux prouveurs effectuant des preuves automatiques ou interactives. Ce processus est illustré dans la figure 3

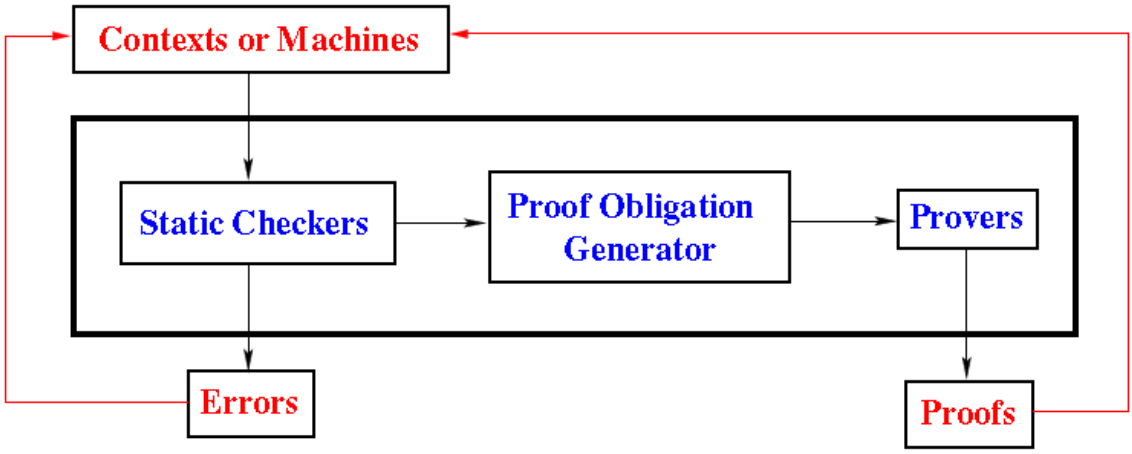


FIGURE 2.5 – Outils du noyau de la plateforme Rodin
[18]

En Event-B un séquent est de la forme H— G où H est un ensemble fini de prédicats appelés hypothèses et G est un prédicat unique appelé objectif (but)[19]. Un séquent signifie essentiellement que l'objectif devrait être une conséquence logique des hypothèses. Preuves de séquents sont effectuées en utilisant des règles d'inférence. Une règle d'inférence signifie : si nous pouvons prouver tous les séquents dans A (antécédent), alors C (conséquent) a également été prouvé. La preuve d'un séquent peut donc être considérée comme un arbre

fini. Les enfants d'un nœud sont les séquents dans l'antécédent de sa règle. Pour générer effectivement une obligation de preuve, nous avons besoin de trouver un arbre de preuve dont le nœud racine est étiqueté avec l'obligation de preuve [20].

2.5 Conclusion

La méthode B est considérée comme étant une des méthodes formelles les mieux acceptées dans le cadre industriel du développement des logiciels critiques. En effet, B n'apporte pas que des concepts, mais également :

•Une méthologie (au sens du Génie Logiciel) puisqu'elle s'inscrit dans le développement complet d'un projet, introduisant une méthodologie propre de conception : machines abstraites, raffinements et implémentation, le tout dans une arborescence structurée, avec des règles de <visibilité>, etc ...

•De spécifications formelles apporte une rigueur de traitement dans le processus de développement, sans compter qu'elle lève toute ambiguïté possible des spécifications non-formelles (écrites en langage naturel) du cahier des charges, donc toutes les équipes partent du même constat et ont des bases compréhensibles par tous.

•une modélisation en <machine>, bien que formelle, se révèle ¡¡graphiquement¿¿ très proche d'un programme informatique élémentaire. Cette représentation s'avère être très <intuitive> , ce qui facilite de le travail des informaticiens n'ayant pas eu la formation appropriée pour manipuler pareille technique.

Globalement, il est raisonnable de penser que B allie la puissance du formel à la flexibilité de la modularité informatique. En outre, cette méthode est une évolution et/ou une alternative à la puissance de Z, sans son formalisme extrême dans la forme (aspect mathématique très, voire trop prononcé), ce qui peut en rebuter certains aux premiers abords. Cette méthode était utile pour nous en parallèle avec la simulation des systèmes afin d'améliorer la puissance du système.

CHAPITRE 3

ARCHITECTURE AUTO-ORGANISÉ MULTI-NŒUD À LA BASE D'UN RÉSEAU SANS FIL ORIENTÉE NOC

3.1 Introduction

Dans ce chapitre, nous montrons comment intégrer les structures de NoC dans un système auto-organisé basé sur le réseau en utilisant une approche conceptuelle décentralisée basée sur le contrôle. La méthodologie adoptée est unique en ce sens qu'elle utilise l'auto-organisation comme outil pour assurer le fonctionnement sûr du réseau. Pour détecter tout défaut dans les réseaux de puces des nœuds auto-organisés qui composent le système réseau, nous appliquons plusieurs codes de correction d'erreurs...[21]
Ce chapitre décrit la méthode de test externalisée basée sur le système auto-organisé proposé, qui permet de garantir le SdF des structures de communication sur une puce tout en mettant en œuvre les interactions mutuelles de toutes les entités qui composent le réseau d'auto-organisé et nœuds de service.

3.2 Nœuds Reconfigurables

Certains FPGA ont la capacité de se reconfigurer de façon dynamique et partielle, ce qui permet une répartition intelligente de la charge de travail à l'intérieur d'un SoC. En effet, en divisant la zone en différents zones.[21]

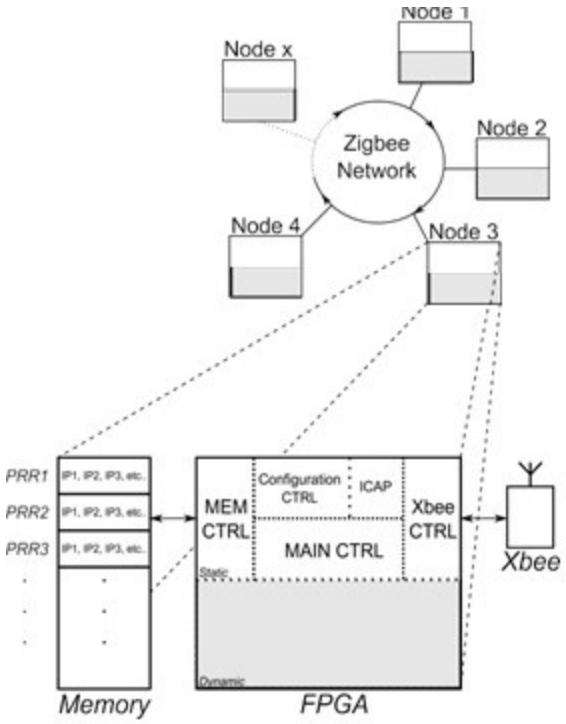


FIGURE 3.1 – Système auto-organisé multi-nœuds en réseau appliqué aux communications sans fil.

[21]

A l'aide des fichiers de configuration connexes, il est possible de construire un certain nombre d'adresses IP séquentielle ment dans un FPGA, conformément à la conception illustrée à la figure 3.1 (bitstreams). De plus, si la logique liée au FPGA est localement défectueuse, elle peut être réaffectée à un autre emplacement.

3.2.1 Accessibilité aux fichiers de re configuration

Le système auto-organisé offre de nombreuses techniques pour récupérer et stocker ces fichiers afin de minimiser la quantité de bitstream qui peut être important. Chaque nœud comprend une mémoire externe qui permet un accès et un stockage rapides des fichiers au début. L'utilisation de cette mémoire permet une re configuration rapide des bitstreams disponibles pour un nœud spécifique

Si un flux binaire n'est pas disponible localement, le nœud affecté peut communiquer avec le reste du réseau via l'échange de données. Si un flux binaire n'est pas trouvé localement dans la mémoire, il peut être récupéré et implémenté, ou récupéré, enregistré (si l'espace de stockage est disponible), et implémenté, ou récupéré pour être stocké en le remplaçant en mémoire par un autre flux binaire.Si un flux binaire n'est pas trouvé localement dans la mémoire, il peut être récupéré et implémenté, ou récupéré, enregistré (si l'espace de sto-

ckage est disponible), et implémenté, ou récupéré pour être stocké en remplaçant un autre bistream en mémoire (si saturé) et implémenté. Un nœud nécessitant une re configuration partielle peut bénéficier de cette flexibilité dans la transmission des fichiers de re configuration sur l'ensemble du réseau.[21]

3.3 Gestion autonome des tâches

Pour commencer, il est essentiel de se rappeler que tous les nœuds du réseau sont identiques et intègrent dans un FPGA une structure architecturale qui permet une auto-adaptation et une forte interaction entre les composants du système, comme le montre la figure 1.1. En conséquence, nous établissons un réseau auto-organisé en combinant l'auto-organisation du réseau avec la reconfiguration partielle du FPGA.Ainsi, en combinant l'auto-organisation du réseau avec la reconfiguration partielle du FPGA, nous développons un système d'auto-organisation capable de répartir les charges de travail sur tous les nœuds du réseau (soit à l'intérieur du nœud lui-même, soit délocalisé sur tous les nœuds du réseau).
Un travail ne peut pas être transféré dans une zone différente du même FPGA en fonction du type de défaillance. Dans cette situation, le nœud non fonctionnel demande que les tâches dans la région non fonctionnelle soient réparties entre les autres nœuds. Pour ce faire, les fichiers de configuration des tâches incriminées et les données qui les accompagnent sont migrés vers un autre nœud. Toutefois, avant d'accepter un dossier, celui-ci doit suivre des lignes directrices précises.

- Ne pas être considéré comme un nœud fautif.
- Disposer de ressources logiques suffisantes afin d'accueillir l'implémentation de la nouvelle tâche.
- Avoir l'accessibilité au fichier de reconfiguration ou être capable de se le procurer à travers un autre nœud du réseau.
- Ne pas être occupé par des tâches prioritaires.

