Effet des radiations cosmiques sur les systèmes numériques embarqués dans les véhicules automobiles

par

Thaweb HAJJI

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE ÉLECTRIQUE
M. Sc. A.

MONTRÉAL, LE 09 AOUT 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Claude Thibeault, directeur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Ricardo Izquierdo, président du jury
Département de génie électrique à l'École de technologie supérieure

M. François Blanchard, membre du jury
Département de génie électrique à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 29 JUILLET 2021

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

C’est avec un grand plaisir que je réserve cette page en signe de gratitude et de profonde reconnaissance à tous ceux qui ont apporté l’assistance nécessaire au bon déroulement de ce travail.

Je tiens à exprimer ma profonde gratitude et mes sincères remerciements à mon directeur de mémoire, le Professeur Claude Thibeault tout d’abord pour son soutien durant toute la période de mémoire, ensuite pour ses directives fructueuses, ses conseils pertinents ainsi les profondes connaissances que j’ai acquises grâce à lui, et finalement pour la confiance particulière qu’il m’a accordée.

Mes remerciements sont également destinés aux honorables membres de jury pour m’avoir fait l’honneur d’accepter l’évaluation du mémoire et la participation à la soutenance.

Je tiens à présenter mes chaleureux remerciements à ma famille, mes parents, ma sœur qui attendent impatiemment ce jour. Que nul remerciement ne puisse exprimer ce que je vous dois.

Ce projet est le fruit d’un fort dévouement et d’une extrême attention dédiée à sa réussite.

À tous ces intervenants, je présente mes remerciements et mon respect.

Effet des radiations cosmiques sur les systèmes numériques embarqués dans les véhicules automobiles

Thaweb HAJJI

RÉSUMÉ

L'utilisation de l'électronique embarquée dans le domaine de l'automobile ne cesse d'augmenter et surtout avec l'apparition des véhicules autonomes. Ce domaine se caractérise par des requis très sévères au niveau de la fiabilité des circuits intégrés, dont les circuits de type FPGA. Ces requis de fiabilité sont tels qu'il devient pertinent de s'interroger de l'effet de certaines sources de corruption habituellement négligées, comme les radiations cosmiques, sachant que les FPGA les plus répandus, ceux basés sur le mémoire statique, y sont particulièrement sensibles. Ceci oblige les chercheurs à développer des stratégies permettant à ces circuits de tolérer l'effet des radiations et d'améliorer leur fiabilité. Le développement de telles stratégies demande qu'elles soient validées tout au long du processus de conception, et idéalement le plus tôt possible.

Dans ce mémoire, nous validons des modèles structuraux de pannes à haut niveau d'abstraction, permettant d'imiter l'effet des événements singuliers (Single Event Upset, SEU) induits par les radiations cosmiques, et ainsi constater leurs influences très tôt dans le processus de conception. De manière plus spécifique, nous validons une stratégie qui suppose que l'injection de pannes aux entrées/sorties d'un bloc de haut niveau, à l'aide des modèles structuraux, mène à des résultats fautifs qui sont représentatifs de ceux qui seraient obtenus sur le même bloc, implémenté au niveau matériel, était soumis au rayonnement cosmique.
Pour effectuer cette validation, un montage expérimental a été développé, selon une approche de type matériel dans la boucle (Hardware In the Loop, HIL). Ce montage implique un modèle de simulation Matlab/Simulink et une plaquette électronique contenant un circuit FPGA basé sur la mémoire statique. Dans ce dernier est implémenté un filtre de Sobel, pris pour la détection des obstacles dans le contexte de véhicules autonomes. Le circuit FPGA contient également un module commercial, le SEM, permettant l'émulation de SEU. Le modèle Matlab/Simulink permet de mesurer et comparer l'effet aux sorties du filtre des deux types de perturbations (SEU et modèles structuraux de pannes de haut-niveau). Ces mêmes effets sont mesurés et comparées à la sortie d'un bloc habituellement placé en aval du filtre de Sobel dans la détection des obstacles, à savoir une transformée de Hough. Des métriques de comparaison ont été décrites à cette fin. Cette étude a permis d'obtenir des résultats validant la stratégie d'injection des pannes à l'aide de modèles structuraux affectant les entrées/sorties d'un bloc, qui permet de couvrir les SEU injectés par le SEM.

Mots-clés : injection de pannes, FPGA, rayons cosmiques, collé-à, Filtre de Sobel, transformée de Hough, Matlab/Simulink, saboteur, émulation

Effect of cosmic radiation on digital systems in automotive vehicles Effect of cosmic

Thaweb HAJJI

ABSTRACT

The use of on-board electronics in the automotive field continues to increase and especially with the appearance of autonomous vehicles. This domain is characterized by very stringent reliability requirements for integrated circuits, including field-programmable gate arrays (FPGAs). These reliability requirements are such that it becomes relevant to investigate in-field error sources that are usually neglected, such as cosmic radiation, knowing that the most popular FPGAs, the SRAM-based ones, are particularly sensitive to this error source. This requires researchers to develop strategies that allow these circuits to tolerate the effect of radiation and improve their reliability. The development of such strategies requires that they be validated throughout the design process, and ideally as early as possible.

In this thesis, we validate high-level structural fault models, allowing to mimic the effect of singular events (SEU: Single Event Upset) induced by cosmic radiation, and thus observe their influences very early in the design process. More specifically, we validate a strategy that assumes that applying these high-level structural fault models only at the inputs/outputs of a given block leads to faulty results similar to those obtained by bombarding the block with ionizing particles. An experimental setup has been developed to perform this validation. The setup is based on a Harware-In-the-Loop (HIL) approach. It contains a Matlab/Simulink simulation model as well as a SRAM-based FPGA board. The FPGA is used to implement a Sobel filter, which is often utilized in the detection of obstacles for autonomous cars. It is also used to implement an IP core, called SEM, allowing SEU emulation. The Matlab/Simulink model allows measuring and comparing the effects of both error sources (emulated SEUs and high-level structural fault models) at the Sobel filter output. The same effects are also measured and compared at the output of a block usually found downstream of the Sobel filter for obstacle detection, namely a Hough transform. The metrics used for these comparisons are described. The results validate the high-level fault injection strategy targeting block I/Os.

Keywords: fault injection, FPGA, cosmic radiations, stuck-at, Sobel filter, Hough transform, Matlab/Simulink, saboteur, emulation

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

HIL	Hardware In the Loop
HILS	Hardware In the Loop Simulation
HLS	Hardware in the Loop Simulation
SEU	Single Event Upset
LUT	Look Up Table
FI	Fault Injection
CMOS	Complementary Metal-Oxide-Semiconductor
VHDL	VHSIC Hardware Description Language
FPGA	Field Programmable Gate Arrays
RTL	Register Transfer Level
SRAM	Static Random Access Memory
SEM	Soft Error Mitigation
VERIFY	VHDL- based Evaluation of Reliability by Injection Fault efficiently
FAST	Fault Simulator for Transient
ASIL	Automotive System Integrity Level
HCU	Hybrid Control Unit
PLD	Programmable Logic Device
OMS	Organisation Mondiale de la Santé

INTRODUCTION

Les radiations cosmiques sont, à juste titre, associées aux missions spatiales. Après tout, c'est l'exploration spatiale qui a permis de prendre conscience de leur effet sur l'électronique embarquée. L'utilisation de plus en plus soutenue des modules électroniques dans les avions, combinée aux altitudes de vol plus élevées et à la réduction de la taille des transistors, a par la suite révélé que l'électronique avionique embarquée pouvait également être affectée, dans une moindre mesure, par les radiations cosmiques (Souari, 2016).

Les radiations cosmiques sont aussi présentes au niveau du sol. Même si leur niveau d'énergie et leur fréquence d'apparition sont significativement réduits par rapport aux altitudes de vol, ils peuvent s'avérer suffisants pour constituer une source de préoccupation pour certaines applications où les niveaux de fiabilité exigés sont très élevés. C'est notamment le cas pour le domaine de l'automobile, où l'électronique est de plus en plus présente, y compris dans ces fonctionnalités les plus critiques. Cette tendance risque d'être amplifiée par l'émergence des véhicules autonomes, de par l'effort de calcul requis pour certaines opérations telles la détection et la reconnaissance des obstacles sur la route.
Parmi les circuits intégrés utilisés par l'industrie automobile et particulièrement dans les véhicules semi-autonomes et autonomes, on retrouve les FPGA (Wei Kaijie, Koki Honda, et Hideharu Amano. 2018). Dans le contexte de l'effet des radiations cosmiques, les circuits FPGA, particulièrement ceux basée sur la mémoire statique, sont d'intérêt car ils sont sensibles au rayonnement cosmique (Souari, 2016).

S'assurer du respect des spécifications strictes dictées par les normes élevées de fiabilité de l'industrie automobile, demande des efforts supplémentaires au niveau de la vérification des systèmes électroniques embarqués, des efforts qui sont déployés tout au long du processus de conception. Pour ce qui est des pannes telles celles induites par les radiations cosmiques, les

efforts supplémentaires incluent, entre autres, l'utilisation d'une technique appelée injection des pannes (Felipe Augusto da Silva, Ahmet Cagri Bagbaba, Sandro Sartoni, Riccardo Cantoro, Matteo Sonza Reorda, Said Hamdioui et Christian Sauer, 2020). Cette technique consiste à perturber, de différentes manières, la valeur de nœuds logiques dans un système et à mesure l'impact de ces perturbations sur son fonctionnement. Elle est habituellement appliquée lorsque la représentation du système atteint le niveau des portes logiques, ce qui signifie que le processus de conception est assez avancé. S'il est découvert à ce moment-là que la réponse du système face aux perturbations n'est pas acceptable, les concepteurs doivent revenir en arrière et modifier le design afin de le rendre conforme. Dans ce contexte, il apparaît pertinent de faire cette évaluation plus tôt dans le processus de conception.

L'objectif principal de ce mémoire est de valider une stratégie d'injection de pannes applicable très tôt dans le processus de conception. Cette stratégie consiste à utiliser des modèles structuraux de pannes à haut niveau d'abstraction, dans des simulations Matlab/Simulink, afin de reproduire des résultats fautifs représentatifs de ceux obtenus sous rayonnement cosmique. Nous ciblons des fonctions typiquement utilisées par l'industrie automobile pour la détection des obstacles, à savoir le filtre de Sobel et la transformée de Hough. Pour effectuer cette validation, un montage expérimental a été développé. Ce montage utilise un PC sur lequel roule une simulation Matlab/Simulink, incluant les modèles structuraux de pannes, et une carte de développement centrée sur un circuit FPGA basé sur la mémoire statique, dans une approche de type matériel dans la boucle (Hardware-IN-the-Loop, HIL). Cette approche est également connue sous le nom de « Hardware In the Loop Simulation » (HILS). Pour des fins d'homogénéité, nous employons le vocable HIL au cours de ce mémoire. En plus d'être utilisé pour implémenter une partie du modèle de simulation, le FPGA contient un utilitaire permettant d'émuler la présence de perturbations produites par le rayonnement cosmique. Cette émulation, qui consiste à inverser la valeur de certains bits de mémoire statique du FPGA, est elle-même considérée comme représentative de l'impact des radiations cosmiques. Ce montage permet donc une comparaison entre l'effet des modèles structuraux de pannes à haut niveau et celui des radiations cosmiques, par l'intermédiaire de l'émulation. L'idée ici est de voir si l'utilisation des modèles à haut niveau d'abstraction, appliqués à des endroits

stratégiques, permet de générer des séquences au moins aussi erronées que celles produites par les émulations, auquel cas ces modèles et leur utilisation seront considérés comme validés.

Les apports principaux de ce mémoire sont présentés comme suit:

- la validation plus poussée d'une récente stratégie d'introduction de modèles structuraux de pannes de haut niveau empruntée des anciens travaux par mesure de couvrir soigneusement les séquences erronées des émulations. Cette contribution inclut le montage expérimental utilisé pour effectuer la validation.

- la modification de modèles de simulation Matlab/Simulink existants, contenant un filtre de Sobel et une transformée de Hough, utilisés pour effecteur cette validation.

Ce mémoire s'articule autour des quatre chapitres suivants :

Le chapitre 1 est intitulé « notions de base et revue de littérature ». Comme son nom l'indique, il contient une description des différentes notions : les sources des radiations et leurs effets sur les circuits intégrés (FPGA), les différentes applications touchant les véhicules autonomes, les différents modèles de pannes et une description des principales techniques et systèmes d'injection de pannes. Nous présentons, dans un dernier point, les travaux existants liés au développement des modèles de pannes à haut-niveau d'abstraction et à la couverture des pannes au niveau du domaine de l'automobile.

Le chapitre 2 décrit les différentes étapes de la méthodologie de notre projet.

Le chapitre 3 présente la stratégie d'insertion des modèles structuraux de pannes de haut niveau aux entrées sorties du modèle Matlab/Simulink et leurs simulations. Cette stratégie, issue de travaux de recherche antérieurs, est celle que nous désirons valider.

Le chapitre 4 s'intéresse à la comparaison des résultats fautifs générés par les modèles structuraux de haut-niveau et ceux générés par les émulations de pannes. Ces résultats sont comparés à la sortie du bloc ciblé par les différentes perturbations (filtre de Sobel) et ainsi qu'à la sortie du bloc en aval (transformée de Hough). Cette comparaison permet de valider l'hypothèse selon laquelle on peut couvrir les comportements fautifs des pannes par émulations avec celle des modèles structuraux de haut-niveau.

Enfin, une conclusion générale clôture ce mémoire avec une discussion des ouvertures futures.

CHAPITRE 1

NOTIONS DE BASE ET REVUE DE LITTÉRATURE

1.1 Introduction

L'existence de diverses catégories de particules causées par les radiations cosmiques, dans l'atmosphère, peut bouleverser le fonctionnement des circuits électroniques intégrés en générant des erreurs, principalement au niveau de la mémoire. C'est entre autres le cas des circuits intégrés FPGA (Field Programmable Gate Array) basés sur la mémoire statique (SRAM). Malgré leur sensibilité aux radiations cosmiques, ces circuits sont de plus en plus présents dans le domaine de l'avionique et dans les applications automobiles, en particulier les véhicules autonomes (Wei Kaijie, Koki Honda, et Hideharu Amano. 2018). Dans ce contexte étudier les effets de ces radiations sur ces circuits FPGA utilisés dans les applications automobiles embarqués s'avère primordial pour s'assurer de la robustesse de ces systèmes.

L'objectif de cette section est d'englober l'état de l'art des ouvrages de recherche relatifs aux radiations cosmiques: les motivations ainsi que l'impact des particules et des neutrons sur les circuits intégrés et en particulier les FPGA, le progrès et l'évolution du domaine de l'automobile tout en passant par les véhicules autonomes, les différents tests exercés pour une vérification de la fiabilité d'un design, les modèles de pannes avec les intérêts et les défauts de chaque modèle en termes de temps ou de ressources et enfin, les systèmes d'injection de pannes dont nous avons eu la charge d'emprunter.

1.2 Radiations cosmiques et circuits FPGA

1.2.1 Définition

L'expression radiations (ou rayons) cosmiques fait référence à un flux de particules se déplaçant à grande vitesse. On y retrouve entre autres des protons, photons, des électrons et d'ions lourds provenant de diverses sources (Robache, 2013). Les empreintes du rayonnement cosmique ont été discernées dans l'année 1912 pour la première fois (Souari, 2016).

1.2.2 Source des radiations cosmiques

Les sources des radiations cosmiques peuvent être classées en trois catégories (Siegle, Vladimirova, Ilstad, & Emam, 2015).