En d'autres termes, le nœud ne peut être tenu responsable de sa propre défaillance. Pour la configuration IP, il doit disposer de ressources matérielles suffisantes (zone logique disponible). Il doit alors avoir accès au fichier de configuration IP pour l'exécuter, ou pouvoir le recevoir ou l'obtenir à partir de la mémoire de stockage d'un autre nœud réseau si nécessaire. La règle finale est que ce nœud ne doit pas déjà exister.Le critère final est que ce nœud ne doit pas être utilisé pour des opérations hautement prioritaires comme tester un autre nœud. Une fois toutes ces conditions remplies, le nouveau nœud IP assigné utilise

les fichiers de reconfiguration pour se reconfigurer et remplace le nœud défaillant dans le réseau. D'un point de vue global, le système continue de fonctionner parce qu'aucune des thérapies n'est interrompue indéfiniment.[21]

3.3.1 Communications inter-nœuds

Le système auto-organisé établi est également basé sur la façon dont les nœuds communiquent entre eux pendant l'opération, ainsi que lorsque des problèmes sont identifiés. Pour ce faire, nous avons développé un protocole de communication inter-nœuds adapté à la structure du réseau de communication Zigbee. Lorsqu'un nœud échoue et que le nœud en question est incapable de réorganiser ses tâches pour s'adapter à l'échec, l'adresse IP qui lui est attribuée dans son PRR est déplacée vers un autre nœud du réseau. L'idée d'externaliser des tâches et de créer une IP de test à distance dans le cadre d'un test à distance d'un nœud défectueux présumé est illustrée à la figure 3.4. Pour ce faire, vous aurez besoin d'un cadre contenant.

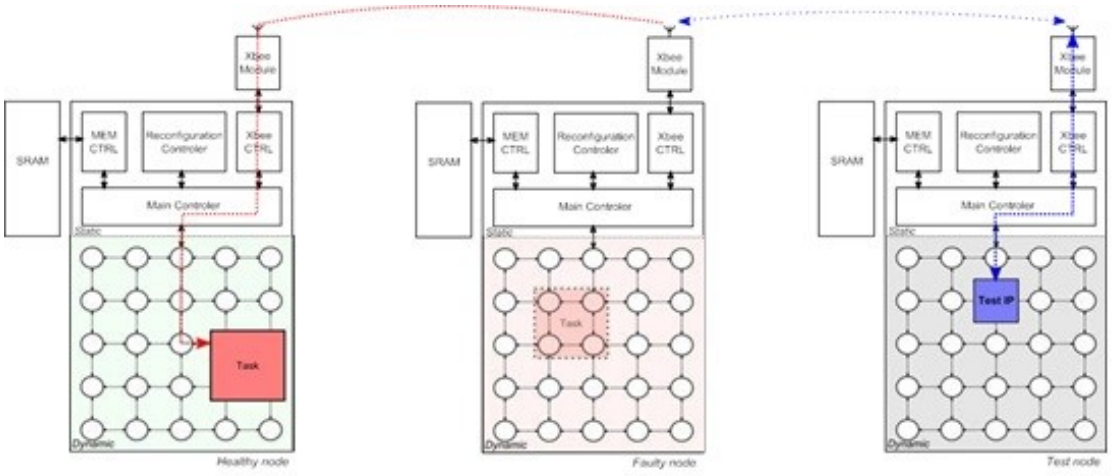


FIGURE 3.2 – Illustration du mécanisme de délocalisation de tâches à travers le réseau de communication Zigbee dans le cadre d'un test à distance.
[21]

identification utilisée pour avertir d'autres nœuds réseau d'une défaillance et de la demande de re-localisation des tâches Cette base de sondage fournit également les données dont les autres nœuds ont besoin pour évaluer leur capacité à aider. Parmi ces données, nous pouvons localiser l'identifiant IP, qui nous permet de déterminer la surface logique nécessaire à la mise en œuvre d'une certaine technologie FPGA. En effet, comme indiqué dans la section précédente, le nœud d'hébergement doit respecter un certain nombre de principes pour

que le système soit main-tenable. Lorsqu'un nœud reçoit une demande de réinstallation, il détermine s'il peut ou non répondre à la demande. Un cadre de retour est envoyé une fois que toutes les conditions ont été remplies, non seulement pour confirmer la demande, mais aussi pour identifier le nœud qui a la tâche. S'il y a plusieurs nœuds,Si plus d'un nœud peut terminer le travail, la première image reçue par le nœud défaillant est utilisée pour choisir le nouveau nœud qui hébergera le travail déplacé.Comme les transmissions sont diffusées, la première image reçue correspond au nœud le plus proche ou au nœud le moins chargé, satisfaisant le protocole dans les deux cas. Lorsque le nœud de re-localisation (nœud "helper") est trouvé, le fichier de configuration IP (si disponible) et les données de la tâche lui sont envoyés. Si le nœud défaillant ne peut pas transférer le fichier de configuration, une nouvelle communication est établie entre le nœud d'assistance et le reste du réseau afin d'identifier un nœud réseau avec bitstream. Une fois les données envoyées, le nœud helper envoie un accusé de réception.Il est essentiel d'établir un réseau de communication sécurisé basé sur des puces afin d'améliorer considérablement la capacité organisationnelle des tâches, en particulier au sein d'un nœud.

BEGIN	ID_d	ID_s	Type Trame	Data	END

TABLE 3.1 – Structuration en champs d'une trame de communication directe entre deux nœuds du réseau

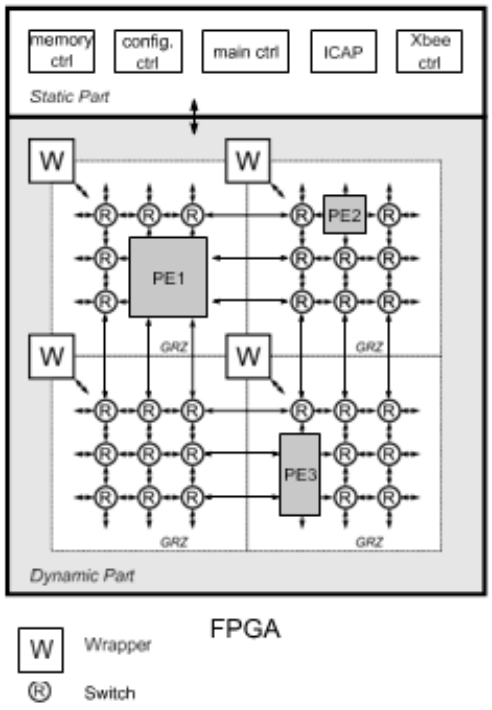


FIGURE 3.3 – Configuration d'un nœud auto-organisé.

3.3.2 Structure des trames

La composition d'un cadre de communication directement entre deux nœuds du système auto-organisé est présentée au tableau 3.1. Les champs BEGIN et END font partie des champs qui composent un cadre. Il comporte également des champs qui indiquent le type d'information Type Trame contenu dans le champ Données, ainsi que les identités des nœuds destinataires des IDd et des IDs sources. Lorsque le système utilise ce type de trame,Lorsque le système détecte une réorganisation de la structure re configurable des nœuds en cas de défaillances détectées ou en réaction à l'évolution externe du système, il utilise ce type de trame.

3.4 Fonctionnement global du test

Voici comment fonctionne notre système de test. La procédure commence par la découverte d'une erreur dans le NoC de RKT, qui sont les composants du. Les FPGA ont été utilisées pour exécuter les noeuds MPSoC, avec des tests en ligne basés sur les CEE.

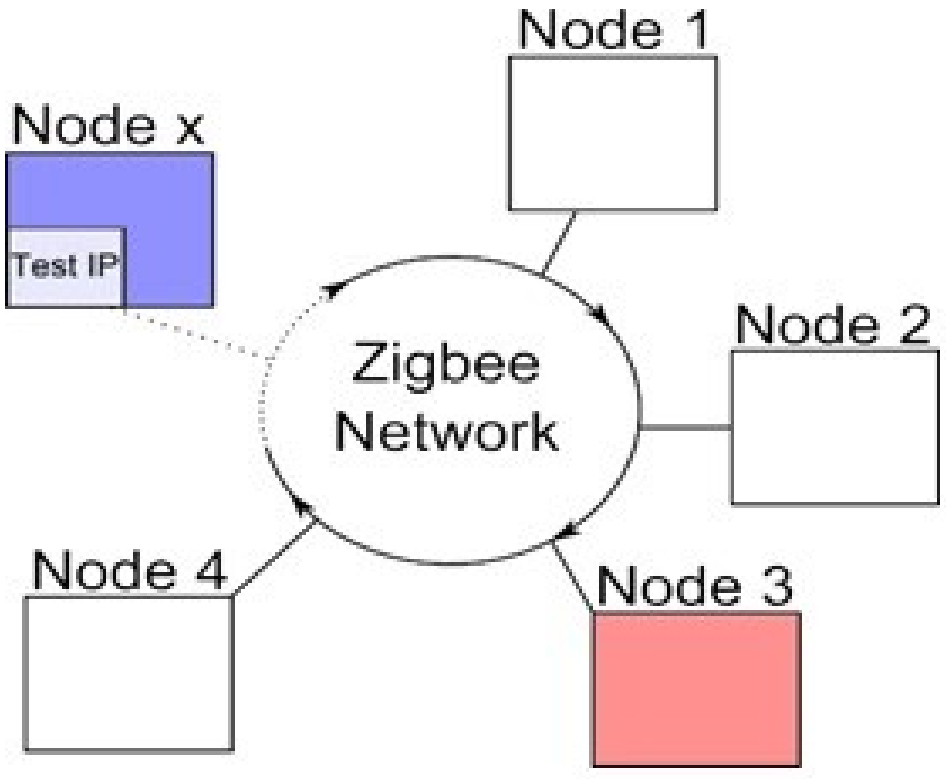


Figure 3.4 – Réseau auto-organisé et auto-reconfigurable au protocole Zigbee.

La structure interne du FPGA dans les différents nœuds du système auto-organisé est présentée à la figure 3.5. Les routeurs Global Safe Zones impliqués entrent en mode test

une fois que le système détecte un ou plusieurs défauts. Les autres nœuds non-défectifs reçoivent alors une requête. Cet appel vous permet de savoir quels nœuds sont capables d'implémenter et de réaliser un test IP. Le FPGA connecté au noeud de test est reconfiguré dynamiquement avec le fichier de configuration IP de test une fois le nœud de test détecté. En déplaçant un test IP vers un nœud éloigné dans le réseau, la Figure 3.6 illustre le principe du test hors ligne suggéré. Ces vecteurs sont des séquences de bits qui provoquent une réponse spécifique d'une architecture connue. En conséquence, le nœud de test commence par mettre ses propres routeurs au test afin de recevoir une réponse à un vecteur produit. L'utilisation des divers vecteurs par le mécanisme d'essai au sein d'un nœud est résumée à la figure 3.7. En raison de la santé du nœud, cette réponse est utilisée comme point de repère. Le test et les vecteurs de réponse sont deux choses distinctes Le nœud à tester reçoit des vecteurs de test et de réponse. Le vecteur de test est ensuite injecté dans le NoC du nœud défaillant à l'aide d'un bloc spécial appelé Wrapper, qui crée un lien entre les éléments statiques et dynamiques du nœud à tester. Le vecteur de test est envoyé à chaque routeur qui est en cours de vérification pour ZGS erronée. Le Wrapper garde une trace de la référence associée.Une fois le vecteur injecté et les réponses collectées, les résultats sont comparés localement à la référence pour identifier à la fois les routeurs défectueux et les éléments défectueux (tampons, logique de routage, etc.) qui sont à l'origine des erreurs. La réponse est renvoyée au test IP pour analyse en fonction de la quantité d'erreurs ou de l'emplacement des problèmes. En conséquence, le Wrapper détecte.En conséquence, le Wrapper identifie les parties problématiques du routeur basées sur des emplacements de bits incorrects dans la réponse vectorielle au vecteur de test. En conséquence, une décision est prise concernant le routeur défectueux (re configuration partielle ou globale ou désactivation du routeur).[21]

Vers IP test [$\vec{V}$] Vecteur test

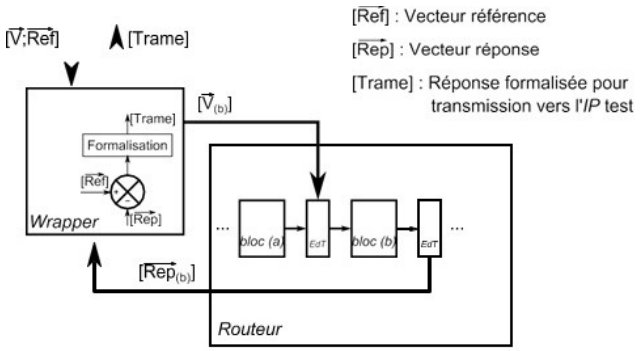


Figure 3.5 – Mécanisme de localisation d'erreur au sein d'un nœud. [21]

3.4.1 Principe de fonctionnement des blocs Éléments de Test

Les ETs ont été créés pour être en mesure de découvrir une faille dans l'architecture. Ces ETs sont des registres "améliorés" qui sont intercalés entre deux composantes architecturales. Ils sont responsables de recevoir et d'injecter certaines données dans l'un des composants du routeur. Ces données sont appliquées aux entrées du bloc à tester à l'aide d'un vecteur de test.

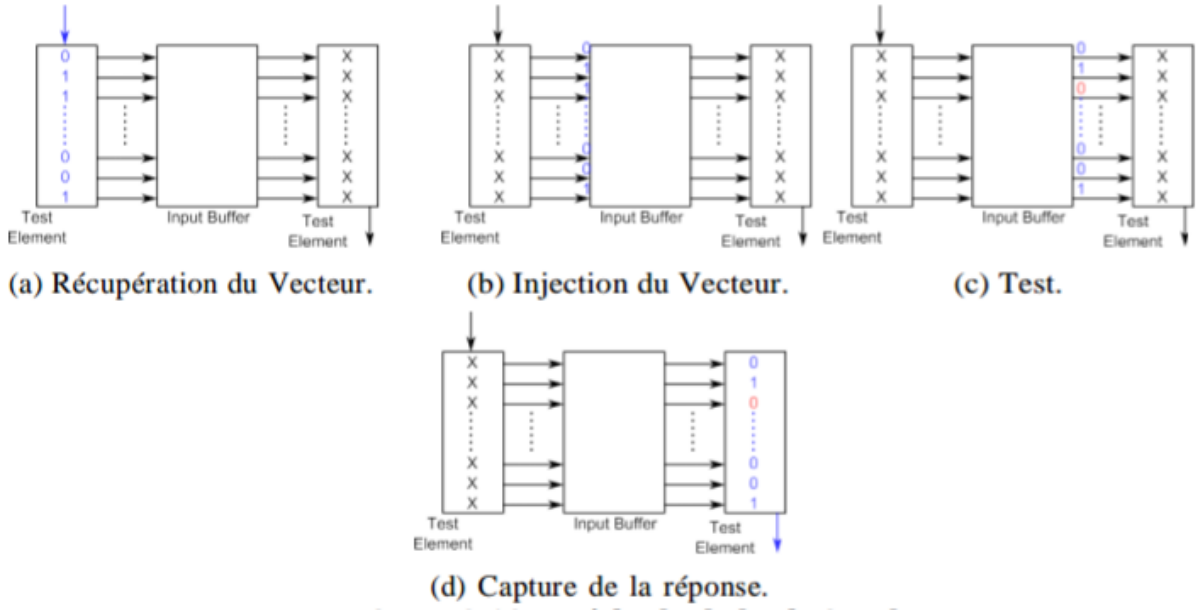


FIGURE 3.6 – Méthode de la chaine de test.
[21]

La figure 3.13 montre les procédures d'utilisation des ET. Les éléments d'essai fonctionnent en cinq étapes. Le premier est le remplissage des ET (voir figure 3.13a), qui est contrôlé par un contrôle à 2 bits et effectué sur une entrée série. La deuxième phase consiste à fournir les données pertinentes à l'architecture à tester lorsque les ET l'ont récupérée (voir la figure 3.13b). Ces informations ont été créées spécialement pour vous.Ces données, qui incluent des données, des signaux de commande, des remerciements, etc., sont générées explicitement pour atteindre un objectif spécifique.En mode normal, la troisième étape est un état d'attente pour le traitement architectural. Lorsque le traitement des données de l'architecture est terminé et que les données sont présentes sur ses sorties, elles sont saisies et conservées dans les ET suivants (voir figure 3.13c).). Le vecteur de réponse au vecteur d'essai est constitué des résultats ET. Parce que le même processus est effectué dans la même architecture dans un nœud sain et problématique, la différence entre les deux réponses peut être analysée pour identifier les défauts. L'IP de test peut alors déduire, pos, en utilisant les positions des bits erronés dans le vecteur de réponse.L'IP de test peut alors identifier les as-

pects de l'architecture causant des défauts basés sur l'emplacement des bits incorrects dans le vecteur de réponse, peut-être après avoir utilisé différents vecteurs de test. Une sortie série retourne les vecteurs de réponse aux vecteurs de test au bloc Wrapper dans la dernière phase.

3.5 Distribution des vecteurs de tests

En termes de temps d'exécution, la distribution des vecteurs de test est une composante essentielle du mécanisme. En fait, le processus comporte deux étapes qui exigent plus ou moins de temps selon la façon dont elles sont mises en œuvre. Le premier est la distribution des vecteurs de test pour chacun des types de routeurs d'un ZGS.. Dans la plupart des cas, un vecteur est injecté en série dans tous les composants d'une conception testée. Quand il y a beaucoup d'éléments à tester, le temps qu'il faut pour remplir tous les éléments de test peut être assez considérable.Pour remédier à cet inconvénient, nous distribuons le vecteur avec une combinaison de transmissions série et parallèle, comme détaillé dans le paragraphe suivant. Le retour des réponses vectorielles aux vecteurs de test est la deuxième étape et plus longue.. Plus de détails sur le formatage des réponses sont donnés dans la section 3.7.2.
L'injection de données de chaîne d'acquisition est une méthode lente dans laquelle le temps nécessaire pour charger les données est déterminé par la taille du vecteur et donc le nombre d'éléments que la chaîne doit traverser. Par conséquent, plus le vecteur est long, plus il se charge rapidement. Pour réduire le temps nécessaire à la charge des données à travers la chaîne de test, nous injectons le vecteur en partie en parallèle. Pour chaque élément de test, une entrée séparée de la chaîne de test est configurée. Une caractéristique unique d'un test de routeur est la valeur constante des données passant par la conception. Seuls quelques signaux de commande diffèrent d'un bloc à l'autre.En conséquence, nous injectons simultanément des données vectorielles de test dans tous les éléments de test de tous les routeurs ZGS. Ensuite, pour tous les routeurs ZGS, les signaux annexés sont injectés séquentielle ment dans chaque élément de test placé entre des blocs identiques. . Par exemple, les annexes vectorielles pour les éléments d'essai placés avant les tampons d'entrée (voir la figure 3.11) sont injectées simultanément dans toutes les ET dans la même position dans tous les routeurs ZGS soumis à l'essai.
Cette adaptation unique de la méthode de la chaîne de test est rendue possible par un multiplexeur général installé dans chaque routeur. Le bloc Wrapper sélectionne tous les ET pour tous les routeurs en même temps et les charge un par un avec les données des signaux an-

nexés. Cette solution est avantageuse en termes de temps d'exécution pour deux raisons. D'abord et avant tout, cette technique.Pour commencer, cette méthode de remplissage des ET est plus rapide que l'injection de vecteurs de test dans une seule chaîne. Deuxièmement, comme nous utilisons des données partagées dans toutes les ET, nous pouvons réduire le temps nécessaire à la transmission des données de test IP au bloc Wrapper.[21]