1. **Les rayons cosmiques galactiques :** ce sont des flux de particules de haute énergie d'origine provenant de l'extérieur du système solaire. Ils sont composés de protons, d'électrons et de noyaux d'atome totalement ionisés;

2. **Les ceintures de Van Allen (Figure 1.1):** elles sont présentes dans des altitudes allant de 100km à 65 000km. Elles sont constituées d'électrons, de protons et d'ions. Le champ magnétique terrestre dans ce cas n'est pas symétrique, ce qui cause des distorsions locales;

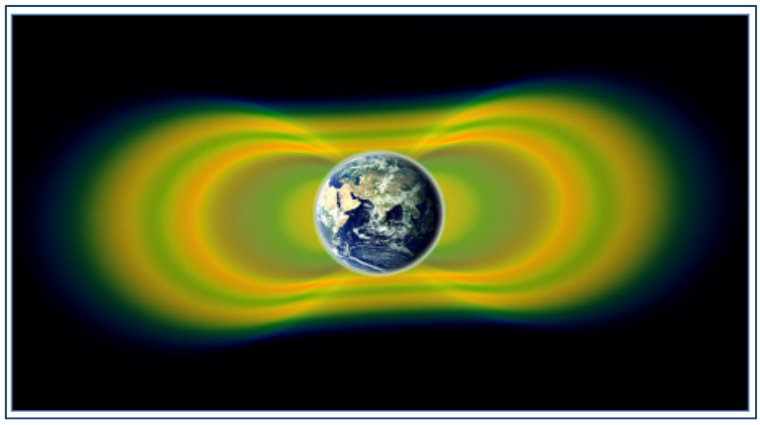

Figure 1.1 Ceinture de Van Allen, les zones de radiation étant en jaune
Tirée de NASA (20 février 2013)

3. **Le rayonnement solaire :** Il est associé à deux sources de radiation (Figure 1.2): le vent et les éruptions solaires. Le vent solaire est un flux de plasma de haute énergie émis par la

couche atmosphérique du soleil. Les éruptions solaires sont des petites rafales solaires de haute énergie qui peuvent atteindre la Terre et qui ne se produisent pas fréquemment. Les flux associés à ces deux phénomènes sont composés d'électrons, de protons et d'ions lourds;

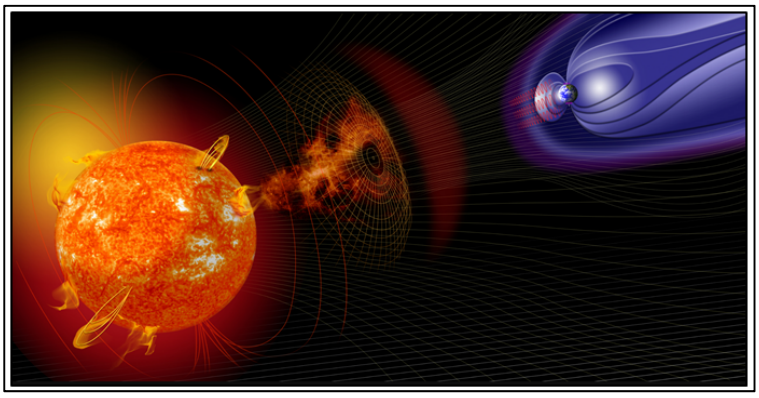

Figure 1.2 Rayonnement solaire
Tirée de NASA (2012)

1.2.3 Radiations secondaires

À l'approche de la Terre, les particules entrent en collision avec des atomes de gaz dans atmosphère, comme ceux de l'oxygène et de l'azote. À la suite de ces collisions, les atomes d'oxygène et d'azote se désintègrent en une variété de particules de haute énergie (Robache, 2013). La plupart de ces dernières se recombinent rapidement. Les neutrons constituent la principale proportion du produit de ces collisions atmosphériques. Ces neutrons ont tendance à ne pas se recombiner et voyagent à grande vitesse jusqu'à ce qu'une deuxième collision se produise; une telle seconde collision pourrait être avec la répartition physique de l'univers à savoir l'oxygène et l'azote, des objets voyageant dans l'atmosphère ou des objets à la surface de la Terre (Velazco, Fouillat et al. 2007).

Comme illustré à la Figure 1.3, au départ de l'atmosphère, deux types de réactions se produisent : la première réaction réside sur le fait que les particules s'ionisent en gaspillant une portion de leurs énergies et la deuxième présente la désintégration des rayons cosmiques en diverses particules. On présente ici « la douche (averses) des particules », des particules mineures qui produisent les neutrons, les protons, les ions lourds, les électrons, les muons et

les pions. La plupart de ces particules générées sont des neutrons où leur recombinaison est quasi impossible, elles sont provoquées par des percussions de rapidité extrême. Au niveau de l'espace, le degré d'énergie dépasse les centaines de mégaélectronvolts (Foucard, 2010).

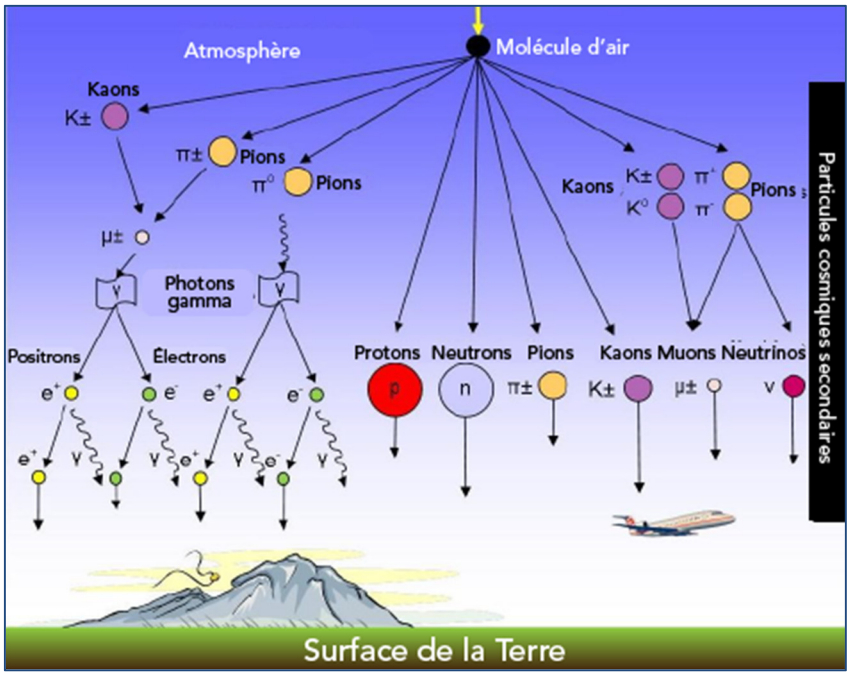

Figure 1.3 Phénomène de «la douche de particules »
Tirée de Parlons science (2019)

1.2.4 Effet des radiations sur les circuits électroniques

L'effet des radiations dépend de l'énergie, du flux des rayonnements incidents et du circuit utilisé. En effet, l'impact des radiations cosmiques sur les systèmes électroniques peut être classé en deux principales catégories : la dose d'ionisation totale (TID, Total Ionising Dose) et les événements singuliers, les SEE (Single Event Effect) (Siegle et al., 2015).

A. Dose d'ionisation totale : TID

La TID cause l'ionisation du composant électronique. C'est la collision des particules ionisantes telles que les protons avec le circuit intégré, résultant en une accumulation des

charges qui affectent le fonctionnement du composant (Siegle et al., 2015). Cette opération est induite par des photons et des électrons.

B. Les événements singuliers : SEE

Les SEE sont générés par la collision d'une particule avec le circuit intégré. Ces événements sont classés selon qu'ils sont destructifs ou non (Siegle et al., 2015).

SEE destructifs : Selon (Siegle et al., 2015), il existe un seul événement destructif : **SEL (Single Event Latchup) :** il s'agit d'un passage d'une particule unique qui induit la création d'un court-circuit parasite au niveau de la jonction PNPN des transistors, qui se traduit par un court-circuit entre l'alimentation et la masse. Cet évènement est suivi d'une augmentation de la consommation du courant, pouvant détruire le composant électronique. La détection d'un tel évènement exige la coupure d'alimentation afin de protéger le circuit. Une jonction PNPN est une structure qui agit comme un transistor PNP et un transistor NPN empilés les uns à côté des autres, formant un thryristor.
SEE non destructifs : nous distinguons trois types d'erreurs (Siegle et al., 2015) :

1. **SET (Single Event Transient) :** c'est un événement singulier qui est provoqué par une impulsion de courant qui se propage dans les circuits combinatoires et qui peut par la suite engendrer une erreur de type SEU;
2. **SEU (Single Event Upset) :** c'est une erreur qui se traduit par l'inversion d'un bit de mémoire ou d'un registre. Un SEU peut mener ou non à une erreur de type SEFI;
3. **SEFI (Single Event Functional Interrupt) :** c'est une erreur provoquant l'interruption du fonctionnement normal du circuit.

Dans ce projet, nous nous intéressons aux SEU, particulièrement ceux affectant la mémoire statique dans le domaine de l'automobile d'une manière à se rapprocher le plus possible de la réalité et de bénéficier d'un temps d'exécution plus court en implémentant le circuit sur un FPGA (Robache, 2013).

1.3 Aspects pertinents du domaine de l’automobile

1.3.1 Les véhicules autonomes

Ces dernières années, la conduite autonome est devenue un sujet assez populaire dans la communauté de la recherche ainsi que dans l'industrie, et même dans la presse (Shaoshan Liu, 2019).

Un véhicule autonome est un véhicule capable d'effectuer les mêmes actions que les conducteurs humains; ils n'ont pas besoin d’une intervention humaine ni supervision en aucune circonstance et ils n'ont même pas besoin d'un volant (Caroline Bianca S. T. Molina, Jorge Rady Almeida Jr., Lúcio F. Vismari, Rodrigo Ignacio R. González, Jamil K. Naufal Jr.et João Batista Camargo Jr, 2017).

En outre les véhicules autonomes peuvent offrir de nombreux avantages et plusieurs contributions dans notre quotidien. Le confort est un avantage évident, mais dans la société actuelle, les avantages pratiques des véhicules intelligents deviennent plus clairs chaque jour. Lorsque les routes sont encombrées, la productivité diminue, l'argent est gaspillé en carburant et une énorme perte en termes de temps est présente. En revanche, les véhicules autonomes coopératifs améliorent le flux de trafic, diminue le temps d’attente (Jamal Raiyn., 2018) et en ce qui touche la sécurité sur la route, les véhicules intelligents travaillent sur la réduction du nombre de blessures et de décès (environ 1,25 million de personnes meurent chaque année des suites d'accidents de la route selon l'Organisation mondiale de la santé (OMS)) (Pidurkar, Sadakale et Prakash. 2019).

1.3.2 Applications dans le domaine de l'automobile touchant les véhicules autonomes

1.3.2.1 Détection des bords pour les véhicules autonomes : Filtre de Sobel

Plusieurs chercheurs ont mis l'accent sur la détection des bords dans le domaine de l'automobile et surtout pour les véhicules autonomes (Preemon Rattanathammawat et Thanarat H. Chalidabhongse, 2006; Jin Zhao, Bingqian Xie et Xinming Huang, 2014; Lv Ning, Niu Shuyan et Li Xinran, 2011; Sheetal Israni et Swapnil Jain, 2016). En effet, cette méthode est omniprésente dans différentes applications en liaison avec les véhicules autonomes. Nous pouvons la trouver dans :

Détection des voies de la route : la première étape dans la détection des voies est la détection des contours suivie de la transformation de Hough pour la détection des lignes, ensuite le filtre de Kalman pour prédire la prochaine position des limites et des lignes de la route (Farid Bounini, Denis Gingras, Vincent Lapointe et Herve Pollart, 2015).

Détection des véhicules : la détection des voitures est basée sur la détection des plaques d'immatriculation ; pour la faire, 3 phases sont essentielles : une détection spéciale des bords, une localisation de la plaque d'immatriculation basée sur la distribution des pixels des contours et une suppression des fausses détections (Preemon Rattanathammawat et Thanarat H. Chalidabhongse, 2006).

Stationnement autonome : Avant de stationner, une détection des places vacantes est nécessaire. Pour la réaliser, 3 méthodes sont essentielles. La première méthode est basée sur la technique de détection de bord simple c'est-à-dire la détection des lignes séparant les places dédiées au stationnement. La deuxième technique est basée sur le comptage d'objets (compter les véhicules), tandis que la troisième méthode est basée sur la détection de premier plan et d'arrière-plan (Junzhao Liu, Mohamed Mohandes et Mohamed Deriche, 2013).

En 1959, la théorie de détection des bords a été étudiée par Julez et en 1963. L.G.Roberts a étudié l'algorithme de détection des contours et a annoncé l'algorithme de Robert. Sur cette base, d'autres algorithmes sont présentés comme le filtre de Sobel, l'opérateur de Prewitt et à la fin l'algorithme Laplacian (Yi Zhang, Xiaoyuan HAN, Han Zhang et Liming Zhao, 2017).

Dans ce mémoire, nous nous intéressons à la détection des contours et plus spécifiquement à un algorithme d'identifications des bords appelé 'Filtre de Sobel'. La première étape dans cet algorithme est de transformer une image colorée en une image RGB (Red, Green, Blue) puis calculer le gradient vertical et horizontal (Lv Ning, Niu Shuyan et Li Xinran, 2011). On utilise l'équation (1.1) pour le calcul du Gradient :

$$G=\sqrt{Sx^2+Sy^2} \tag{1.1}$$

Où Sx présente le gradient horizontal et Sy le gradient vertical.

Pour obtenir une détection de contour plus générale et pour un calcul simplifié du gradient nous pouvons récrire l'équation de la manière suivante (Lv Ning, Niu Shuyan et Li Xinran, 2011) :

$$G=|Sx|+|Sy| \tag{1.2}$$

1.3.2.2 Détection des voies de la route pour les véhicules autonomes : Transformée de Hough

La détection de voie est une fonctionnalité fondamentale permettant une conduite autonome en zone urbaine. Elle fournit des informations pour les applications en aval telles que le maintien de voie ou bien le changement de voie, et est généralement obtenue en détectant les marqueurs de voie correspondants sur la surface de la route (Ying Li et Sihao Ding, 2019).
Lorsque des véhicules autonomes se déplacent avec d'autres véhicules sur la route, il est essentiel de détecter la bonne voie et de suivre la route pour avoir une conduite intelligente et

sécuritaire sans aucun accident (Bandarage Shehani Sanketha Rathnayake et Lochandaka Ranathunga, 2018). Et pour la détection des lignes, les chercheurs ont souvent recours à la transformée de Hough, comme le cas pour « Springrobot » de (Qing Li et Nanning Zheng, 2003). C'est une ingénierie développée à l'origine pour reconnaître les lignes (Paul V. C. Hough, Ann Arbor, Mich. 1960), et a ensuite été généralisée pour couvrir des formes arbitraires.

A. Représentation des lignes dans l'espace de Hough

Une ligne dans le plan peut être représentée de différentes manières, par exemple sous l'équation 1.3.

$$y=a * x+b \tag{1.3}$$

Les lignes peuvent être représentées uniquement par deux paramètres. Souvent, la forme de l'équation 1.3 est utilisée avec les paramètres a et b.
Cette forme ne peut pas cependant représenter des lignes verticales. Par conséquent, la transformation de Hough utilise la forme de l'équation 1.4 (Ning Xin, Guo-hong Wang et Jing Zhang, 2006), qui peut être réécrite dans l'équation 1.5.

$$\rho = x \cdot \cos\theta + y \cdot \sin\theta \tag{1.4}$$

$$Y = -\cos\theta \sin\theta \cdot x + \rho \sin\theta \tag{1.5}$$

Les paramètres θ et ρ sont respectivement l'angle de la ligne et la distance de la ligne à l'origine. Toutes les lignes peuvent être représentées sous cette forme lorsque $\theta \in [0, 180[$ et $\rho \in R$ (ou $\theta \in [0, 360[$ et $\rho \geq 0$)).

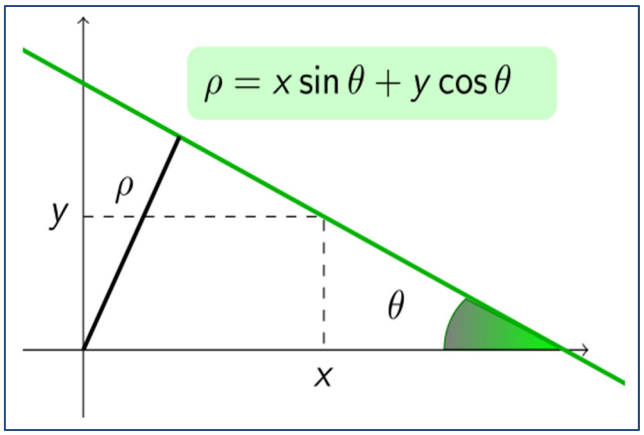

Figure 1.4 Représentation de l'équation de Hough dans un repère cartésien
Tirée de TSD Conseil, Digital signal processing

B. Algorithme de Hough

L'algorithme de la détection des lignes commence par détecter les contours 'Edge Detection' soit par le filtre de Canny, le filtre de Sobel, le filtre de Robert ou bien par le Laplacien. Après une détection des différentes lignes en se basant sur les équations de Hough déjà décrites suivie d'un bloc de maximum local sachant que le bord de la ligne est la position où la valeur maximale de la dérivée est détectée (Libiao Jiang, Jingxuan Li et Wandong Ai, 2019). Enfin un bloc pour concaténer les coordonnées des lignes détectées et un bloc pour schématiser les lignes sur l'image initial.