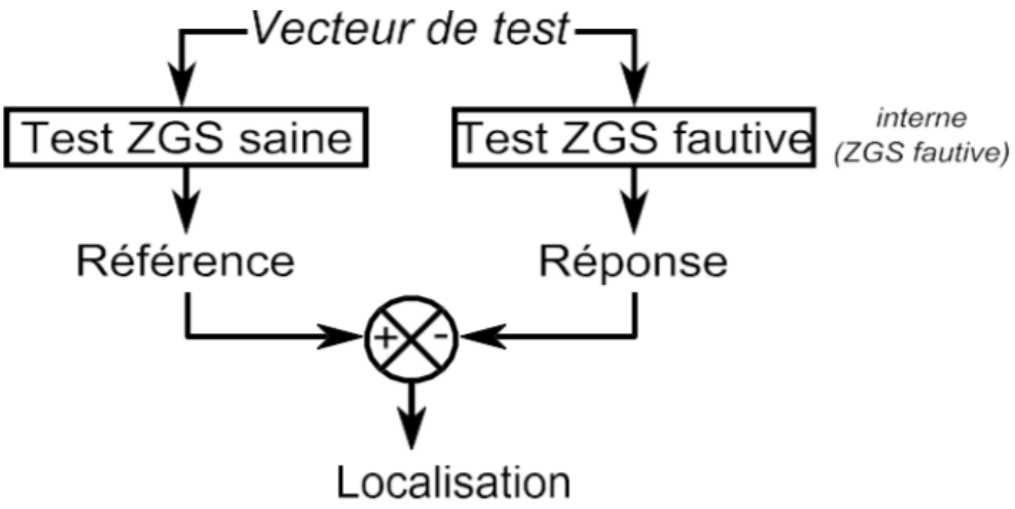


FIGURE 3.7 – Principe de fonctionnement de la localisation d'une erreur.

3.6 Traitement des réponses gérées par le mécanisme SDF proposé

3.6.1 Mise en forme des résultats

Les résultats des tests locaux des routeurs ZGS sont contenus dans l'EdT. Comme pour la charge de ces appareils, les données sont récupérées de manière séquentielle.

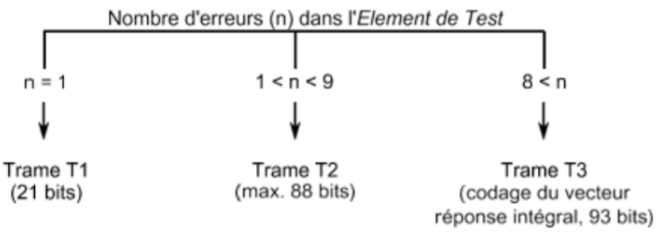


FIGURE 3.8 – Codage trame des vecteurs réponses d'un EdT en fonction du nombre d'erreurs détectées.

[21]

simultanément Les réponses sont ensuite envoyées au bloc Wrapper par routeur. La série de sortie de chaque EdT est utilisée par chaque routeur pour créer un bus de données qui est envoyé au bloc Wrapper. Ce bus collecte les réponses au bloc Wrapper après avoir traversé un multiplexeur pour sélectionner la route. Une fois les données disponibles, la réponse de chaque EdT est analysée.Une fois les données disponibles, la réponse de chaque EdT est analysée en la comparant à la référence. Par la suite, la comparaison génère trois éléments d'information :

•Les vecteurs qui répondent aux vecteurs de test.

•L'emplacement des bits erronés dans les vecteurs qui répondent.

•Le nombre de réponses non prédictives

Lorsqu'une réponse EdT est comparée et que trois éléments d'information sont générés, le bloc Wrapper formate l'information en fonction des résultats. En fait, afin de réduire le temps nécessaire pour envoyer les résultats au test IP, les réponses aux vecteurs de test sont envoyées sous différentes formes de trames. La figure 3.15 illustre le codage des réponses vectorielles d'un élément d'essai.La figure 3.15 illustre le codage des réponses vectorielles d'un élément d'essai en fonction du nombre d'erreurs détectées. En conséquence, en fonction du nombre d'erreurs commises par un Élément of Test, quatre différents types de trames de réponse vectorielle peuvent être envoyés sur le réseau sans fil. Plusieurs champs de ces trames sont identiques.Lorsqu'il y a une seule erreur dans la réponse vectorielle d'un élément de test, un formulaire T1 est codé comme indiqué dans le tableau 3.3. [21]

Switch ID	Side	EdT ID	Type	Localisation
6-b	2-b	4-b	2-b	7-b

Table 3.2 – Structure d'une trame T1 du vecteur réponse d'un EdT dans le cas de

Switch ID	Side	ET ID	Type	Nb Fautes	$Loc.n_1$	$Loc.n_2$	$Loc.n_3$
6-b	2-b	4-b	2-b	4-b	7-b	7-b	7-b

Table 3.3 – Structure d'une trame T2 de vecteur réponse pour 2 à 8 fautes détectées dans un Élément de Test.

Switch ID	Side	ET ID	Type	Vecteur Réponse
6-b	2-b	4-b	2-b	29-b

Table 3.4 – Structure d'une trame T3 de vecteur réponse pour un nombre de fautes.

Switch ID	Side	ET ID	Type	Nb EdT Non-fautifs
6-b	2-b	4-b	2-b	7-b

TABLE 3.5 – Structure d'une trame T4 de vecteur réponse routeur.

3.6.2 IP de test et prise de décision

Le module de test délocalisé déduit la meilleure approche pour éradiquer ou contourer la défaillance en fonction des types d'erreurs identifiées. À la lumière de ces constatations, il a un certain nombre d'options à envisager :

• Isolation de la totalité ou d'une partie du chemin erroné.

• Isolation du ZGS

• Re configuration totale ou partielle du routeur ou des routeurs défectueux Le ZGS a été reconfiguré.

Il est également possible de réorganiser les tâches qui ont été mises en œuvre dans le ZGS testé. Ils peuvent être déplacés vers d'autres ZGS dans le même nud ou délocalisés à un autre nud en raison d'un manque de ressources matérielles.

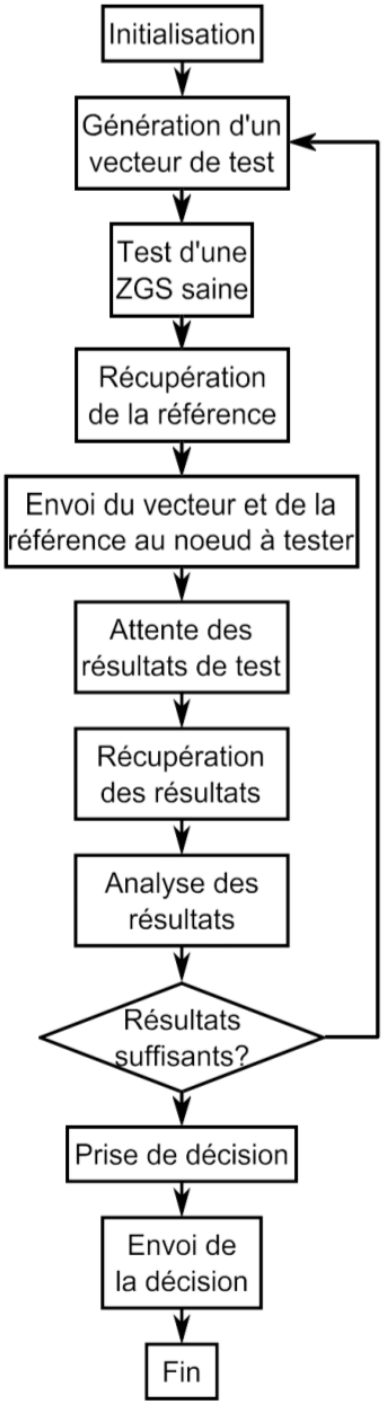


FIGURE 3.9 – Algorithme du principe de fonctionnement de l'IP test.

3.7 Conclusion

Dans ce chapitre, nous avons montré et détaillé la conception de mécanismes de test logiques adaptés à un contexte d'auto-organisation pour les structures d'intercommunication ou de communication sur le fil (NoC) des systèmes de réseau re configurables basés sur la technologie FPGA.. La principale nouveauté des techniques développées est qu'elles permettent la création d'un système multi-nuds re configurable capable d'échanger et d'interagir, permettant une gestion avancée des tâches et une autogestion des mécanismes de tolérance aux pannes.[21]

Par étude de la proposition architectural on a conclu que cette architecture a forcement touché les grands aspects de sécurité qui sont l'authentification l'intégrité et la disponibilité et sa apparaitre comme suite :

l'authentification :est assure par l'identification des trame par id switch et id élément de test et cela garantie l'identification de trame sous 4 forme bien défini préalablement pour interdire tous attaque non prévu ou un fichier malveillant ou non autorise Lorsque L'authentification est garantie on passe à l'intégrité .

L'intégrité :les donnes sont transférées en tous sécurité dans la trame entre les nœuds défaillant et les nœuds test par le codage de cette trame.

Aussi dans un autre lieu l'intégrité est assure lorsque les fichiers de configuration bitstream sont transfère par le nœud défaillant une confirmation de réception de la part le nœud testeur a été délivre pour assurer l'accuse de réception aussi sont stockée sans aucun modification ou détruit de façon non autorise les donnes sont préserver en tout.

Aussi le codage des trame c'est l'un de notion base de sécurité .

La disponibilité :est garante par ce système car il est auto-re configurable il maintient son fonctionnement automatiquement et aucun des traitements sont stoppée sur une durée intermédiaire.

Un transfert des tache d'un zone défectueux vers un autre zone pour assurer l'intégrité et la sécurité des tache et la préservation de ces dernier aussi assure la disponibilité et sur de fonction.