Pour récapituler on peut schématiser le fonctionnement de Hough comme suit (Figure 1.5) :

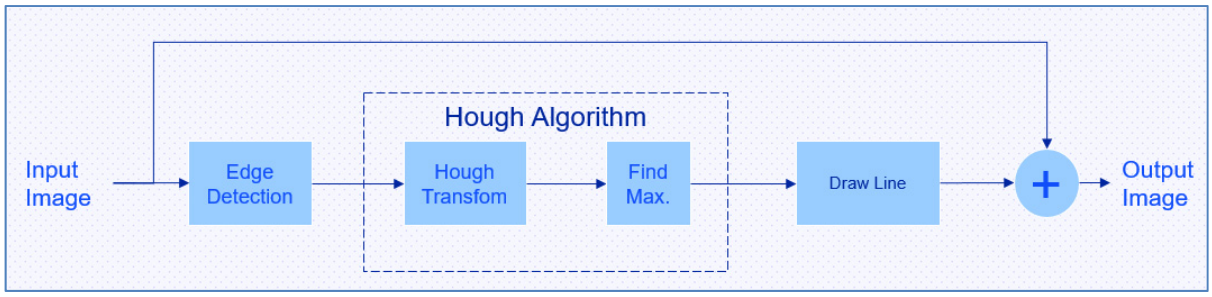

Figure 1.5 L'algorithme de la détection des lignes
Tirée de 2007 Texas Instruments Inc

1.4 Modèles de pannes

Un modèle de pannes est une abstraction de la réalité qui, au départ, représente les défectuosités pouvant se produire lors de la fabrication. Pour l'injection de pannes dans les modèles du circuit numérique, des modèles sont définis. Deux modèles de pannes, le modèle de délai et le modèle de collage (Borecky, Kohlik et al.2011) sont explorés dans cette section.

1.4.1 Modèle de délai

Un modèle de délai est utilisé pour représenter une hausse de temps de propagation dans un signal, menant à une violation des spécifications temporelles du circuit. Malgré qu'il ait été démontré que les radiations cosmiques pouvaient induire des délais supplémentaires dans les FPGA à mémoire statique (Quentin LAMBERT, 2019), ce modèle n'est pas considéré dans ce mémoire.

1.4.2 Modèle de collage

Le modèle (Stuck at fault : S@F) est un modèle de panne qui perturbe la valeur logique d'un signal. Lorsqu'un modèle collé à 1 ou 0 est appliqué à un signal, ce signal conserve cette valeur logique de manière permanente 1 (Wang et al., 2006). Sur la base de l'analyse faite par le professeur Thibeault (C. Thibeault, 2019), il a été décidé d'utiliser le modèle S@F parce qu'il représente le mieux ce qui se passe dans un circuit intégré de type FPGA.

1.5 Techniques et systèmes d'injection de pannes

La méthode d'injection de pannes fournit une prévision de la robustesse et de la fiabilité des composants face aux erreurs induites. Diverses méthodes sont mises en œuvre pour évaluer l'effet des radiations cosmiques sur les performances et le fonctionnement des circuits FPGA à base de SRAM (Siegle et al., 2015). Elles sont utilisées à différentes étapes du cycle de vie du circuit : de la conception jusqu'à la mise en fonction. Nous distinguons principalement trois catégories.

1. Approches physiques d'injection des pannes.
2. Injection des pannes par émulation.
3. Injection des pannes par simulation.

Dans ce qui suit, nous allons décrire les principales techniques, en commençant par celles appliquées sur les systèmes complets et en remontant à rebours le cycle de conception.

1.5.1 Approches physiques d'injection des pannes

A) Injection des pannes à travers les broches d'entrées/sorties

Cette méthode n'influe pas le mécanisme du circuit. Beaucoup d'instruments ont été déployés dans la simplification d'introduction des erreurs sur les nœuds. MESSALINE (Arlat, 1990) considéré le père des outils au sein de cette approche. L'erreur est introduite immédiatement sur l'une des broches du circuit intégré (à l'entrée ou à la sortie) dont laquelle une sonde soumet un niveau logique bas ou haut dans la broche sélectionnée.
À l'heure actuelle, tout en tenant en compte de la délicatesse des puces électriques, cette démarche est extrêmement restreinte et est employée uniquement dans le but d'éprouver les apparences de robustesse, en contrepartie cette approche n'est jamais employée pour introduire des pannes par émulation (Entrena et al., 2011).

B) Corruption de la mémoire

Cette démarche a été utilisée pour l'introduction de pannes au niveau de la mémoire des processeurs. Citons par exemple, DEFI (Michel et al., 1994), un injecteur de pannes commercial, qui a été employé dans ces différents types de tests. L'introduction des pannes par l'intermédiaire de la corruption de la mémoire est basée soit sur l'emploi de sondes, soit par l'intermédiaire de logiciels en insérant des bouts de code. Les mémoires et les registres sont les cibles par cette démarche d'injection des pannes.

C) Perturbation de l'alimentation

Cette méthode se base sur l'ajout des transistors MOS entre les deux bornes, la source d'alimentation et le nœud VCC dans le but d'introduire d'erreurs tout en jouant sur la tension de la grille. Cette démarche est semblable aux chutes de tension qui provoquent des pannes transitoires (Karlsson et al., 1991).

1.5.2 Injection des pannes par émulation

Cette approche consiste à ajouter des modules additionnels dans le circuit FPGA, permettant de contrôler la procédure d'injection de pannes (Ziade, Ayoubi et Velazco, 2004).
Au niveau de la démarche d'introduction des pannes par émulation à base des FPGA, on différentie essentiellement, selon (Khatri, Milde, Hayek, & Börcsök, 2014), l'injection des pannes par instrumentation et l'injection des pannes par reconfiguration. Dans ce qui suit, les deux approches sont présentées.

A) Technique d'injection de pannes par instrumentation

L'injection de pannes par instrumentation exige la modification du circuit original (Robache, 2013; Souari, 2016). L'instrumentation des bascules (Juneau, 2010) est l'une des démarches les plus connues des approches d'introduction des pannes par instrumentation. Elle se base sur l'ajout de logique permettant de former un ou des registres à décalage à partir des bascules existantes, dans le but de faciliter l'injection des pannes (David M et Wu, Intel, 2002).

B) Technique d'injection de pannes par reconfiguration

Ce type d'injection cible les bits de configuration du FPGA. Il est effectivement possible d'injecter des pannes en modifiant les bits de configuration (bitstream) du FPGA. Essentiellement, on trouve deux sortes de configurations des FPGA à base de mémoire SRAM, à savoir la configuration dynamique et la configuration statique. La première aide dans la

reconfiguration des FPGA complètement ou en partie pendant l'exécution de l'application. La deuxième incite à reconfigurer le FPGA à chaque fois où on veut apporter une modification (Souari, 2016) . La configuration dynamique offre un temps de reconfiguration plus court que celui offert par la reconfiguration statique (Souari, 2016). Diverses apparitions (Di Carlo et al., 2014; Ghaffari et al., 2014; Gosheblagh et Mohammadi, 2013;) emploient la reconfiguration dynamique partielle pour introduire des pannes.

1.5.3 Injection des pannes par simulation

Cette démarche se base sur l'injection des pannes aux différentes étapes de la conception des circuits électroniques et à différents niveaux d'abstraction (Vanhauwaert, 2008). Dans ce qui suit, nous distinguons les principaux niveaux d'abstraction couverts, du plus bas vers le plus haut : portes logiques, transfert de registre (RTL) et système.

A) Niveau des portes logiques

Il existe dans la littérature plusieurs outils permettant l'injection de pannes à ce niveau d'abstraction. Parmi ces outils, on peut citer **FAST** (Cha, Rudnick, Patel, Iyer, & Choi, 1996), **VERIFY** (Sieh, Tschache, & Balbach, 1997) , **Roban** (Vanhauwaert, 2008) et **VFIT** (Baraza, Gracia, Gil, & Gil, 2000). Différents travaux (Amaricai et al., 2014; Sootkaneung et Saluja, 2010) choisissent l'introduction des pannes au niveau des portes logiques pour juger la fiabilité de leurs designs. Cette méthode est plus représentative que celles utilisées aux niveaux d'abstraction plus élevés, mais elle est plus lente quant à l'application sur des circuits électroniques contemporains de grandes dimensions (Souari, 2016).

B) Niveau RTL

La littérature n'a pas laissé l'introduction d'erreur au niveau RTL inaperçue (Berrojo et al., 2002; Bombieri, Fummi et Guarnieri, 2011; Chen, Mishra et Kalita, 2012): cette injection de pannes s'effectue au moment de la simulation RTL suivant deux démarches: la première se

base sur le changement du schéma RTL avec l'ajout des saboteurs et la deuxième consiste à forcer des signaux internes à des grandeurs déterminées à l'aide des commandes du simulateur comme l'outil MAFISTO (Jenn, Arlat, Rimen, Ohlsson, & Karlsson, 1995).
L'atout de cette démarche est d'estimer, relativement tôt de la conception, la robustesse du design. Ses désavantages incluent une couverture plus faible des pannes et elle n'est pas représentative des pannes obtenues aux niveaux plus bas.

C) Niveau système

Plusieurs langages de programmation (SystemC, POLIS,...) ainsi que des outils (Matlab/Simulink) ont été utilisés pour la modélisation des circuits au niveau système (Bolchini, Pomante, Salice, & Sciuto, 2001; Robache, 2013). Ce haut niveau d'abstraction fournit une base permettant d'analyser la fiabilité du design, le plus tôt possible, lors de l'injection de pannes. Le but de cette approche est de limiter les risques liés à un manque de robustesse, et en permettant, le cas échéant, de modifier les choix des designs.

Dans ce projet, nous ciblons particulièrement l'outil de simulation Matlab/Simulink. C'est un outil très utilisé et qui présente une vitesse de simulation rapide. Il modélise les systèmes sous forme de blocs (entrées, sorties, bloc fonction, bloc constant, etc.), ce qui facilite la conception.

Nous travaillons également sur l'outil d'émulations par SEU controller. Cette méthode, est très proche de la réalité et rapide d'exécution. Pour une injection par SEU, l'outil détermine en premier l'adresse de la trame et en second la position du bit dans la trame. Afin d'injecter un pannes, l'outil lit la trame sélectionnée aléatoirement puis inverse le bit cible et enfin, la trame modifiée est écrite à la même adresse de la mémoire de configuration à partir de laquelle la trame originale est lue (Souari, 2016).

1.6 Travaux existants ciblant les effets des radiations cosmiques sur les circuits intégrés dans le domaine de l'automobile

Cette section présente les travaux les plus pertinents de la littérature liés au développement de modèles de pannes à haut niveau d'abstraction que nous allons emprunter par la suite et l'avancement des travaux dans la couverture des pannes dans le domaine de l'automobile.

1.6.1 Modèles structuraux de pannes de haut-niveau

Les modèles structuraux de pannes de haut niveau sont basés sur le modèle structural de pannes de bas-niveau S@F, présenté plutôt dans ce chapitre. Les modèles structuraux de pannes de haut niveau ont été développés par le professeur Claude Thibeault dans le cadre du projet EPICEA (C. Thibeault, 2019). La stratégie derrière ce développement était de se contenter si possible de pannes aux entrées et aux sorties des blocs Matlab/Simulink, afin d'éviter de modifier ces blocs et de réduire le nombre de pannes à injecter. Ces modèles ont été implémentés à travers de fonctions Matlab. Une première fonction Matlab est présentée à la Figure 1.6.

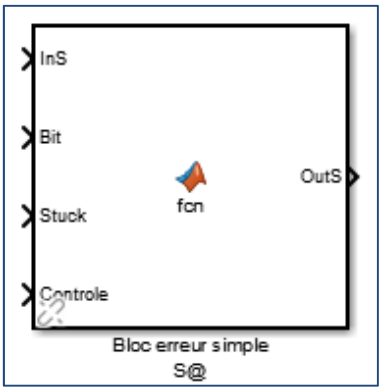

Figure 1.6 Fonction d'injection de pannes pour les circuits combinatoires
Tirée de Bilel Bahloul (2020)

Cette fonction a été développée pour injecter des pannes aux entrées des circuits combinatoires et séquentiels, ainsi qu'aux sorties des circuits combinatoires. Elle injecte des pannes de types collées à 0 ou à 1 sur les bits choisis d'un bus. Cette fonction d'injection de pannes s'avère toutefois insuffisante pour bien représenter le comportement fautifs de circuits séquentiels tel

que vu à leur sortie (C. Thibeault, 2019). En effet, plusieurs pannes affectant un circuit séquentiel, comme une machine à états finis, ont comme effet de le « geler » complètement, ce qui fait que sa sortie reste bloquée à la même valeur. Un saboteur plus complet, s'appuyant sur la première version et couvrant les circuits combinatoires et séquentiels, a été créé. Le nouveau saboteur est composé de trois blocs (Figure 1.7). Le premier est la fonction d'injection de pannes collée-à déjà présentée (Figure 1.6); sa sortie est sélectionnée quand le signal de sélection *Select* égal à 1. Le deuxième bloc (« CLK erreur ») est responsable de l'émulation du comportement fautif du signal d'horloge. Sa sortie est sélectionnée lorsque le signal *Select* est égal à 2. Cette émulation consiste à maintenir la valeur de sortie, lorsque l'entrée "Capture" de ce bloc est activée (dans ce cas "Capture"= 0), et ceci, quel que soit l'instant choisi. Le troisième bloc (« Reset erreur ») est pour l'émulation du comportement fautif du "reset" collé à 1 (i.e. toujours actif). Sa sortie, égale à 0, est sélectionnée lorsque le signal *Select* est égal à 3.

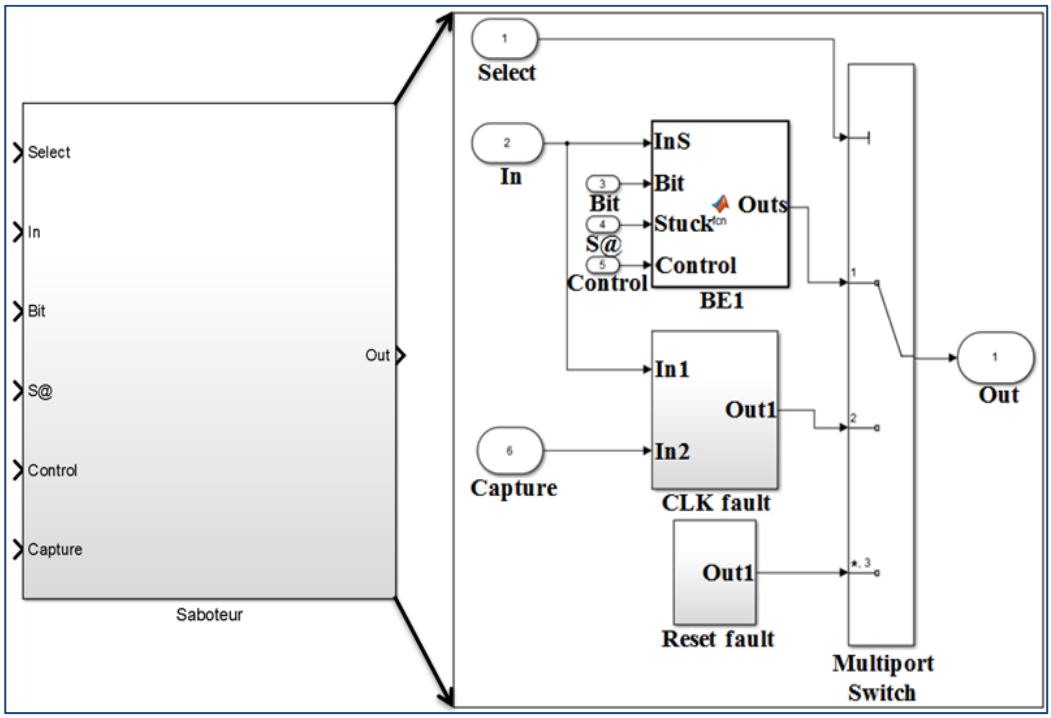

Figure 1.7 Saboteurs pour les circuits combinatoires et séquentiels
Tirée de Bilel Bahloul (2020)

1.6.2 Les circuits intégrés dans le domaine de l'automobile et l'injection de pannes

Le développement de circuits intégrés pour le secteur automobile pose des défis majeurs (Felipe Augusto da Silva, Ahmet Cagri Bagbaba, Sandro Sartoni, Riccardo Cantoro, Matteo Sonza Reorda, Said Hamdioui et Christian Sauer, 2020). La conformité au standard ISO26262, dans le cadre de ce processus, implique une analyse complexe pour l'évaluation des défectuosités matérielles aléatoires potentielles. Et la norme ISO26262 pour « Véhicules routiers - Sécurité fonctionnelle » est une norme émergente pour les systèmes de sécurité dans les véhicules routiers (Federico Pratas, Thomas Dedes, Andrew Webber, Majid Bemanian et Itai Yarom, 2018). C'est une norme multipartite qui définit les exigences et fournit des directives pour assurer la sécurité fonctionnelle des systèmes électriques/électroniques installés à bord des véhicules routiers.

En outre, la complexité croissante des applications automobiles entraîne un changement dans le flux de conception traditionnel. Un circuit intégré qui met en œuvre des applications critiques pour la sécurité, telles que la conduite autonome, doit intégrer des mécanismes pour réduire le risque de pannes entraînant des situations potentiellement mortelles. Pour de telles applications, le système doit être capable de détecter un pourcentage extrêmement élevé de problèmes potentiels alors qu'il est déjà déployé sur le terrain. Dans les circuits intégrés automobiles les plus avancés, où des millions de composants sont susceptibles d'être affectés par des défectuosités matérielles aléatoires, ce processus devient difficile (Felipe Augusto da Silva, Ahmet Cagri Bagbaba, Sandro Sartoni, Riccardo Cantoro, Matteo Sonza Reorda, Said Hamdioui et Christian Sauer, 2020).

Habituellement, des simulations d'injection de pannes « Fault Injection » (FI) sont déployées pour évaluer les effets de pannes en mode opérationnel. Cette méthode (FI) est une méthode de pointe pour la vérification de la sécurité fonctionnelle, recommandée par ISO26262. En tant que telle, plusieurs chercheurs ont exploré l'optimisation des campagnes d'injection de pannes (Felipe Augusto da Silva, Ahmet Cagri Bagbaba, Sandro Sartoni, Riccardo Cantoro, Matteo

Sonza Reorda, Said Hamdioui et Christian Sauer, 2020). Cette démarche est présente dans le domaine de l'automobile à différents endroits, tels :

***Motor Controller Unit* (MCU):** Selon la norme ISO26262, la technique d'injection de pannes est une méthode de vérification très utile dans les tests et la vérification des systèmes liés à la sécurité. En particulier, le MCU est qualifié comme critique par divers fournisseurs (Wang Bin, Ma Kai, Huang Xin, Ren Shan, Wang Fang et Shi Haotian, 2019).

CAN Controller : les chercheurs ont ciblé ce design représentatif des enjeux de l'industrie automobile. Pour cette raison, le périphérique adopté est une implémentation matérielle du contrôleur CAN SJA1000, développée par Philips au début des années 2000. Leur méthodologie, en combinaison avec la simulation de pannes, a été appliquée et a abouti à une couverture de 93%. (Felipe Augusto da Silva, Ahmet Cagri Bagbaba, Sandro Sartoni, Riccardo Cantoro, Matteo Sonza Reorda, Said Hamdioui et Christian Sauer, 2020)

Hybrid control unit (HCU) : les chercheurs adoptent le mode d'injection de pannes au niveau matériel. Les pannes sont introduites dans le HCU en cours d'exécution via une carte PCB d'injection de pannes matérielles pour tester le comportement du contrôleur en cas de pannes (Guan Jing, Zhou Yaling et Sun Hanwen, 2018).

Ces travaux présentent certaines similarités avec ceux de ce mémoire. Les éléments similaires incluent l'utilisation de modèles structuraux de pannes de haut-niveau, basés sur le modèle de bas niveau collé-à et l'adoption de la démarche d'injection de pannes pour la vérification du fonctionnement des applications critiques dans le domaine de l'automobile. Les différences fondamentales incluent : 1) le fait que les injections de pannes se fassent au niveau d'un modèle Matlab/Simulink en liaison avec le domaine de l'automobile, 2) le fait que les injections de pannes se fassent au haut-niveau, 3) le fait que les injections de pannes (FI) se fassent de deux manière par simulation et par émulation et 4) le fait que l'analyse des effets d'injections de pannes et de la qualité de la couverture de l'injection de pannes se fasse sur le bloc cible et le bloc en aval.

1.7 Conclusion

Ce chapitre a présenté une revue des notions et concepts importants pour la compréhension de ce projet. Cette revue inclut premièrement les radiations cosmiques et leur impact sur les circuits électroniques, plus particulièrement les circuits FPGA. Elle inclut également les aspects pertinents du domaine de l'automobile et les différentes applications dans ce secteur. On y retrouve aussi un résumé des principales techniques d'injection de pannes et des modèles de pannes, ainsi qu'une revue des travaux pertinents portant sur les modèles d'injection de pannes et leur utilisation dans le domaine de l'automobile. Le prochain chapitre porte une présentation de l'environnement de travail ainsi qu'une description des différentes étapes suivie tout au long de notre projet.

CHAPITRE 2

MÉTHODOLOGIE PROPOSÉE

2.1 Introduction

Au chapitre précédent, un aperçu de la littérature global touchant diverses approches d'injection de panne, des applications contribuant dans le domaine de l'automobile et des vérifications de sensibilité et de fiabilité des circuits intégrés ont été représentées. Dans ce chapitre, nous représentons l'environnement de notre projet, incluant la carte FPGA utilisée et la représentation des étapes suivies pour la réalisation de notre travail.

2.2 Calcul de la sensibilité des FPGA

Les FPGA sont de plus en plus utilisés pour les applications spatiales et terrestres en raison de leur grande densité, de leurs hautes performances, de leurs coûts de développement réduits et de la flexibilité de leurs programmations. En particulier, les FPGA basés sur la mémoire statique (SRAM) sont très utiles pour les missions à distance en raison de la possibilité d'être reprogrammé par l'utilisateur autant de fois que nécessaire dans un délai très court. Cependant, les FPGA basés sur la SRAM contiennent un grand nombre de cellules de mémoire qui sont très sensibles à la perturbation d'événement unique (SEU). Un SEU dans un FPGA basé sur SRAM peut entraîner des erreurs fonctionnelles, ce qui peut nécessiter la reconfiguration du FPGA (Rongsheng Zhang , Liyi Xiao , Member, IEEE, Jie Li , Xuebing Cao et Linzhe Li, 2020).

Même l'environnement terrestre est moins hostile que l'environnement spatial du point de vue des radiations cosmiques, les FPGA à base de SRAM sont quand même sujets aux SEU. Considérant que ces FPGA puissent être utilisés dans des fonctions critiques des véhicules automobiles (ex. conduite autonome), nous avons voulu vérifier si leur sensibilité aux SEU au

niveau du sol pour être une source de préoccupation pour la fiabilité de ces véhicules. Pour ce faire, nous avons fait le calcul de sensibilité pour différents circuits FPGA sur la base de la classification ASIL (Automotive System Integrity Level). Cette classification est imposée par la norme ISO26262, une norme qui traite la fiabilité de fonctionnement des systèmes électriques et électroniques incorporés au sein des véhicules routiers. Elle définit des niveaux de criticité et chaque situation dangereuse sera cotée selon un niveau d'exigence en matière de sécurité en passant du niveau ASIL A, B , C et D. Le niveau ASIL D est le niveau le plus critique (Ravindra Reddy Sabbella et Maheswaran Arunachalam, 2019) avec une valeur inférieure à 10 FIT (Failure In Time), ce qui signifie que le taux d'erreurs doit être inférieur à une erreur à toutes les 10 milliards d'heures.

Les résultats détaillés sont présentés à l'annexe I. En supposant que tous les bits de configuration sont des bits critiques, la grande majorité (75%) des circuits analysés n'atteignent même pas le niveau critique le moins élevé, i.e. le niveau ASIL A où le taux d'erreurs doit être inférieur à 1000 FIT. Il s'agit de l'hypothèse le plus pessimiste et l'atteinte de niveau ASIL D aurait signifié que l'effet des radiations cosmiques aurait pu être négligé.
Nous avons par la suite supposé des taux plus réalistes de bits critiques parmi les bits de configuration, à savoir 25, 20, 15 et 10%. Même à 10%, seulement 2% des circuits (provenant de la famille Spartan3, basée sur une « vieille » technologie CMOS 90nm) atteignent le niveau ASIL D, alors que seulement 22% atteignent le niveau ASIL C (taux d'erreurs < 100 FIT) et que 8% n'atteignent même pas le niveau ASIL A. Ces résultats clairement suggèrent que l'effet des radiations cosmiques sur les FPGA basés sur la SRAM ne peut être négligé dans des applications où un haut niveau de fiabilité doit être atteint, comme c'est le cas des systèmes embarqués dans les véhicules automobiles.

2.3 Flot de conception et niveaux d'abstractions

Dans l'intention d'éclaircir l'objectif de notre projet, il est essentiel d'expliquer les différentes phases du flot de conception d'un circuit ainsi que les niveaux d'abstractions. Les conceptions numériques sur des plateformes PLD (Programmable Logic Device) enchainent une démarche de conception qui peut être illustrée par une série de diverses étapes, parfois itératives, telles que précisées à la Figure 2.1.

Un ingénieur commence par la validation de l'algorithme de son circuit, puis son architecture, avant de produire la représentation physique de son circuit. Le déroulement de la conception numérique se divise sur plusieurs niveaux d'abstraction étalés du plus haut niveau (les spécifications du système étudié) au plus bas niveau (le circuit interne de la puce électronique utilisée). Dans la partie droite de la Figure 2.1, nous pouvons observer les différents niveaux d'abstraction suivis pendant le déploiement. Chacune de ces phases est effectuée avec différents outils et à différents niveaux. À haut niveau, le détail est abstrait, le modèle est théorique. À bas niveau d'abstraction, le modèle devient plus concret et détaillé.

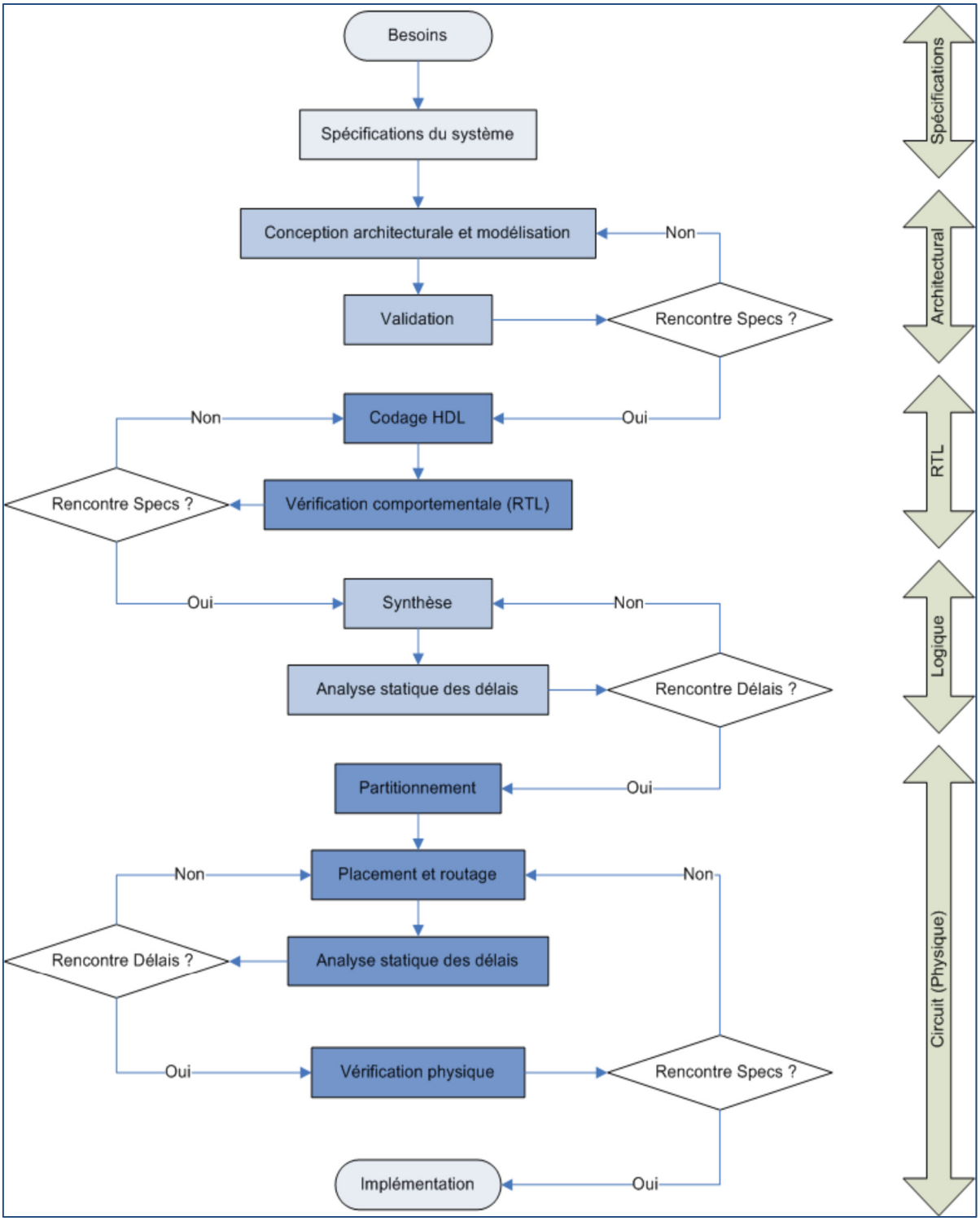

Figure 2.1 Diagramme du flux de conception d'un circuit et des niveaux d'abstractions
Tirée de Luc provencher (2010)

Dans ce projet de recherche, on vise le haut niveau d'abstraction, et nous introduisons des modèles d'injection de pannes semblables à celles provoquées par les rayons cosmiques, en ciblant tout particulièrement deux fonctions utilisées le domaine de l'automobile. Rappelons que ces fonctions sont le filtre de Sobel et la Transformée de Hough.

2.4 Les étapes de développement de ce projet

Afin de mener à bien cette réalisation et de répondre aux objectifs précédemment définis, les quatre étapes suivantes sont proposées : 1) la génération des signatures, collectées à la sortie du filtre de Sobel et du bloc en aval (Hough), et obtenues à l'aide de saboteurs placés sur les entrées/sorties du filtre de Sobel, 2) la synthèse d'un modèle Matlab/Simulink du filtre de Sobel dans une approche de type HIL (Hardware In the Loop), 3) la génération des signatures, collectées à la sortie du filtre de Sobel et du bloc en aval (Hough), obtenues à l'aide d'émulations de pannes ciblant le module synthétisé du filtre de Sobel dans un FPGA, et 4) une comparaison et une analyse des résultats. Le but ultime est de présenter une méthodologie globale pouvant confirmer la couverture des injections de pannes aléatoires par les saboteurs.

2.4.1 Étape 1 : Génération des Signatures des différents saboteurs

Cette phase commence par insérer les saboteurs hérités des travaux précédents (Bilel Bahloul, 2020) au niveau des ports d'entrées/sorties du modèle Matlab/Simulink « Sobel_HW » du filtre de Sobel et par la suite simuler le modèle tout en injectant des pannes, une pour chaque simulation. Afin de comparer l'information obtenue dans les rapports d'injection de pannes, le concept de signature a été utilisé, où une signature est simplement l'ensemble des signaux de sortie des blocs d'intérêt (Sobel, Hough). Dans ce projet, la première étape est de simuler au niveau Matlab/Simulink le bloc synthétisable avec les saboteurs à ses entrées/sorties et puis accumuler les signatures pour les comparer par la suite avec celles des émulations.

2.4.2 Étape 2 : Synthèse du Modèle Matlab par HIL

Dans ce projet, nous utilisons des simulations de type HIL, pour prendre avantage de la présence d'un FPGA dans la boucle, effectuer des émulations de pannes et de tirer profit de l'environnement Matlab/Simulink dans la collection et l'analyse des résultats de ces émulations pour effectuer la comparaison avec ceux obtenus avec les saboteurs.

Il a fallu au départ sélectionner la plateforme matérielle (carte) FPGA. À cette fin, une étude de différentes cartes Xilinx a été effectuée afin d'en trouver une ayant les caractéristiques suivantes :1) être disponible au Lacime, 2) présenter un environnement HIL complet fourni par Matlab, et 3) être compatible avec les outils déployés pour effectuer l'émulation de pannes.

Nous avons choisi à la fin la carte Nexys 4 centrée autour du FPGA Artix-7 XC7A100T-1-CSG324 qui satisfait les trois caractéristiques.

Pour la carte ayant été identifiée, il a fallu ensuite définir dans la partie commande de Matlab le chemin de Vivado pour permettre une synthèse de notre modèle avec Vivado Xilinx comme illustrée dans la (Figure 2.2).

```
>> hdlsetuptoolpath...
   ('ToolName','Xilinx Vivado','ToolPath','C:\Xilinx\Vivado\2015.4\bin')
Prepending following Xilinx Vivado path(s) to the system path:
C:\Xilinx\Vivado\2015.4\bin
>> hdlcoder_ip_core_sobel_video
```

Figure 2.2 Choix du Chemin « Path »

Une fois le chemin fixé, il a fallu indiquer la plateforme cible et l'outil de synthèse. Dans notre projet, la plateforme est « Generic Xilinx Plateform » et l'outil de synthèse n'est autre que « Xilinx Vivado ».

Finalement, le code VHDL du filtre a pu être généré par Matlab, concaténé par celui du module de l'émulation, synthétisé et implémenté sur le FPGA.