CHAPITRE 4

L'IMPLÉMENTATION

4.1 Introduction

L'architecture multi-nœud qui a été proposé par Mikael Heil repose sur l'auto-détection par test et correction d'erreur au sein, des structures NoC du nœud communication d'un système réseau

Cette auto-détection d'erreur dans notre architecture aide beaucoup à la sécurité de ce système contre les menaces inaperçue et tous les bugs possibles au sein d'un nœud et cela apparaitre dans les différents mécanismes de test entre les entités de l'architecture

Pour vérifier le bon fonctionnement de cette architecture et leur état de sécurité on a proposé et organisé le travail qui commence par une spécification abstraite du problème en définissant le rôle du réseau qui envoyer et recevoir des paquets après on a passé aux différents modèles qui sont reliés entre eux par une relation appelée raffinement.

Dans ce chapitre nous allons voir la spécification et la vérification formelle de ces différents raffinements avec la méthode évent B

4.2 Vue d'ensemble de Vertex algorithmes de coloration

Brisure de symétrie a toujours été un problème central dans les systèmes distribués. Plusieurs techniques ont été mises au développement afin de parvenir, comme les algorithmes de coloration de graphe.

4.2.1 La Coloration

Une coloration d'un graphe consiste en l'attribution de couleurs aux sommets, de telle sorte que deux sommets adjacents n'aient jamais la même couleur.
Le nombre chromatique $\chi(G)$ d'un graphe G est le nombre minimum de couleurs nécessaires à sa coloration, c'est-à-dire le plus petit nombre de couleurs permettant de colorier tous les sommets du graphe sans que deux sommets adjacents soient de la même couleur[22]
Ces règles sont généralement appliquées sur les graphes simples (connecté, ir réflexive, non orienté, non pondéré). Dans notre cas, on n'a pas de sommet mais on va garder le principe pour l'appliquer sur notre réseau.

4.2.2 Aperçu de Vertex algorithmes de coloration

les algorithmes de coloration de sommets peuvent être classés en deux catégories :
•Algorithmes utilisant des techniques centralisées [23] [24] : Le mot «centralisée» implique qu'il existe au moins un «administrateur» qui décide pour la coloration de graphe. **Il peut s'agir d'un sommet du graphe** ou d'une entité ce qui n'est pas une partie de celui-ci, avec la connaissance complète de celle-ci (de la structure, des bords, nombre de sommets...). Quand "L'administrateur" découvre une coloration correcte,**il envoie un message à tous les sommets que l'algorithme est compléter et leur donne leurs couleurs.**
•Algorithmes utilisant des techniques distribuées : les algorithmes de coloration de graphes centralisées ont été développés en premier. Mais certains problèmes, qui ne peuvent être résolus à l'aide de ces solutions, dans le but de les améliorer ou à en découvrir de nouveaux. Ces nouveaux algorithmes impliquer tous les sommets du graphe qui est coloré et les sommets ont leur propre «intelligence» : en général, ils choisissent leurs propres couleurs en utilisant les probabilités et aléatoire, savoir quand ils ont choisi la même couleur que son voisin, et quand ils ont une bonne couleur, dans ce cas, ils se retirent de la courbe non coloré [25] [26] [27] [28] [29] [30] .
Dans ce travail, nous nous concentrons sur le développement d'algorithmes utilisant des techniques distribués

4.2.3 Domaine d'Applications des algorithmes de graphes colorés

Il existe de nombreuses applications pratiques des algorithmes de graph coloré qui comprennent :

•Planification [31] : les algorithmes de coloration de graphes peuvent être utilisés pour commander un ensemble des nœuds. Deux nœuds sont considérés comme étant adjacents quand ils peuvent avoir lieu en même temps. Le but est d'éviter que les emplois adjacents se produisent en même temps. Mais dans notre cas il peut y avoir deux nœud qui ont le même emploi sauf les nœuds test ne peuvent pas corriger le même nœud défaillant en même temps.
• Chaque nœud correct doit être coloré en vert, un nœud correct est un nœud qui peut envoyer et recevoir des paquets.
•Chaque nœud défaillant doit être coloré en rouge, un nœud défaillant est un nœud qui ne peut ni envoyer ni recevoir des paquets ou l'un des deux.
•Chaque nœud test doit être coloré en bleu, un nœud test est un nœud choisi pour corriger le nœud défaillant.

4.3 Spécification Et Vérification Formelle

4.3.1 Niveau abstrait

Nous commençons par une spécification abstraite du problème en définissant le rôle du réseau qui envoyer et recevoir des paquets
On définit alors deux ensemble dans ce niveau qui sont : les nœuds existant (NODES) et paquets (PACKETS) envoyé par une source unique (SRC) et reçu par une unique destination (DST), notant que les sources sont différents des destinations la spécification des axiomes sont écrit dans le contexte du ce niveau comme suite

```
AXIOMS
    axm1:   NODES ≠ ∅ not theorem ›
    axm2:   PACKETS ≠ ∅ not theorem ›
    axm3:   src ∈ PACKETS ⟶ NODES  not theorem
    axm4:   dst ∈ PACKETS ⟶ NODES  not theorem
    axm5:   ∀ p· p∈ PACKETS ⇒ src(p) ≠ dst(p)
END
```

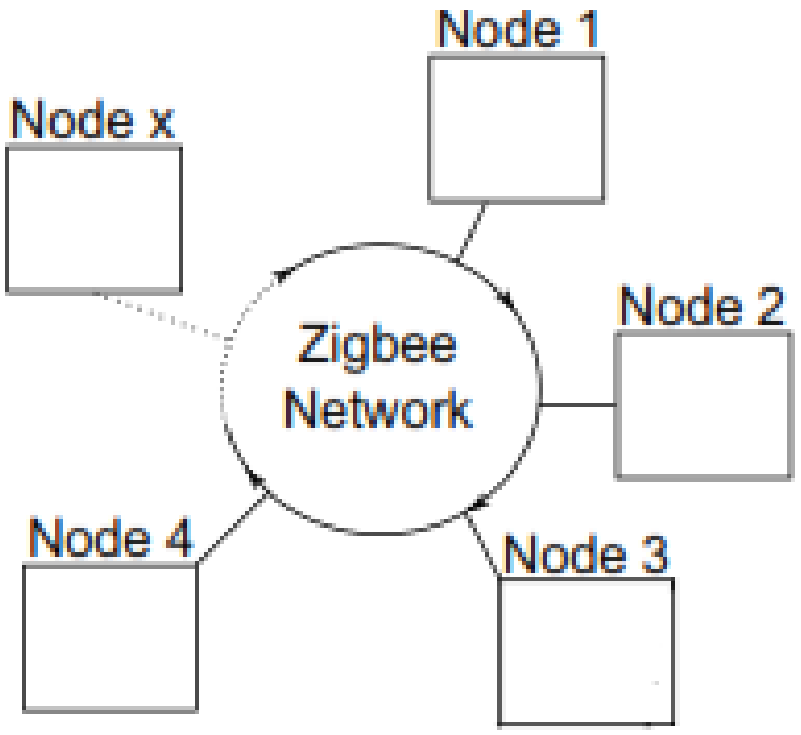


FIGURE 4.1 – Système Auto-Organise Multi-Noeud En Réseau Applique Aux Communication Sans Fil

On définit les variables : sent et rcvd qui nous permettent d'effectuer les actions SEND et RECEIVE. Les invariants suivant sont décrits dans la machine du niveau abstrait comme suit

```
INVARIANTS
    inv1:   sent ∈ NODES ↔ PACKETS not theorem ›
    inv2:   rcvd ∈ NODES ↔ PACKETS  not theorem ›
    inv3:   ran(rcvd) ⊆ ran(sent) not theorem ›chaque paquet reçu a été envoyé
    inv4:   ∀ s,p· s∈NODES ∧ p∈ PACKETS ∧ s↦p ∈ sent ⇒ s = src(p) not theorem ›
    inv5:   ∀ d,p· d∈NODES ∧ p∈ PACKETS ∧ d↦p ∈ rcvd ⇒ d = dst(p) not theorem ›
    inv6:   ∀ s1, s2, p· s1∈NODES ∧s2∈NODES ∧ p∈PACKETS ∧ s1↦p∈ sent ∧ s2↦p∈sent⇒ s1=s2
    inv7:   ∀ d1, d2, p· d1∈NODES ∧d2∈NODES ∧ p∈PACKETS ∧ d1↦p∈rcvd ∧ d2↦p∈rcvd⇒ d1=d2
EVENTS
```

Les valeurs initiales des variables sont vides

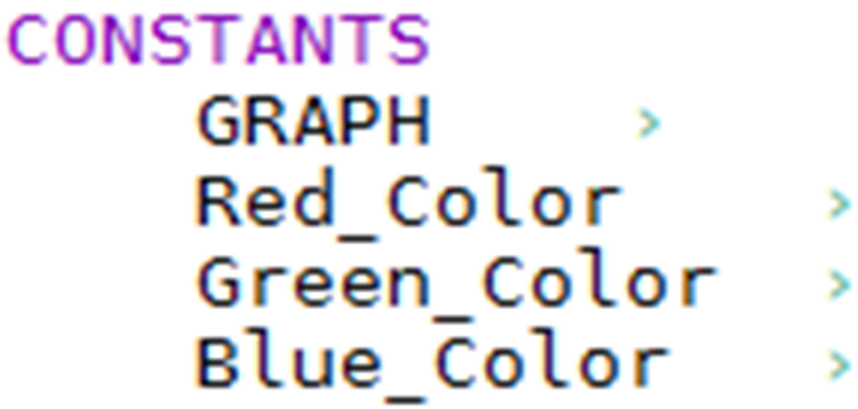
```
INITIALISATION:     not extended ordinary ›
    THEN
        act1:   sent := ∅ ›
        act2:   rcvd := ∅ ›
    END
```

L'événement SEND fait l'action :