2.4.3 Étape 3 : Génération des Signatures des différentes émulations

Après avoir réalisé le montage HIL, nous perturbons le fonctionnement de notre carte NEXYS4 par des injections de pannes aléatoires avec l'outil « Injector7 » (décrit dans le chapitre 4 dans la section 4.2.1) tout en utilisant le module SEU (Chapman, 2010a; 2010b), développé par Xilinx. Ce dernier présente deux fonctionnalités principales:

1. La correction des erreurs de configuration causées par des SEU dès qu'elles sont détectées en effectuant un balayage de la mémoire de configuration, ce qui permet de détecter et corriger les SEU; cette fonctionnalité n'est pas utilisée dans ce projet.

2. L'émulation des SEU dans les FPGA Artix-7 en injectant des pannes dans la mémoire de configuration, ce qui permet par exemple d'observer le comportement des designs face aux événements singuliers.

L'émulation des SEU a été implémentée d'une façon à ressembler le plus possible le modèle de panne existant dans la réalité. L'injection des pannes au niveau de la mémoire de configuration du FPGA peut être aléatoire ou déterministe. Dans notre projet, nous travaillons avec des injections aléatoires. Pour une injection aléatoire avec SEU, l'outil Injector7 détermine les adresses de mémoire visées à l'aide de deux générateurs de nombres pseudo-aléatoires : le premier détermine l'adresse de la trame et l'autre la position du bit dans la même trame. Afin d'injecter un SEU, le module SEM identifie la trame prise d'une façon aléatoire puis permute le bit objet et à la fin, la trame réformée est libellée à la même adresse de la mémoire de configuration à partir de laquelle la trame initiale est lue.

Comme mentionné précédemment, l'injection de pannes par émulation se fait au niveau du bloc synthétisable « Sobel_HW » du filtre de Sobel et nous regardons à ce niveau non seulement les signatures à la sortie du filtre, mais aussi les signatures à la sortie du bloc en aval, celui de la transformée de Hough pour les analyser dans la phase suivante.

2.4.4 Étape 4 : Une comparaison et une analyse de la couverture

Une comparaison des différentes signatures est faite par l'intermédiaire d'un code « Python » et une vérification de la couverture des différentes émulations est effectuée. Nous parlons d'une analyse de la couverture qui consiste en premier lieu à comparer les séquences de sortie du circuit défectueux dues aux injections de pannes d'une façon aléatoire par l'outil « Injector7 » à celles dues aux injections de pannes sur les ports externes au niveau de filtre de Sobel. En second plan, une deuxième analyse de la couverture pour conclure que les saboteurs couvrent aussi bien les blocs en aval.

2.5 Conclusion

Dans ce chapitre, une description des quatre étapes de notre méthodologie a été présentée. Les différentes phases requises pour l'implémentation ont été détaillées. Leur fonctionnement a été également décrit, et implémenté dans l'environnement Matlab/Simulink. Les résultats de simulation obtenus à l'aide du montage déployé avec les saboteurs seront présentés au chapitre 3. Les résultats de l'émulation des pannes seront présentés et comparés avec ceux des saboteurs au chapitre 4.

CHAPITRE 3

SIMULATION DES MODÈLES STRUCTURAUX DE PANNES DE HAUT-NIVEAU DANS LES PORTS D'ENTREES SORTIES

3.1 Introduction

Une description détaillée des modèles Matlab/Simulink à étudier, à savoir le filtre de Sobel et la transformée de Hough, est fournie dans ce chapitre. Ces blocs sont choisis parmi les ressources génériques de Matlab. Nous nous intéressons par la suite à l'ajout des modèles structuraux de pannes de haut niveau hérités des travaux précédents aux niveaux des différentes entrées/sorties de filtre de Sobel et nous simulons les modèles (nous utilisons un seul saboteur dans chaque simulation) et enfin, nous enregistrons les résultats obtenus pour les vérifier par la suite.

3.2 Présentation détaillée de l'application : détection des bords

Dans ce projet de recherche, un modèle omniprésent dans le domaine de l'automobile et qui touche en premier lieu les véhicules autonomes a été choisi. C'est la détection des contours en utilisant le filtre de Sobel.

3.2.1 Modèle Matlab : Sobel Edge Detection

Comme il a été mentionné dans le premier chapitre, la détection des contours et des bords est l'une des importantes applications dans le domaine de l'automobile. Conséquemment, un modèle Matlab existant appelé « Sobel Edge Detection » sera employé (Figure 3.1).
Ce modèle est utilisé lors de simulations avec injection de pannes à l'aide des saboteurs qui seront placés aux différentes bornes d'entrée/sortie du bloc « Sobel_HW » afin de mesurer l'impact de ces pannes.

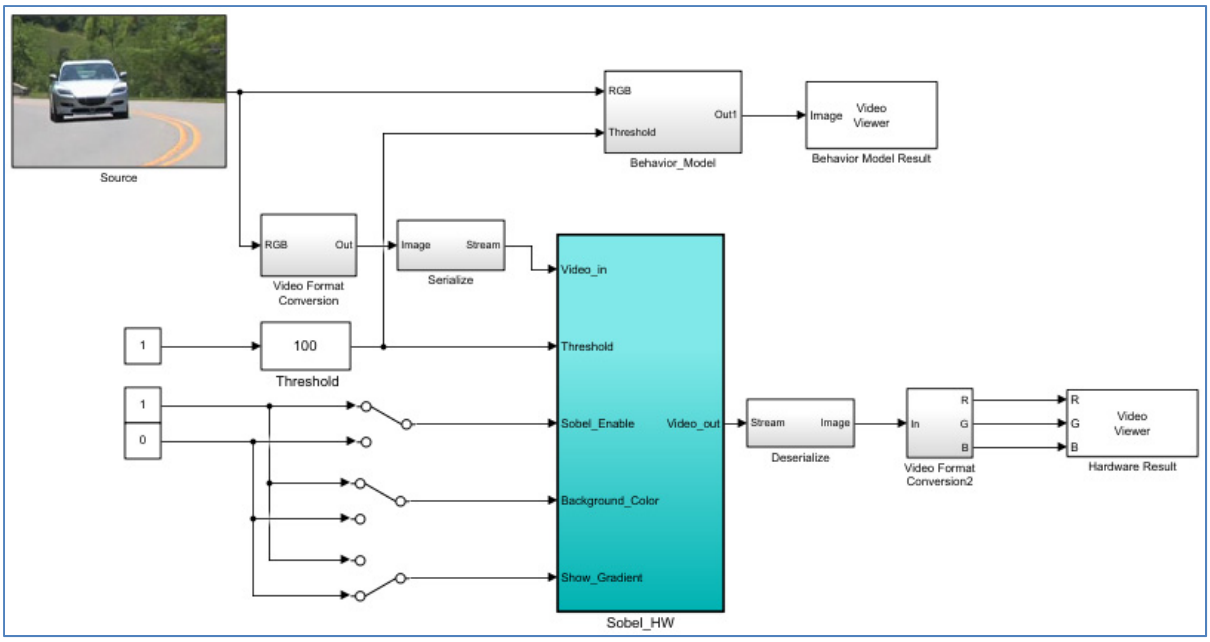

Figure 3.1 Modèle 1 : « Sobel Edge Detection » sans pannes
Tirée des exemples de MathWorks Matlab

À l'entrée du modèle Matlab/Simulink « Sobel Edge Detection », une image de dimension [105x 160] est présente pour être traitée par la suite. La première étape et de convertir les pixels de notre image RGB (Rouge, Vert et Bleu) sous format 32'hFFRRBBGG, via le bloc « Video Format Conversion ».

La deuxième phase est de transformer la matrice des pixels de l'image en un vecteur de données de 16800 valeurs en passant par le bloc « Serialize ». Ce vecteur est par la suite fourni au bloc « Sobel_HW », qui comprend cinq entrées et une seule sortie. La copie de référence (sans pannes) comporte une entrée de 32 bits 'Video_in', une deuxième de 8 bits 'Threshold', une troisième, une quatrième et une cinquième de 1 bit représentant 'Sobel_Enable', 'Background_Color' et 'Show_Gradient' respectivement, et une sortie de 32 bits 'Video_out'. Notons que ce bloc « Sobel_HW » est synthétisable, ce qui va permettre le déploiement du matériel en boucle (« Hardware-In-The-Loop ») afin d'effectuer des émulations de pannes. Les résultats de ces émulations seront présentés ci-dessous et comparés avec les résultats obtenus avec les saboteurs.

Les fonctionnalités des entrées du bloc « Sobel_HW » sont décrites ci-dessous :

1. La variable **Sobel Enable** est la variable responsable de l'activation du filtre. Si elle est égale à 1, le filtre de Sobel est activé et la sortie 'Video_out' présente l'image filtrée. Si elle est égale à 0, le filtre de Sobel est désactivé et la sortie 'Video_out' prend la valeur de l'entrée 'Video_in'.

2. La variable **Background_Color** n'est qu'une variable qui permet de faire apparaitre les couleurs de l'arrière-plan. Si la variable est égale à 1 donc l'arrière-plan de l'image est noir par la suite, la sortie 'Video_out' prend la valeur 1 pour une représentation des bords en blanc.

3. La variable **Show_Gradient** n'est qu'une variable de contrôle qui permet d'afficher le gradient. Si elle est égale à 1, la sortie 'Video_out' prend la valeur du gradient.

4. L'entrée **Threshold** représente le seuil. Lorsque la valeur de seuil est égale à une valeur donnée bien définie, il est évident que les bords nets et précis sont détectés. Le Filtre ignore tous les bords dont leurs valeurs sont supérieures au valeur seuil (B.Poornima, Y.Ramadevi et T.Sridevi , 2011)

Après le filtrage de l'image, une phase de transformation vecteur à matrice (bloc «Deserialize») est nécessaire pour obtenir une matrice de pixel d'une image et par la suite appliquer une séparation des couleurs rouge, vert et bleu.

3.2.2 Explication du fonctionnement du bloc synthétisable « Sobel_HW »

Le bloc synthétisable « Sobel_HW » est composé de trois sous blocs et d'un sélecteur (« switcher ») comme illustré dans la figure 3.2.

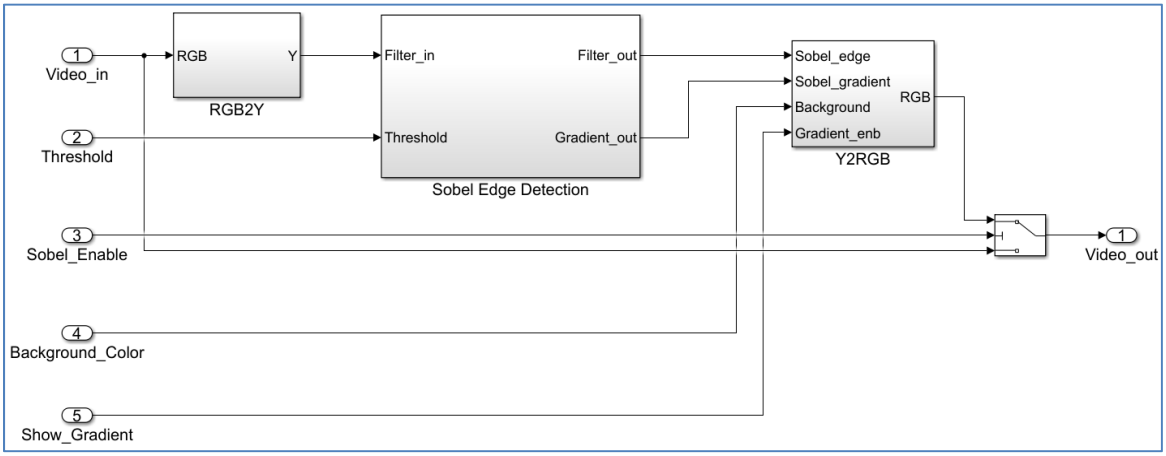

Figure 3.2 Représentation du bloc synthétisable « Sobel_HW »
Tirée des exemples de MathWorks Matlab

Le premier bloc RGB2Y est un bloc qui transforme les valeurs rouge, verte et bleue d'une image RGB en valeurs de luminance (Y) d'une image YCbCr (figure 3.3) dans le but de créer une bande passante réduite pour les composants de luminance, permettant ainsi de masquer plus efficacement les erreurs de transmission par la perception humaine. La luminance des couleurs est déterminée par la formule suivante (Somchart Chokchaitam, Phakdee Sukpornsawan, Nutcha Pungpiboon et Saowalak Tharawut, 2019) :

$$Y = \frac{\big((\text{Red value} * 299) + (\text{Green value} * 587) + (\text{Blue value} * 114)\big)}{1000} \qquad (3.1)$$

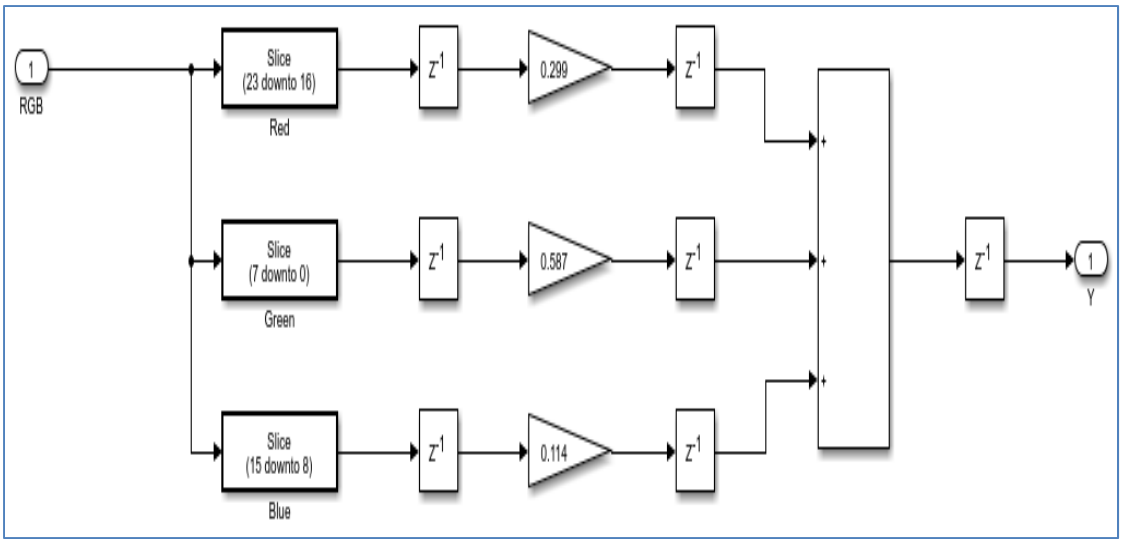

Figure 3.3 Conversion de l'image RGB to Y
Tirée des exemples de MathWorks Matlab

Le deuxième bloc appelé « Sobel Edge Detection » correspond au bloc qui calcule les deux gradients vertical et horizontal. Pour ce faire, faudra suivre les matrices des masques (figure 3.4).

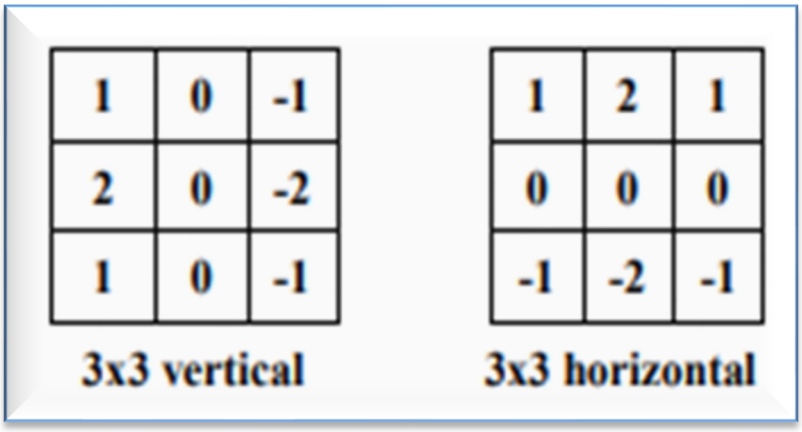

Figure 3.4 Masques des gradients
Tirée de Lv Ning, Niu Shuyan et Li Xinran (2011)

Pour le calcul des gradients, ce modèle Matlab/Simulink est subdivisé en deux parties : Verticale et Horizontale. X-filter représente le calcul du gradient vertical dans la figure 3.5.

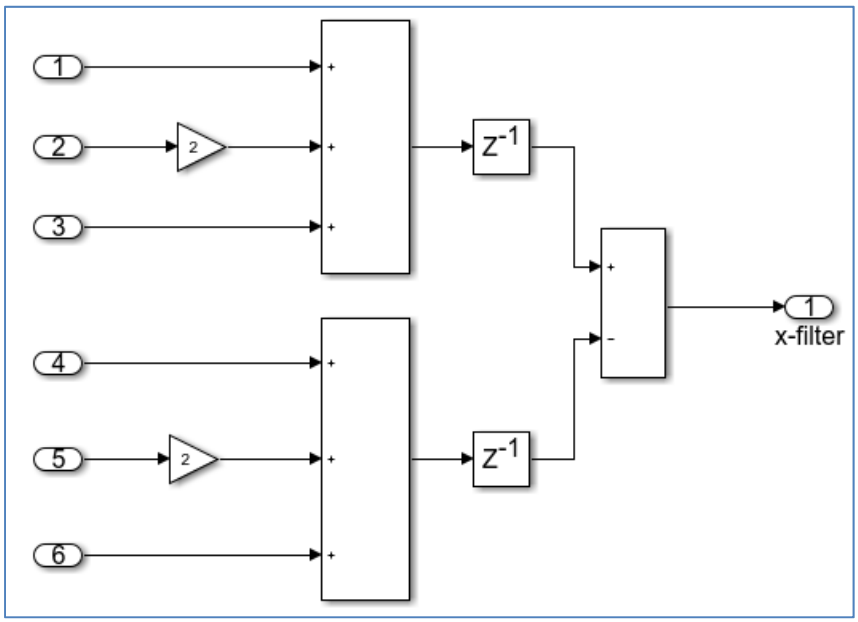

Figure 3.5 Calcul du gradient vertical
Tirée des exemples de MathWorks Matlab

Et Y-filter représente le calcul du gradient Horizontal dans la figure 3.6.

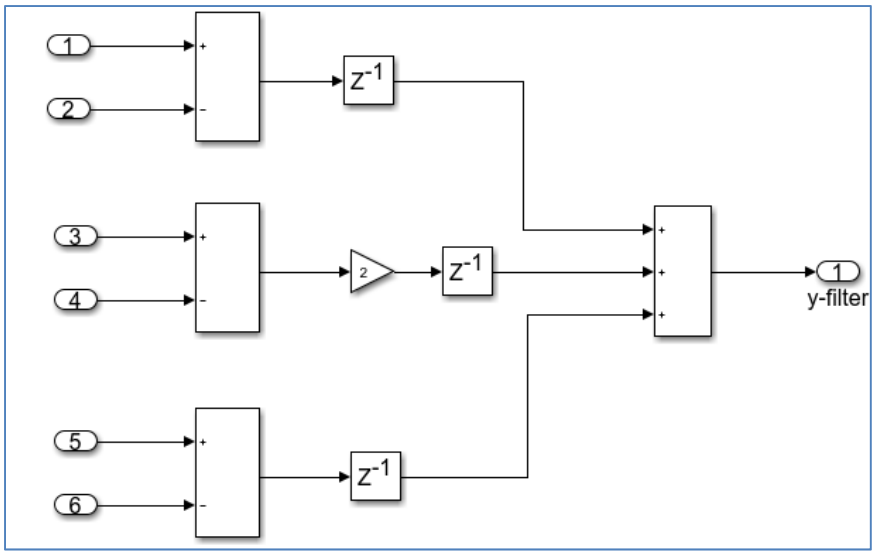

Figure 3.6 Calcul du gradient horizontal
Tirée des exemples de MathWorks Matlab

Le dernier bloc « Y2RGB », est le bloc responsable de visualiser les bords (les bords sont en blanc avec un arrière-plan noir). Pour représenter les bords, la fonction Matlab suivante est exécutée:

Algorithme 3.1 Algorithme de la fonction du bloc « Y2RGB »

```
Fonction RGB_out

Si Show_Gradient = 1
  Intensity <= fi (sobel_gradient, 0, 8, 0); % Intensity prend la valeur du gradient
déjà calculé
   RGB_out <= bitconcat (white, intensity, intensity, intensity); % la sortie prend la
valeur du gradient
Sinon
        Si background_color =1 % le background est noir
             Si sobel_edge % le Sobel _edge est activé
                RGB_out <= RGB_white; % les bords sont en blancs
             Sinon
                RGB_out <= RGB_black; % les bords sont en noirs
        Sinon % background_color = 0, background est blanc
             Si sobel_edge % le Sobel _edge est activé
                RGB_out <= RGB_black; % les bords sont en noirs
             Sinon
                RGB_out <= RGB_white; % les bords sont en blancs
```

Pour terminer, le sélecteur nous permet de choisir la sortie du bloc, c'est-à-dire de choisir entre l'image filtrée et l'image non-filtrée.

3.3 Présentation détaillée de la deuxième application : détection des voies de la route

3.3.1 Modèle Matlab : ex_vision_detect_lines

Comme pour le modèle « Sobel Edge Detection », une copie d'un deuxième modèle est utilisée (Figure 3.7) afin de détecter les voies de la route pour les véhicules autonomes, à savoir c'est une copie de référence originale (sans modifications).

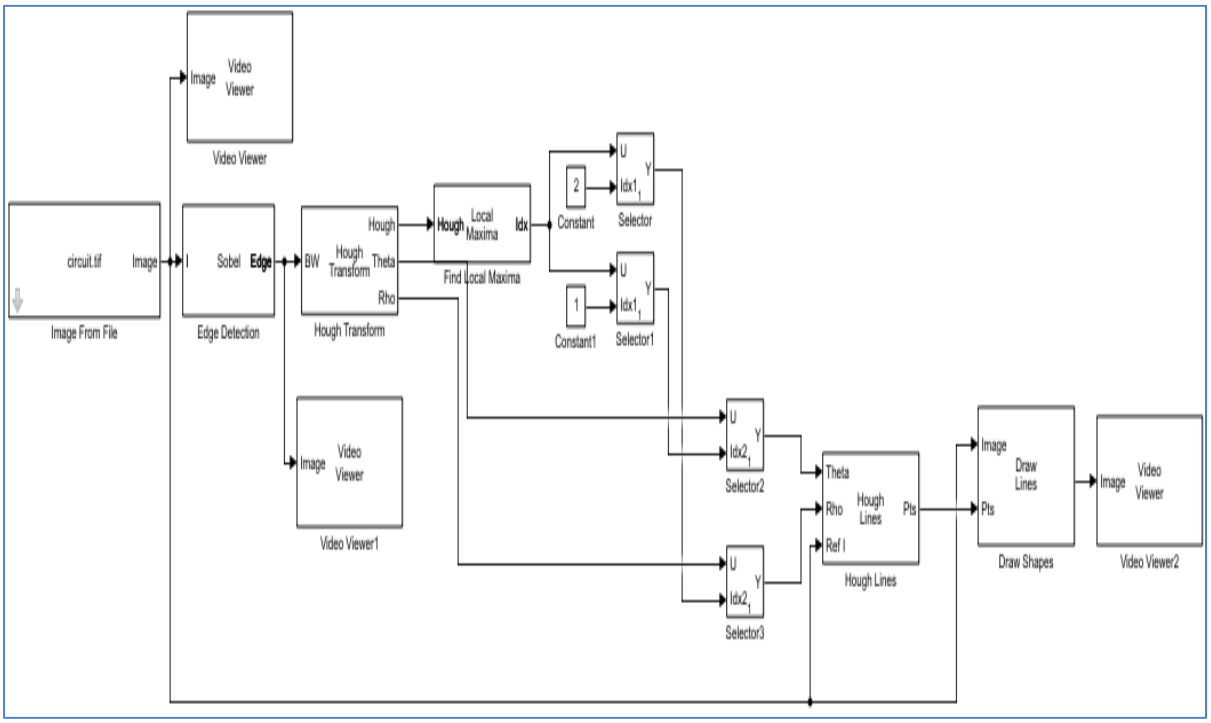

Figure 3.7 Modèle 2 : « ex_vision_detect_lines » sans modifications
Tirée des exemples de MathWorks Matlab

Ce modèle Matlab est constitué des blocs suivants :

1. Un bloc représentant un fichier d'image sous format d'image RGB utilisé comme entrée.
2. Un bloc 'Edge Detection' représentant le filtre de Sobel pour la détection des contours.
3. Un bloc 'Hough Transform' pour la détection des différentes lignes.
4. Un bloc 'Find Local Maxima' identifiant la position avec la valeur maximale de la dérivée, ce qui corrrespond à la bordure (Libiao Jiang, Jingxuan Li et Wandong Ai, 2019).
5. Un bloc 'Hough Lines ' pour concaténer les coordonnées Thêta et Rho des lignes détectées.
6. Un bloc 'Draw Shapes' pour schématiser les lignes sur l'image initiale.

3.3.2 Modification du Modèle Matlab : ex_vision_detect_lines

Une modification est apportée au niveau du modèle 1 « Sobel Edge Detection », qui consiste à ajouter à ce dernier la partie en aval du filtre de Sobel du modèle 2 « ex_vision_detect_lines » comme illustrée dans la figure 3.8. On commence par ajouter le bloc de la transformée de Hough pour le calcul de Rho et Thêta, suivi d'un bloc pour la détection d'un maximum local et un bloc pour la concaténation des deux valeurs Rho et Thêta et enfin la schématisation des lignes détectées.

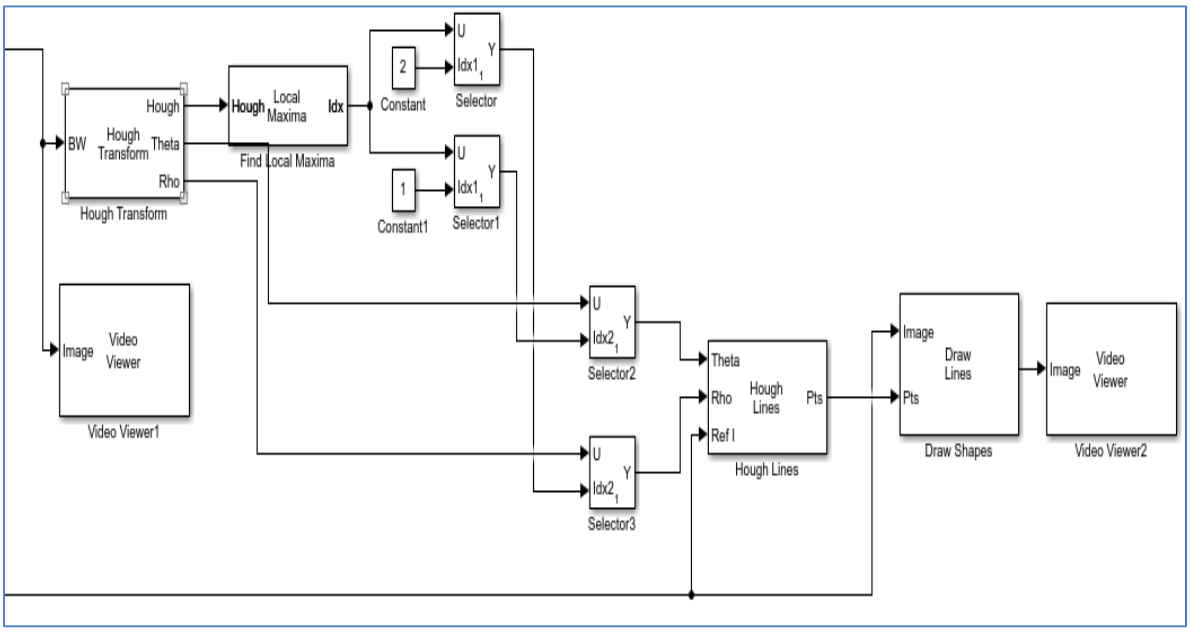

Figure 3.8 La partie ajoutée au modèle 1 « Sobel Edge Detection »
Tirée du Modèle 2 « ex_vision_detect_lines »

Finalement, le Modèle 3 (Figure 3.9) est obtenu en combinant les figures 3.1 et 3.8, où la partie ajoutée (Figure 3.8) est branchée à la sortie du bloc « Deserialize » (Figure 3.1).

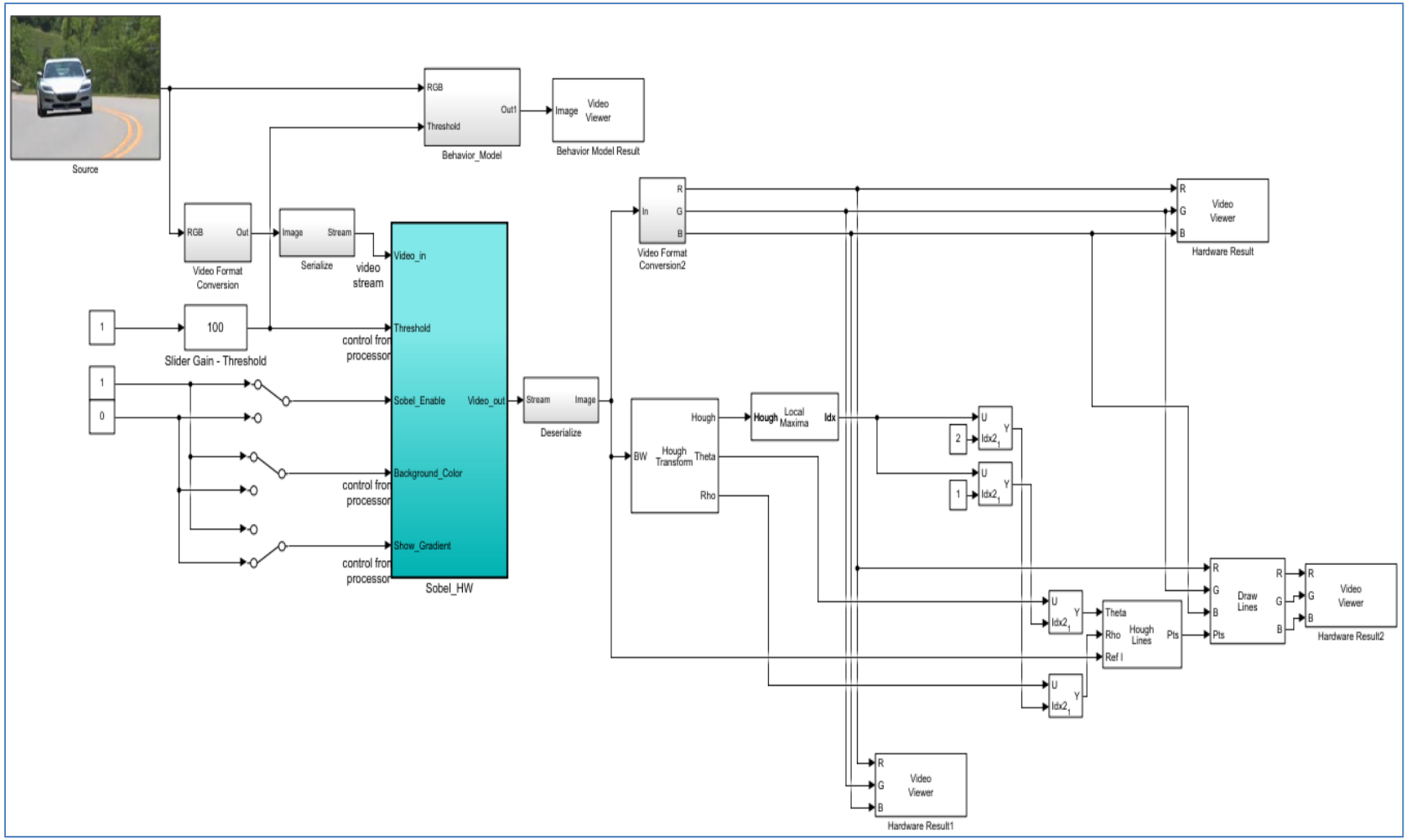

Figure 3.9 Modèle 3: Ajout de la partie de la détection des voies de la route au modèle 1

Ci-dessous trois figures (Figure 3.10, Figure 3.11 et Figure 3.12) représentant les trois étapes du projet. L'image en haut n'est autre que l'image initiale (une image choisie ou les lignes de la route sont bien visibles) suivi de l'image après détection des bords et enfin une image représentante de la détection des voies.

Figure 3.10 Image initiale

Figure 3.11 Détection des bords (Modèle 1)

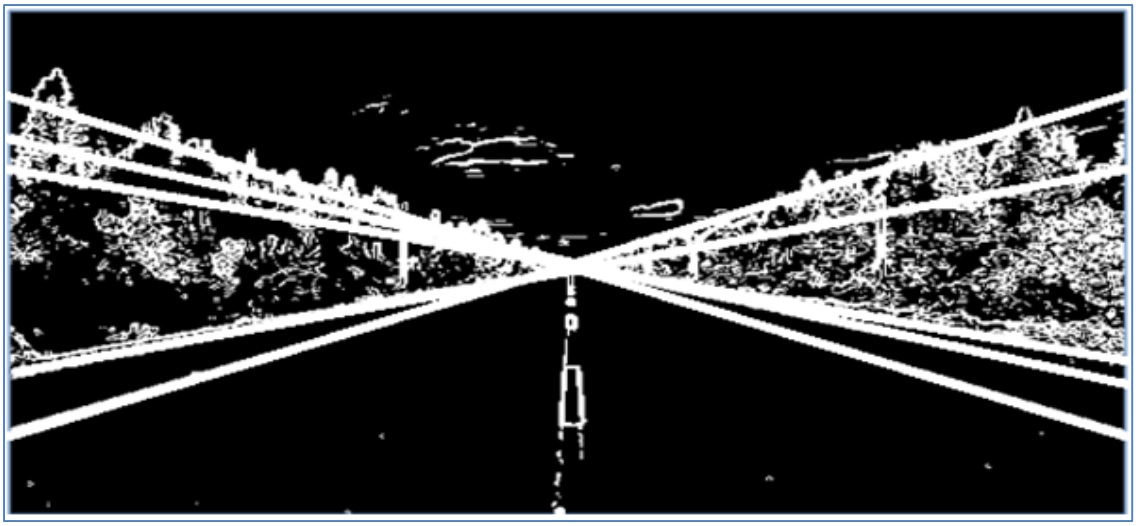

Figure 3.12 Détection des voies de la route (Modèle 3)

3.3.3 Étude théorique du changement des dimensions :

Comme mentionné dans la section précédente 3.3.2 le modèle « ex_vision_detect_lines » a connu des changements, ce qui a entrainé des modifications dans les dimensions de l'image à la sortie du filtre et de la transformée de Hough. Dans cette section une étude théorique complète permet d'expliquer le changement des dimensions à la sortie de la transformée de Hough.