L'événement RECEIVE fait l'action

4.3.2 Premier raffinement

Nous supposons que le graphe est donné sur un ensemble de nœuds. Après, nous définissons un ensemble de couleurs rouge, vert, et bleu (Red-Color, Green-Color, Blue-Color), qui vont être associé aux nœuds lors de l'exécution de l'un des algorithmes de coloration de graphes Nous spécifions les axiomes de ce niveau comme suite



```
AXIOMS
    axm1:   GRAPH ∈ NODES ↔ NODES not theorem ›
    axm2:   GRAPH ≠ ∅ not theorem ›
    axm3:   COLOR ≠ ∅ not theorem ›
    axm4:   ∀c·c∈COLOR⇒ c=Red_Color ∨ c=Green_Color ∨ c=Blue_Color not theorem
END
```

Dans la machine de ce niveau raffiné du niveau abstrait on va supposer que tous les nœuds peuvent envoyer et recevoir des paquets et ca nous permet alors de les colorer en vert (voir la figure.). La variable CCorrecte Node est définie avec la propriété suivante :

```
INVARIANTS
    inv1:   CCorrectNode ∈ NODES ⇸ COLOR theorem ›un noe
    inv2:   CorrectNode ∈ NODES  not theorem ›
```

Cette propriété permet de définir un nœud fait partie de l'ensemble des nœuds colorés

```
green_color:    extended ordinary ›
    ANY
        node    ›
        GrnClr  ›
        p   ›
    WHERE
        grd7:   p ∈ PACKETS not theorem ›
        grd1:   node∈ NODES not theorem ›
        grd2:   GrnClr ∈ COLOR not theorem ›
        grd3:   node ∈ dom(GRAPH) not theorem ›
        grd4:   node = CorrectNode  not theorem ›
        grd5:   GrnClr ∉ CCorrectNode[GRAPH[{node}]] not theorem ›
        grd6:   CorrectNode↦p ∈ sent not theorem ›
        grd8:   CorrectNode↦p ∈rcvd not theorem ›
    THEN
        act1:   CCorrectNode(node) :=GrnClr  ›
    END
```

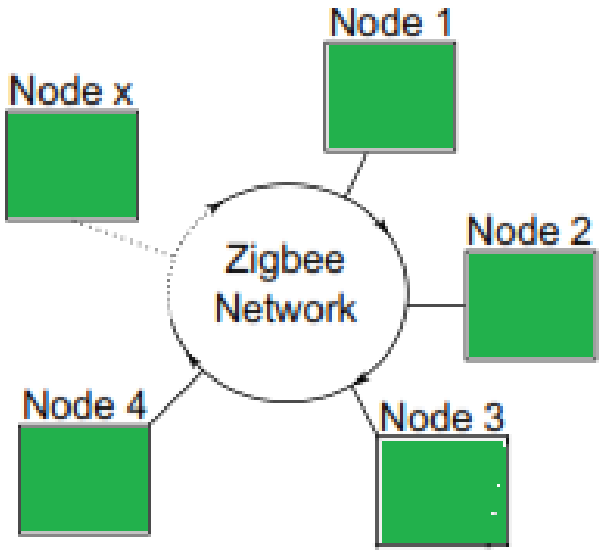


FIGURE 4.2 – Coloration Des Nœuds Du Système Auto-Organise En Vert

4.3.3 Deuxième raffinement

Afin de coloré les nœuds défaillants par la couleur rouge, Deux variables seront ajoutés à ce niveau, FaultyNode et CFaultyNode qui sont définis avec les propriétés suivantes :

```
INVARIANTS
    inv1:   FaultyNode ∈ NODES not theorem ›
    inv2:   CFaultyNode ∈ NODES ⇸ COLOR not theorem ›
```

FaultyNode est un nœud défaillant donc il être coloré avec la couleur rouge l'action sera donc :

```
red_color:  not extended ordinary ›
    ANY
        node     ›
        rdClr    ›
        p    ›
    WHERE
        grd1:   p∈ PACKETS not theorem ›
        grd2:   node∈ NODES not theorem ›
        grd3:   rdClr ∈ COLOR not theorem ›
        grd4:   node ∈ dom(GRAPH) not theorem ›
        grd5:   node = FaultyNode  not theorem ›
        grd6:   rdClr ∉ CFaultyNode[GRAPH[{node}]] not theorem ›
        grd7:   FaultyNode ↦p ∉sent not theorem ›
        grd8:   FaultyNode ↦p ∉rcvd not theorem ›
    THEN
        act1:   CFaultyNode(node) :=rdClr   ›
    END
```

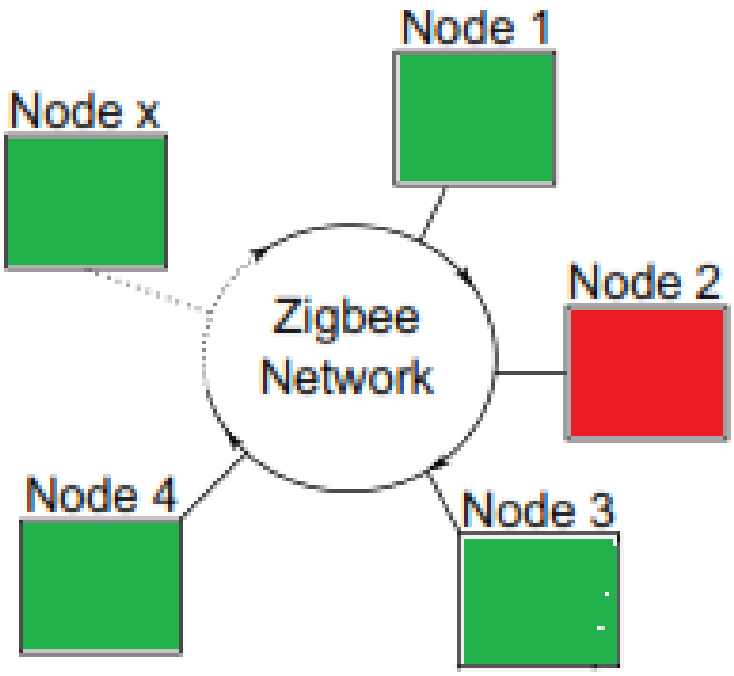


FIGURE 4.3 – Coloration Des Nœuds Défaillants Du Système Auto-Organisé En Rouge.

La spécification obtenus dans la partie au-dessus a été proposés par MAHOUEL Youcef dans Thème Spécification et Vérification Formelle d'une architecture à base d'un réseau sans fil orientée NoC

Dans la partie suivante nous allons proposer la suite de la vérification formelle qui consiste a vérifié l'état de la sécurité de système l'ors la détection d'erreur au sein d'un nœud de l'architecture auto-re configurable, notre proposition est comme suite

4.3.4 Troisième raffinement

Afin de sélectionner AdoptNode parmi l'ensemble des nœuds, Deux variables seront ajoutés à ce niveau, adoptNodes , BitstreamPacket qui sont définis avec les propriétés suivantes :

```
INVARIANTS
    inv1:   AdoptNodes ∈ NODES not theorem ›AdoptNode est un noeud qui récupère les
    inv2:   BitstreamPacket ∈ PACKETS not theorem ›Bitsream est le fichier de reconfig
```

AdoptNodes est un nœud qui recevoir les tâches des nœuds défaillants donc il doit respecter certain conditions qui sont

Il ne doit pas être un FaultyNœud, ce qui est définit par la coloration en vert

La condition la plus importante c'est qui il doit avoir un fichier de configuration bitstrem, ce qui est défini dans axiome 1

Il possède les sources logiques nécessaires pour adopter une tache ce qui est définit dans axiome 2

Les axiomes sont comme suit :

```
AXIOMS
    axm1:   BITSTREAM ∉ ∅ not theorem ›
    axm2:   LOGICRSC ∉ ∅ not theorem ›
    axm3:   Bitstrm ∈ BITSTREAM not theorem ›
    axm4:   Lrsc ∈ LOGICRSC not theorem ›
END
```

Si AdoptNodes ne possède pas un fichier de configuration bistream il doit avoir la possibilité de le recevoir par le nœud défaillant comme il peut envoyer ce fichier vers un autre nœud dans l'architecture selon le besoin

Les actions sont comme suit

```
SENDT:  not extended ordinary ›
    REFINES
        SEND
    ANY
        p    ›
        s    ›
    WHERE
        grd1:   s ∈ NODES not theorem ›
        grd2:   p ∈ PACKETS not theorem ›
        grd4:   p ∈ sent[{s}] not theorem ›
        grd5:   AdoptNodes↦p ∈ sent not theorem ›
        grd6:   src(BitstreamPacket)= AdoptNodes not theorem ›
    THEN
        act1:   sent ≔ sent ∪{s↦p} ›
    END
```

```
RECEIVET:   not extended ordinary ›
    REFINES
        RECEIVE
    ANY
        p    ›
        d    ›
        s    ›
    WHERE
        grd1:   p ∈ PACKETS not theorem ›
        grd2:   d ∈ NODES not theorem ›
        grd3:   d= dst(p) not theorem ›
        grd4:   s ∈ NODES not theorem ›la source c'est le node faulty
        grd5:   p∈ sent[{s}] not theorem ›
        grd6:   s= src(p) not theorem ›
    THEN
        act1:   rcvd ≔ rcvd ∪{d↦p} ›alors adopt node peut recevoir les paquet du
    END
```

Après la vérification des conditions nécessaires :
L'action RECIVET s'effectue , le NoeudAadopt peut reçoit maintenait des paquets de la part de nœud défaillant ces paquets contient les taches à adopter et le fichier de configuration bitstream et d'autre information nécessaire qui concerne le nœud défaillant
Par l'affectation de ces taches le système maintien son fonctionnement et aucun traitement sera stoppé cela garanti l'aspect de sécurité qui est disponibilité de service à tout moment

4.3.5 Quatrième raffinement

Ce niveau set a sélectionné TestNode qui va tester le nœud défaillant,les variables de ce niveau, TestNode , CTestNode BitstreamPacket seront définis avec les propriétés suivantes :

```
INVARIANTS
    inv1:   TestNode ∈ NODES not theorem ›
    inv2:   CTestNode ∈ NODES ⇸ COLOR not theorem ›
    inv3:   BitstreamPacket ∈ PACKETS not theorem ›
```

TestNode est un nœud qui effectue des tests pour avoir ou se trouve exact l'erreur et la panne dans les nœuds défaillants et par la suite prendre une discussion, ce TestNode doit respecter certain conditions qui sont :
•Il ne doit pas être un FaultyNœud ce qui est défini dans la grd 9
•ne pas être occupé par une tâche prioritaire. La grd7 veut dire que ce nœud n'est pas occupé avec l'envoie des paquets
•ne pas être occupé par une tâche prioritaire. La grd8 veut dire que ce nœud n'est pas occupé avec réception des paquets
•Disposer des ressources matérielles nécessaires pour mettre en œuvre l'IP testeur
•disposer du bitsream de configuration de l'IP testeur. Les grd10 et grd11 respect cette règle parce que si le nœud ne dispose pas du fichier de re configuration il peut le recevoir à partir d'un notre nœud.
•Après avoir respecté les règles le nœud peut être choisi comme nœud test (voir figure.) et l'action de colorer le nœud test en bleu sera effectuée.
Après avoir respecté les règles le nœud peut être choisi comme nœud test (voir figure.) il va générer des vecteur de test pour tester lui-même et avoir des vecteur de référence aussi il va envoyer des vecteur de test et le vecteur de référence vers le nœud défaillant pour avoir une

réponse de la part de ce dernier, une décision concernent ce nœud sera effectuée

Nous avons remarqués que l'auto-re configuration fait des contrôles de test a chaque détection d'erreur pour rendre le système en état sein et garantir la sureté de fonction ce qui assure la sécurisation de note système par la suite

L'action de colorer le nœud test en bleu sera effectuée Comme suite :

```
blue_color:     not extended ordinary ›
    ANY
        node      ›
        blClr     ›
        p     ›
    WHERE
        grd1:   p∈ PACKETS not theorem ›
        grd2:   node∈ NODES not theorem ›
        grd3:   blClr ∈ COLOR not theorem ›
        grd4:   node ∈ dom(GRAPH) not theorem ›
        grd5:   node = TestNode  not theorem ›
        grd6:   blClr ∉ CTestNode[GRAPH[{node}]] not theorem ›
        grd7:   TestNode ↦p ∉sent not theorem ›regle 4
        grd8:   TestNode ↦p ∉rcvd not theorem ›regle 4
        grd9:   TestNode ≠ FaultyNode theorem ›testnode n'est pas un faultynode
        grd10:  src(BitstreamPacket)= TestNode theorem ›regle 3
        grd11:  dst(BitstreamPacket)= TestNode not theorem ›
    THEN
        act1:   CTestNode(node) :=blClr  ›
    END
```

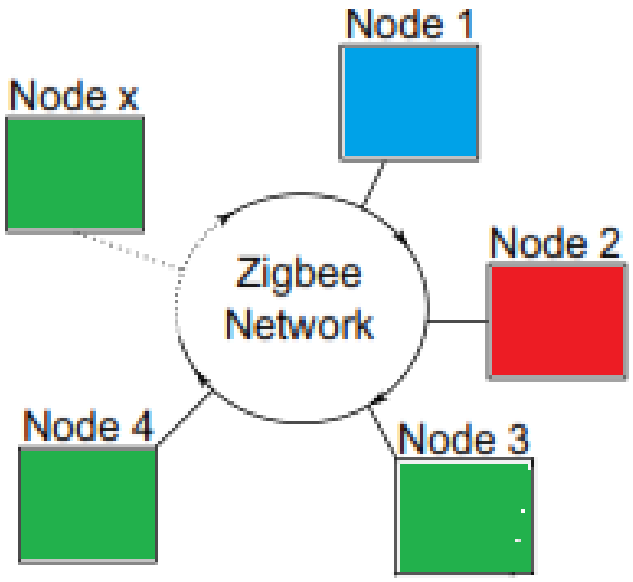


FIGURE 4.4 – Coloration Du Nœuds Du Système Auto-Organisé En Bleu.

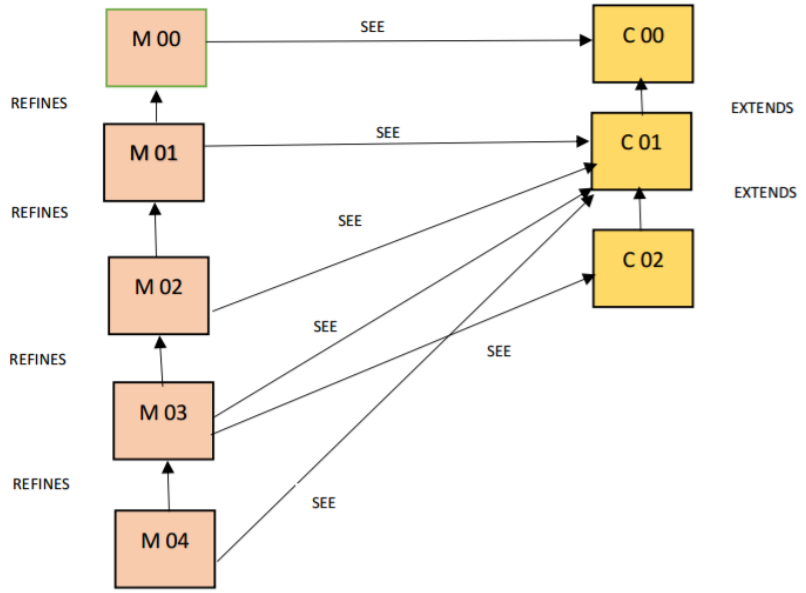


FIGURE 4.5 – Modélisation étape par étape de l'architecture NoC.

4.4 Résultat

Lors la détection automatique des erreurs dans notre architecture le système maintien son fonctionnement en respectant les aspects base de sécurité, c'est ce qui apparait dans l'auto-re configuration par les méthodes de test comme suite :

L'authentification est assure par l'envoi de trame le nœud défaillant vers les nœuds testeur ce trame contient l'identifiant de nœud défaillant qui permet de garantir que seulement le nœud défaillant de ce système peuvent envoyer une demande vers les autre nœuds de système.

La confidentialité : l'identification des nœuds défaillants garantir que seulement le nœud défaillant de ce système peuvent effectuer et envoyer leur demande vers un autre nœud sein de système.

La confidentialité est assurée par le maintien des tache de système aucun des traitements sont stoppée sur une durée intermédiaire .

L'intégrité est assure par le codage de trame les informations sans coder selon défirent forma pour garantir la sécurité des donner Au plus l'intégrité est assurée par le stockage des données et information et le bit Stream des nœuds défaillant et aucune modification sera effectuer.

Par l'utilisation des méthodes formelles évent B nous avons prouvé mathématiquement la sureté de fonction de ce système par la génération des obligations de preuve, ces obliga-

tions libèrent automatiquement ou de manière interactive par la déclaration de certaines conditions de fonctionnement de la méthode de test Nos résultats sont comme suit

Element Name	Total	Auto	Manual	Reviewed	Undischarged
Test M00	**0**	**0**	**0**	**0**	**0**
INITIALISATI...	0	0	0	0	0
inv1	0	0	0	0	0
inv2	0	0	0	0	0
inv3	0	0	0	0	0
inv4	0	0	0	0	0
inv5	0	0	0	0	0
inv6	0	0	0	0	0
inv7	0	0	0	0	0
SEND	0	0	0	0	0
RECEIVE	0	0	0	0	0

Element Name	Total	Auto	Manual	Reviewed	Undischarged
Test M01	**7**	**4**	**0**	**0**	**3**
INITIALISATI...	3	1	0	0	2
SEND	1	1	0	0	0
RECEIVE	2	2	0	0	0
green_color	1	0	0	0	1
inv1	2	1	0	0	1
inv2	0	0	0	0	0

Element Name	Total	Auto	Manual	Reviewed	Undischarged
Test M02	**4**	**3**	**0**	**0**	**1**
INITIALISATI...	3	3	0	0	0
SEND	0	0	0	0	0
RECEIVE	0	0	0	0	0
green_color	0	0	0	0	0
red_color	1	0	0	0	1
inv1	0	0	0	0	0
inv2	2	1	0	0	1

Element Name	Total	Auto	Manual	Reviewed	Undischarged
Test M03	**5**	**4**	**0**	**0**	**1**
INITIALISATI...	2	2	0	0	0
SEND	0	0	0	0	0
RECEIVE	0	0	0	0	0
green_color	0	0	0	0	0
red_color	0	0	0	0	0
inv1	0	0	0	0	0
inv2	0	0	0	0	0
SENDT	2	1	0	0	1
RECEIVET	1	1	0	0	0

Element Name	Total	Auto	Manual	Reviewed	Undischarged	
Test M04	**8**	**5**	**0**	**0**	**3**	
INITIALISATI...	3	3	0	0	0	
SEND	0	0	0	0	0	
RECEIVE	0	0	0	0	0	
green_color	0	0	0	0	0	
red_color	0	0	0	0	0	
blue_color	5	2	0	0	3	
inv1	0	0	0	0	0	
inv2	2	1	0	0	1	
inv3	0	0	0	0	0	

A la fin de notre travaille Vous remarquez la plus part des preuves sont faites automatiquement ce qui nous permet de dire qu'on n'a pas forcé la preuve manuellement,Nous avons obtenu notre but qui est prouvé mathématiquement la sureté de fonction et la forte sécurisation de notre système embarque

4.5 Conclusion

Ce chapitre présent les premiers éléments d'une méthodologie globale de vérification formelle par les méthodes évent B sur une architecture multi-noud auto-organise à base de réseau sans fil NoC ,notre but dans ce chapitre est de prouver mathématiquement tous les conditions possible et les mécanismes nécessaires pour une méthode de test des nœuds défaillants ,cette méthode de test garantis le bon fonctionnent et la sureté de système ce qui assure forcement la sécurité de système.