Au niveau du filtre de Sobel les dimensions du signal de sortie sont de [105x 160]. À la sortie du bloc en aval les dimensions changent et deviennent [381x180].

180 : présente l'angle maximal : la valeur maximale de thêta est de Pi/180.
381 : présente la valeur maximale de Rho.

Pour déterminer la plage des valeurs de Rho, cette démarche s'impose : Définir l'équation de la transformée de Hough.

$$Rho=x*\cos(theta) + y*\sin(theta) \quad (4.5)$$

Et chercher à chaque reprise les valeurs de Xmin, Xmax,Ymin et Ymax comme illustré dans la figure 3.13.

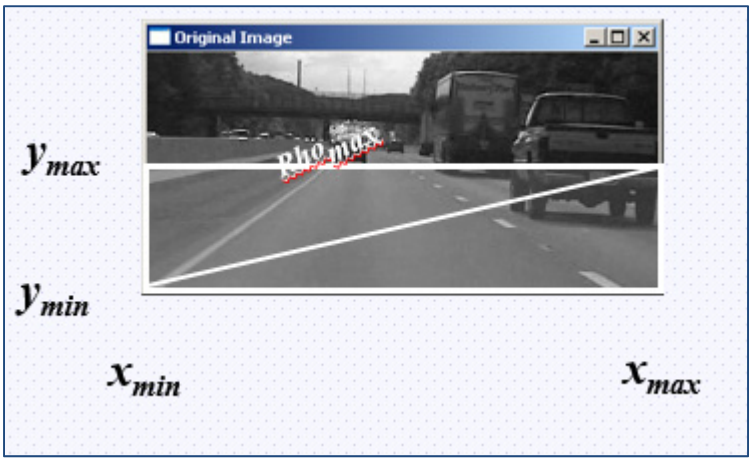

Figure 3.13 Représentation des coordonnées max et min

L'équation suivante permet de déterminer la valeur de $Rho_{\max}$:

$$\left.\begin{matrix} x_{\min} \le x \le x_{\max} \\ y_{\min} \le y \le y_{\max} \end{matrix}\right\} \Rightarrow Rho_{\max} = \sqrt{(x_{\max} - x_{\min})^2 + (y_{\max} - y_{\min})^2} \Rightarrow \begin{cases} -Rho_{\max} \le Rho \le Rho_{\max} \\ 0 \le \theta \le 180° \end{cases}$$

Dans ce Cas

$x_{\min} = 1$ $\quad$ $y_{\min} = 1$

$x_{\max} = 105$ $\quad$ $y_{\max} = 160$

Par la suite : $Rho_{\max} = \sqrt{(105-1)^2 + (160-1)^2} = 190 \Rightarrow \begin{cases} -190 \le Rho \le 190 \\ 0 \le \theta \le 180° \end{cases}$

Donc Rho est dans la plage -190 à 190 soit 381 valeurs entières (en passant par 0). C'est un passage ici d'un repère cartésien à un autre polaire.

3.4 Ajout des modèles structuraux de pannes de haut-niveau et simulation d'un nouveau modèle cible

L'objectif de cette section est de présenter les modèles structuraux de pannes à haut niveau, les saboteurs, utilisés dans ce projet de recherche. L'injection de pannes sera effectuée, une seule panne à la fois, sur les ports externes des circuits (entrées, sorties), grâce à l'insertion de saboteurs sur ces nœuds, qui sont placés manuellement.

3.4.1 Les modèles structuraux de pannes à haut-niveau : les saboteurs existants

Comme il a été mentionné au chapitre de la revue des travaux existants, les modèles structuraux de pannes de haut niveau ont été développés par le professeur Claude Thibeault dans le cadre du projet EPICEA (C. Thibeault, 2019). Dans ce projet, ces modèles de saboteurs combinatoires et séquentiels seront utilisés pour les introduire au niveau des bornes d'entrées sortie de notre modèle synthétisable « Sobel_HW ».

La stratégie derrière ce développement était de voir s'il était possible de se contenter de cibler les pannes aux entrées et aux sorties des modèles Matlab/Simulink, afin d'éviter de modifier ces blocs et de réduire le nombre de pannes à injecter.
Pour les circuits combinatoires, les modèles structuraux de pannes de haut niveau consistent simplement en l'application des modèles collés-à simples (i.e. une seule panne à la fois) dans Matlab/Simulink. Ces modèles ont été implémentés à travers d'une fonction Matlab (Figure 3.14). Cette fonction Matlab comporte 4 entrées (*d2, c, i, e*) et une sortie (*f2*). L'entrée *d2* est un bus de données contenant un certain nombre de bits, correspondant à la cible des modèles de pannes. La sortie *f2* est un bus de données de même largeur que d2. L'entrée *c*, une fois activée (i.e. *c*=1), met la fonction en mode d'injection de panne, si non *f2= d2*. L'entrée *e*

permet de choisir le type de pannes (i.e. aucune, s@0, s@1). L'entrée *i* permet de choisir lequel des bits de *d2* qui sera affecté par la panne, s'il y a lieu (*i*=1 correspondant au LSB). Si aucune panne n'est appliquée, *f2= d2*.

d2 : nœud d'entrée à saboter.

c : contrôleur; si c=1 ; injection de panne sinon pas d'injection.

i : numéro du bit sur lequel on va injecter la panne.

e : type de panne ; collé à 1 ou à 0.

f2 : la sortie de notre modèle, i.e. le nœud saboté.

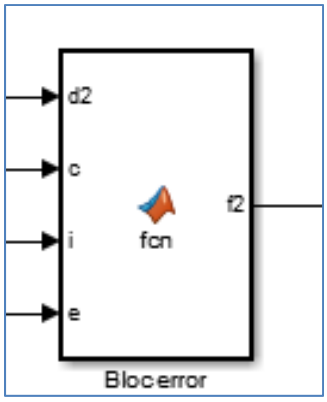

Figure 3.14 Le saboteur combinatoire emprunté

Pour la sortie du 'Sobel_HW', le deuxième type des saboteurs, les saboteurs séquentiels, est utilisé (Figure 3.15). Cette deuxième version contient six entrées (Select, In, Bit, S@, Contrôle, Capture) et une sortie (Out). Le signal « Select » permet de choisir le type de pannes à injecter. Le signal « In » est connecté au nœud cible (bit ou bus de plusieurs bits). Le saboteur séquentiel est composé de trois blocs. Le premier est la fonction d'injection de pannes collée- à déjà présentée (saboteur combinatoire, Figure 3.14); sa sortie est sélectionnée (i.e. connectée au signal de sortie du saboteur, « f2 ») lorsque le signal qui contrôle le multiplexeur (Select), est égal à 1. Le deuxième bloc (« CLK Erreur ») est responsable de l'émulation du comportement fautif du signal d'horloge. Sa sortie est sélectionnée lorsque le signal « Select » est égal à 2, ce

qui bloque le signal de sortie à sa dernière valeur avant l'application de la panne. Le troisième bloc (« Reset erreur ») est pour l'émulation du comportement fautif du "reset" collé à 1 (i.e. toujours actif). Sa sortie, égale à 0, est sélectionnée lorsque le signal « Select » est égal à 3.

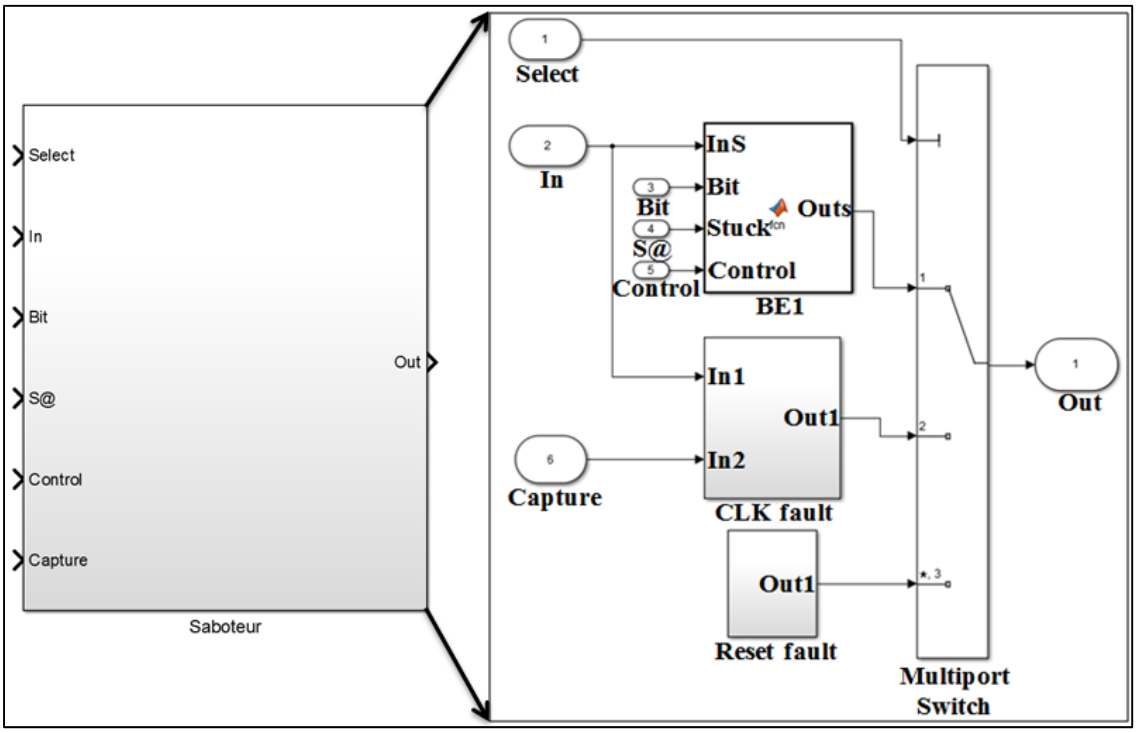

Figure 3.15 Saboteur séquentiel emprunté

3.4.2 Injection de pannes sur les nœuds externes

Dans le but de détecter les valeurs fautives du modèle « Sobel Edge Detection » subissant l'injection de pannes, un nouveau modèle est créé en ajoutant des saboteurs (Figure 3.16) au modèle d'origine. Pour le bloc cible « Sobel_HW », six saboteurs (E_{in}, E_T, E_{se}, E_B, E_{sg} et E_{out}) sont insérés, un saboteur combinatoire pour chacune des cinq entrées (E_{in}, E_T, E_{se}, E_B et E_{sg}) et un saboteur séquentiel pour sa sortie (E_{out}).

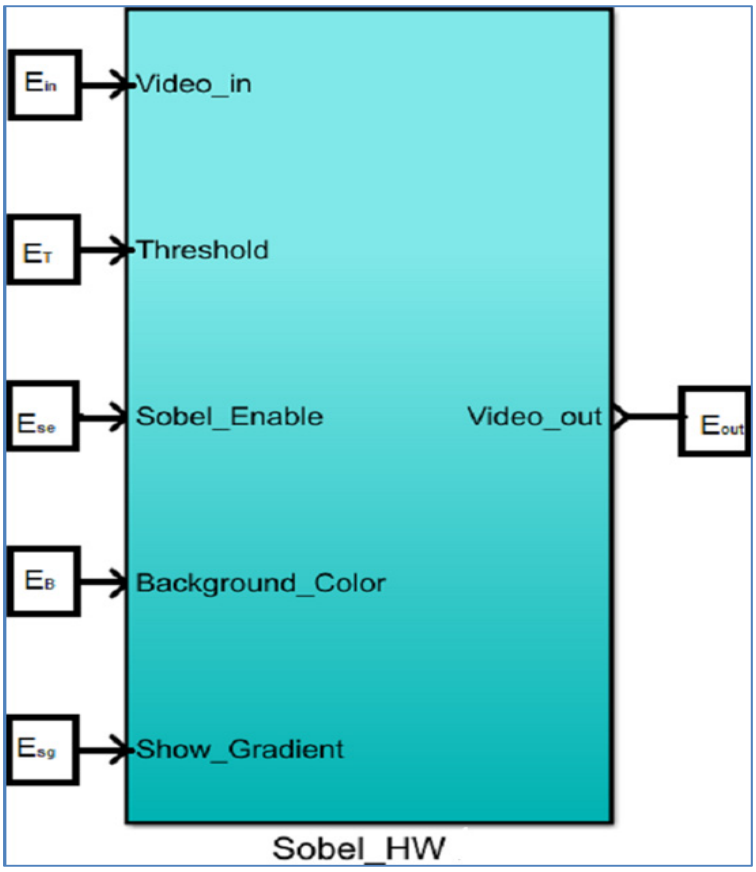

Figure 3.16 Localisation des saboteurs sur le modèle cible

3.4.3 Résultats des simulations de l'injection de pannes au niveau du bloc « Sobel_HW »

Le comportement fautif du filtre de Sobel est évalué par l'injection des pannes aux niveaux des différentes entrées et de la sortie. Dans ce qui suit quelques exemples sont présentés.

3.4.3.1 Injection de pannes au niveau de Video_in

L`entrée Video_in est sur 32 bits, donc il faut 64 injections en mode S@0 et S@1 pour couvrir l'ensemble des possibilités. Les figures 3.17 et 3.18 illustrent le comportement fautif dans le

cas d'injection de pannes 'E_{in}' sur le bit 8 de « Video_in », pour le mode Stuck at 0. D'autres exemples de reste des séquences sont fournis à l'annexe II.

La figure droite est le résultat d'injection de pannes sur Video_in 8 S@0 et celle de gauche est le résultat de la simulation sans panne.

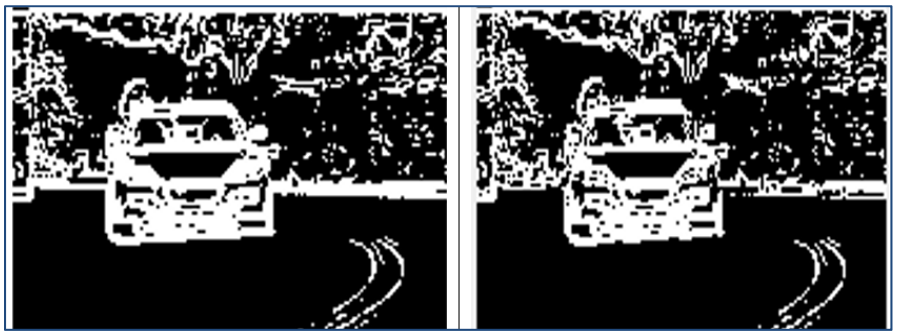

Figure 3.17 Résultats d'injection de pannes sur Video_in 8 S@0 (figure droite) et résultat de la simulation de référence sans panne (figure gauche)

Les deux résultats (sans pannes et avec panne) paraissent semblables à l'œil nu. Pour vérifier la différence entre les 2 images, l'histogramme de la différence est présenté. C'est un histogramme basé sur l'équation: Pixel avec Pannes – Pixel sans Pannes (pixel par pixel).
Dans ce qui suit une représentation de l'histogramme de la simulation sans panne à gauche et une représentation de l'histogramme de la différence de l'entrée Video_in= 8 S@ 0 à droite.