Dans ce chapitre nous avons vu les différents niveaux de raffinement et leur spécification formelle nous avons prouvé chaque niveaux afin de passer aux niveaux suivant ainsi de suite jusqu'à la dernière raffinement comme suite.

Première raffinement :

Nous supposons que le graphe est donné sur un ensemble de nœuds. Ensuite, nous définissons un ensemble de couleurs (Red-Color, Green-Color, Blue-Color), dont les composantes seront les couleurs choisies par les nœuds lors de l'exécution de l'un des algorithmes de coloration de graphes et après le graphe va être coloré en vert

Deuxième raffinement :

Coloration avec la couleur rouge, le nœud défaillant doit être coloré avec la couleur rouge

Troisième raffinement :

Assurée que les taches d'un nœud défaillant ont été préserves dans un autre nœud sein

Quatrième raffinement :

Vérifier et assurer que le nœud test respecte les conditions de test et lui colore en bleu Assurer l'envoi des donner important et le fichier de configuration par le nœud défaillant ver le nœud test et l'accuse de réception par ce dernier

Après avoir respecté les règles le nœud peuvent être choisies comme nœud test et l'action de colorer le nœud test en bleu sera effectuée.

Comme résultat les preuves de ces niveaux sont automatiquement prouvées ce qui nous permet de dire qu'on n'a pas forcé la preuve manuellement car nous avons bien spécifié nos conditions et invariants dans chaque niveaux , notre résultat montre que notre système est sur de fonction et qui il est sécurise par la méthode formelle évent B

CONCLUSION GÉNÉRALE ET PERSPECTIVE

L'évolution actuelle des architectures des SoC tendant vers des MPSoC re configurables dynamiquement et sûres de fonctionnement à base de technologie FPGA. En effet, la flexibilité et la bande passante sont nécessaires à une mise en œuvre de SoC contenant des IP. Le support de communication associé joue un rôle important dans les MPSoC qui nécessite d'introduire des mécanismes de sûreté de fonctionnement (SdF) afin de prévenir la défaillance d'un système.

L'objectif est la vérification de la sécurité d'une structure architecturale de type NoC dans le système embarqué compte tenu de leur importance dans la conception des systèmes puce actuels. Il s'agit donc de prouver qu'une architecture d'un routeur NoC soit validée et sur de fonction par l'utilisation de la méthode évent B compte tenu des propriétés dont celle de la sureté de fonctionnement que le routeur doit posséder.

La méthode formelle évent B consiste alors à prouver mathématiquement tous les conditions possibles et les mécanismes nécessaires pour l'auto-détection et l'auto-re configuration des nœuds défaillants, ces mécanismes aident beaucoup a sécurisé le système par la détection des menaces inaperçues et tous les bugs ou attaque possibles au sein d'un nœud et rendre par la suite le système a son état sein et maintien son fonctionnement sans aucun interrompions

Afin de vérifier ces mécanismes on a proposé et organisée le travail qui commence par une spécification abstraite du problème en définissant le roule du réseau qui envoyer et recevoir des paquets après nous avons passés aux différents modèles qui sont reliés entre eux par une relation appelée raffinement

Nous avons vu les différents niveaux de raffinement et leur spécification formelle, nous avons prouvé chaque niveaux afin de passer aux niveaux suivant ainsi de suite jusqu'à l'ar-

rive aux derniers niveaux, les preuves de ces niveaux sont automatiquement ce qui nous permet de dire qu'on n'a pas forcé la preuve manuellement, puisque nous avons bien spécifiés nos conditions et invariants dans chaque niveaux

Notre travail se termine par une proposition afin d'appliquer des techniques de coloration des nœuds par un algorithme spécifié et prévu par le cadre de l'événement B (Event-B).

perspective :

Notre perspective consiste à développer une nouvelle façon de vérification de l'architecture NoC basé sur la traduction de modèles Event- B par une nouvelle approche de raffinement. Cette approche de raffinement sert à bien vérifier et prouver mathématiquement tous les étapes du mécanisme de test en détail et les conditions associer à la génération des vecteurs de tests au sein d'nœud auto-re configurable

Cette nouvelle approche nous permet de détailler de plus en plus la spécification de toute l'interaction et l'interconnexion possible au sein d'nœud l'or sa re configuration automatique, ce qui nous assure la grande fiabilité et la sureté de fonction dans le système, et la preuve de leur fort état de sécurité.

BIBLIOGRAPHIE

[1] Amin Ouafi,Thèse Mapping multi-objectif à l'aide d'un algorithme génétique pour les applications embarquées ,2008-2009

[2] Alain Aina,La gestion réseau et le NOC : concepts, pratiques, et outils Maputo, Mozambique , AfNOG 2005

[3] https ://www.clicours.com/mémoire-online-une-architecture-d'un-réseau-sur-puce-noc-dediee-au-routage-de-paquets/ consulte 21/04/2021

[4] Taha, Bachir Meziane,CVers la modélisation à base de B-Event d'un routeur NoC : Mémoire de Master GI. Univ de Tiaret. 2011.

[5] Srinivasan, K. C,Application Spécifique Network-on-Chip Design with Guaranteed Quality Approximation Algorithmes. Design Automation Conférence. Dept. of Comput. Sci. Eng., Arizona State Univ., Tempe, AZ. janvier 2007.

[6] Mohamed Hadj Kacem Modélisation des applications distribuées architecture dynamique : Conception et Validation, l'Université Toulouse III - Paul Sabatier .Le Jeudi 13 Novembre 2008

[7] https ://www.futura-sciences.com/tech/definitions/technologie-fpga-8700/ 15-03-2021

[8] MIkael HeIl,conception architectural pour la tolérance ou faut d'un système auto-organisée multi noued en réseau à base de NOC re-configurable ,université lorraine France , 04 décembre 2015 .

[9] Rajeev, Alur. Techniques for automatic verification of real-time. Stanford university. August 1991

[10] Xavier, Renault.Mise en oeuvre de notations standardisées, formelles et semi-formelles dans un processus de développement de systèmes embarqués temps-réel répartis. de-

cember 2009. Université Pierre et Marie Curie - Paris VI (03/12/2009), Fabrice Kordon (Dir.)

[11] https ://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445198-z-notation-en-langage-informatique-definition-et-utilisation/ 30-04-2021

[12] Article John S. Fitzgerald Peter Gorm Larsen Marcel Verhoef Wiley Encyclopedia of Computer Science and Engineering

[13] https ://www.methode-b.com/methode-formelle/ 05-05-2021

[14] M.Jastram, Rodin User's Handbook, 2012.

[15] A. Matoussi, «Construction de spécification formelles abstraites dirigée par les buts Ordinateur et société,» Thèse de doctorat. Université Paris-Est, 2011.

[16] Manamiary Bruno Andriamiarina,Dominique Méry-Neeraj Kumar Singh. Revisiting Snapshot Algorithme by Raffinement-based Techniques. LORIA, Univ. de Lorraine, Vandœuvre-les-Nancy, France. September 2012.C BN

[17] Imen, Sayar. D'Event -B vers UML/OCL en passant par UML/EM-OCL.Cornell university library. 2012.

[18] Jean-Raymond, Abrial and L. Mussa. Introducing dynamic constraints in B. In B'98 : Proceedings of the Second International B Conference on Recent Advances in the Development and Use of the B Méthode, pages 83–128, London, UK, 1998. Springer- Verlag.

[19] Olivier Ligot, Jens Bendisposto et Michael Leuschel,«Debugging Event-B Models using the ProB Disprover Plug-in,» EU funded research projects : IST 511599 RODIN, technical repport for deploy project, AFADL 2007, pp 1-4.

[20] Taha, Bachir Meziane.Vers la modélisation à base de B-Event d'un routeur NoC : Mémoire de Master GI. Univ de Tiaret. 2011.

[21] MIkael HeIl,conception architectural pour la tolérance ou faut d'un système auto-organisée multi noued en réseau à base de NOC re-configurable ,université lorraine France , 04 décembre 2015

[22] http ://www.gymomath.ch/javmath/polycopie/th-graphe7.pdf 11-06-2021

[23] Brian R. Nickerson. Graph coloring register allocation for processors with multi-register operands. In Pro-ceedings of the ACM SIGPLAN 1990 conference on Programming language design and implémentation , PLDI'90, pages 40–52, New York, NY, USA, 1990. ACM.

[24] Noga Alon and Nabil Kahale. A spectral technique for coloring random 3-colorable graphs. SIAM J. Comput. 26 :1733–1748, Décembre 1997.

[25] K. R. Duffy, N. O'Connell, and A. Sapozhnikov.Complexity analysis of a decentralised graph colouring algorithm. Inf. Process. Lett. , 107 :60–63, July 2008.

[26] Y. Métivier, J.M. Robson, N. Saheb-Djahromi, and A. Zemmari.An analysis of an optimal bit complexity randomised distributed vertex colouring algorithm (extended abstract). OPODIS , pages 359–364, 2009.

[27] Johannes Schneider and Roger Wattenhofer. A new technique for distributed symmetry breaking. In Proceeding of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing , PODC '10, pages 257–266, New York, NY, USA, 2010. ACM.

[28] Jennie C. Hansen, Marek Kubale, Lukasz Kuszner, and Adam Nadolski.Distributed largest-first algorithm for graph coloring. In Euro-Par , pages 804–811, 2004.

[29] Marek Kubale and Lukasz Kuszner. A better practical algorithm for distributed graph coloring. In PARELEC , pages 72–75, 2002.

[30] Lehning Hervé. Téléphonie cellulaire - coloriages pour allocation de fréquences. La Recherche - N.390 , pages 98–99, 2005.

[31] Mohammad Malkawi, Mohammad Al-Haj Hassan, and Osama Al-Haj Hassan.A new exam scheduling algorithme using graph coloring. Int. Arab J. Inf. Technol. , 5(1) :80–86, 2008.