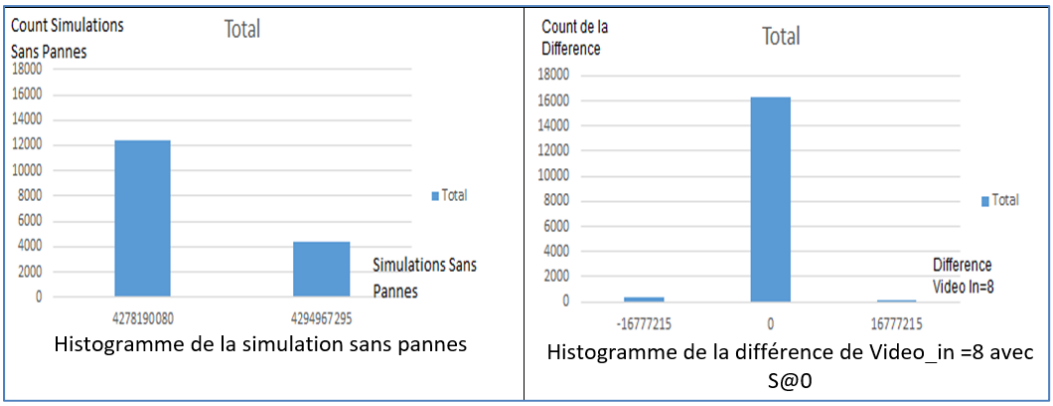

Figure 3.18 Histogrammes de la différence d'injection de pannes (figure droite) et Histogramme de la simulation de référence sans panne (figure gauche)

L'histogramme de la différence de Video_in 8 S@0 présente 16310 valeurs identiques et 490 valeurs différentes. Ceci montre que l'injection de pannes au niveau de Video-in 8 provoque un changement de valeurs sur 490 pixels.

3.4.3.2 Injection de pannes sur Threshold

Les comportements fautifs générés lors de l'injection de pannes 'ET' sur le bit 8 de l'entrée « Threshold », pour le mode Stuck at 1sont illustrés dans les Figures 3.19 et 3.20.
L'injection de pannes 'ET' au niveau de l'entrée "Threshold" résulte une génération de 16 séquences (l'entrée Threshold est sur 8 bits) représentant le comportement fautif du filtre. D'autres exemples sont fournis dans l'annexe III.

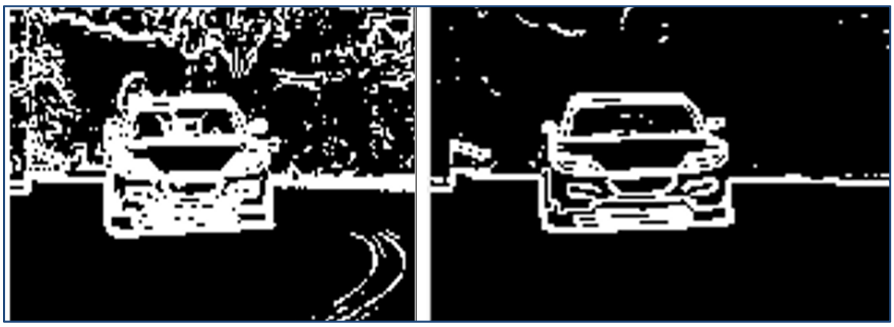

Figure 3.19 Résultats d'injection de pannes sur Threshold 8 S@1 (figure droite) et résultat de la simulation de référence sans panne (figure gauche)

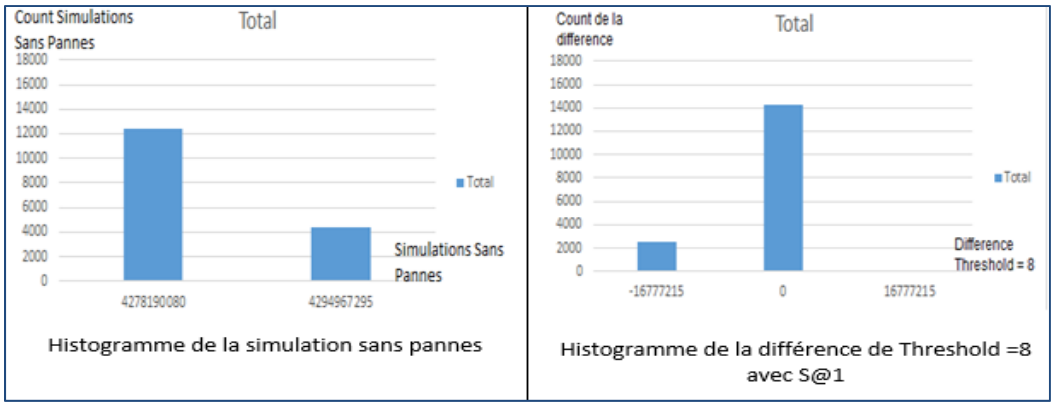

Figure 3.20 Histogrammes de la différence d'injection de pannes (figure droite) et Histogramme de la simulation de référence sans panne (figure gauche)

L'histogramme de la différence de Threshold 8 S@1 présente 14287 valeurs identiques et 2514 valeurs différentes. Donc on a 2514 pixels différents du cas sans pannes.

3.4.3.3 Injection de pannes sur Sobel_Enable

La présence du mode Stuck at 0 sur l'entrée "Sobel_Enable" désactive le fonctionnement du filtre et fige la sortie sur la dernière valeur générée avant l'apparition de la panne, comme détaillé sur la Figure 3.21, où la sortie du filtre correspond à l'image originale lorsque la panne est appliquée.

D'autre part, le comportement fautif du bloc filtre de Sobel à la suite de l'injection de panne 'Ese' en mode Stuck at 0 au niveau de l'entrée "Sobel_Enable" est schématisé par l'histogramme de la différence de la Figure 3.22.

La simulation a montré également que l'injection de pannes sur l'entrée "Sobel_Enable" en mode S@1, est sans effet sur le fonctionnement du filtre, ce qui coïncide avec l'algorithme représenté dans la section 3.2.2.

Figure 3.21 Résultats d'injection de pannes sur Sobel_Enable S@0 (figure droite) et résultat de la simulation de référence sans panne (figure gauche)

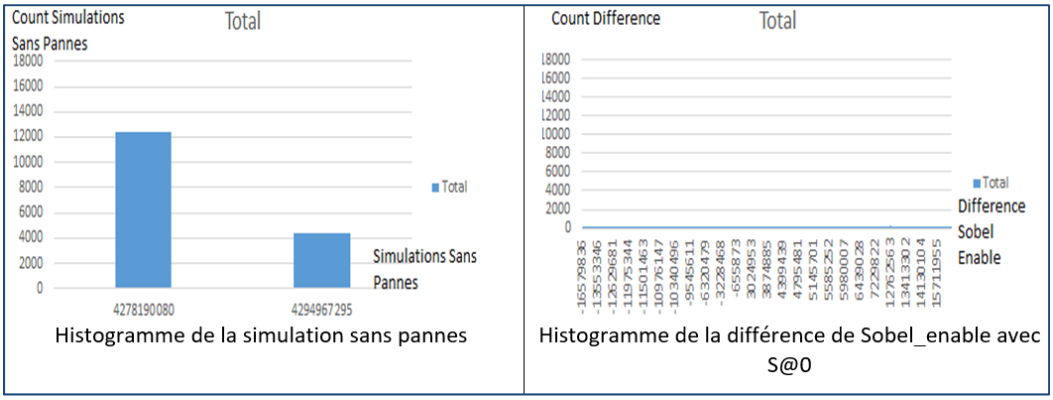

Figure 3.22 Histogrammes de la différence d'injection de pannes (figure droite) et Histogramme de la simulation de référence sans panne (figure gauche)

Il s'agit ici d'une plage de valeurs de pixels (les pixels de l'image colorée) et non pas de deux valeurs (les valeurs du noir = 4278190080 et blanc= 4294967295).

3.4.3.4 Injection de pannes sur Background_color

Le comportement fautif du filtre de Sobel suite à l'injection de panne 'EB' en mode Stuck at 0 sur l'entrée "Background_color" est schématisé par la Figure 3.23 suivie d'un histogramme de la différence Figure 3.24. La simulation a montré également que l'injection de pannes sur l'entrée "Background_color" en mode S@1 est sans effet sur le fonctionnement du filtre, ce qui coïncide avec l'algorithme représenté dans la section 3.2.2.

Figure 3.23 Résultats d'injection de pannes sur Background S@0 (figure droite) et résultat de la simulation de référence sans panne (figure gauche)

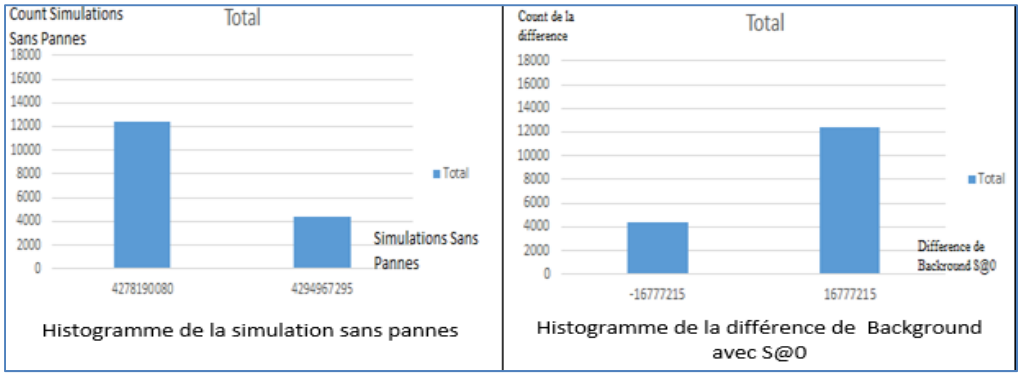

Figure 3.24 Histogrammes de la différence d'injection de pannes (figure droite) et Histogramme de la simulation de référence sans panne (figure gauche)

L'histogramme de la différence de Background S@0 présente l'inverse du comportement de sans pannes. C'est à dire avec un Background forcé à 0 (un arrière-plan en blanc) le contour est toujours en noir (l'inverse du cas sans pannes).

3.4.3.5 Injection de pannes sur Show_gradient

Le comportement fautif du filtre de sobel à la suite de l'injection de panne 'E_{sg}' en mode Stuck at 1 sur le "Show_gradient" est schématisé par les Figures 3.25 et 3.26. La simulation a montré également que l'injection de pannes sur l'entrée " Show_gradient" en mode S@0 est sans effet

sur le fonctionnement du filtre, ce qui coïncide avec l'algorithme représenté dans la section 3.2.2.

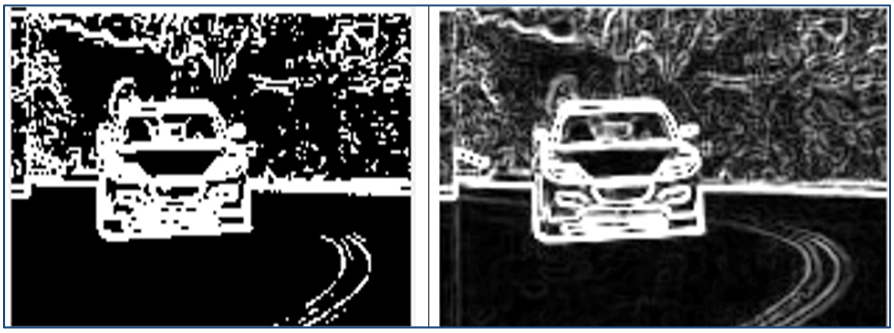

Figure 3.25 Résultats d'injection de pannes sur Show_gardient S@1 (figure droite) et résultat de la simulation de référence sans panne (figure gauche)

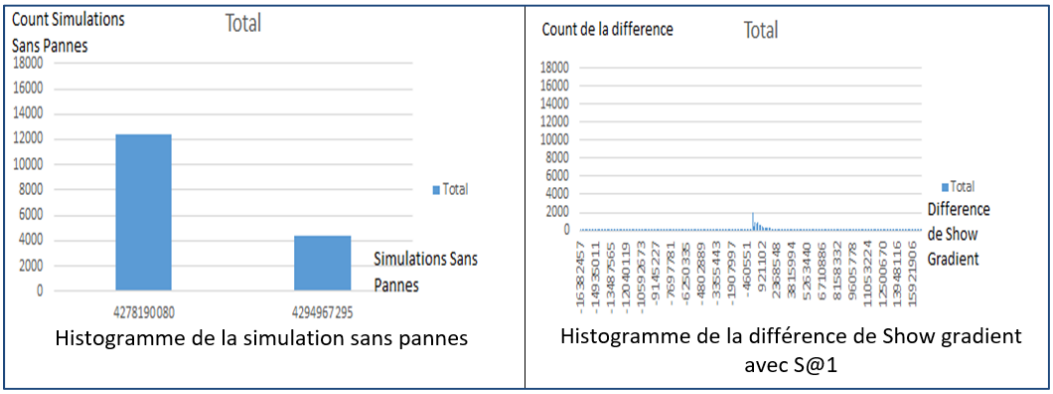

Figure 3.26 Histogrammes de la différence d'injection de pannes (figure droite) et Histogramme de la simulation de référence sans panne (figure gauche)

Dans le cas présent, l'entrée "Show_gradient" est la responsable de la représentation de la valeur du gradient. Si "Show_gradient" = 1, il ne s'agit plus de deux valeurs à la sortie du filtre (les valeurs de pixels en blanc et noir) mais d'une plage de valeurs qui représente les gradients des pixels au niveau de gris, comme illustré à la Figure 3.26.

3.4.3.6 Injection de pannes sur la sortie Video_out

L'injection de panne 'Eout' au niveau de la sortie "Video_out" résulte une génération de 64 séquences (la sortie est sur 32 bits) représentant le comportement fautif du filtre. La génération

de chaque séquence nécessite la fixation de trois paramètres. À savoir : 1) le mode de fonctionnement du saboteur 2) le bit cible sur laquelle va être faite la procédure d'injection et 3) le mode de Stuck at. Les figures 3.27 et 3.28 illustrent le comportement fautif dans le cas d'injection de pannes au niveau du bit 32, pour le mode Stuck at 0. Des exemples des séquences sont fournis à l'annexe IV.

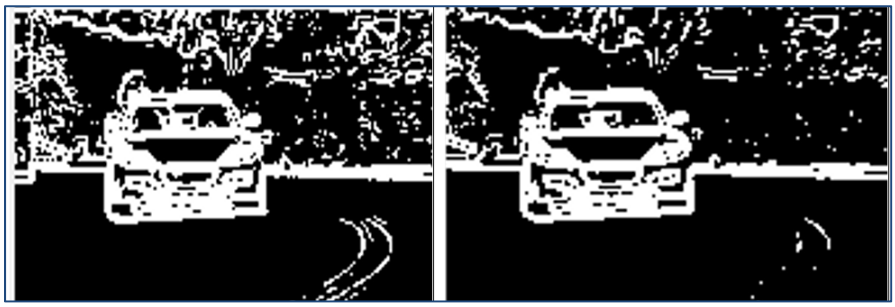

Figure 3.27 Résultats d'injection de pannes sur Video_out =32 S@0 (figure droite) et résultat de la simulation de référence sans panne (figure gauche)

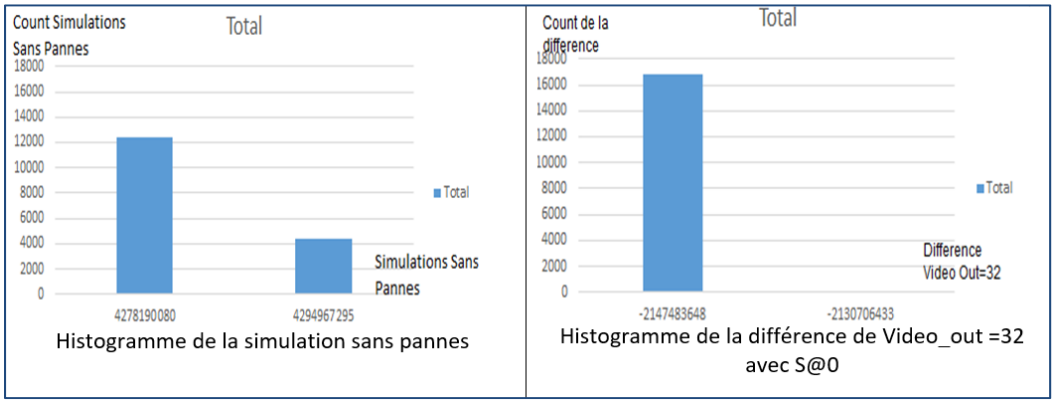

Figure 3.28 Histogrammes de la différence d'injection de pannes (figure droite) et Histogramme de la simulation de référence sans panne (figure gauche)

L'histogramme de la différence de Video_out 32 S@0 présente 0 valeurs identiques et 16800 valeurs différentes. Par contre à l'œil nu les deux images paraissent identiques. La seule différence est au niveau des valeurs des pixels. L'image est au niveau de gris et présente 2 nouveaux valeurs de pixels (2130706432 et 2147483647).

3.4.4 L’impact des saboteurs sur le bloc en aval de « Sobel_HW »

Le bloc en aval du modèle 1 est le bloc « Hough Transform » (Modèle 3). La sortie de ce bloc est une matrice de [381x180] soit 68 580 valeurs. Pour représenter l’effet des saboteurs, le Tableau-A V-1 de l’annexe V récapitule le nombre de la différence des pixels en présence des saboteurs à celle sans pannes.

3.5 Conclusion

Une description des modèles Matlab/Simulink et des circuits de pannes émulant le comportement fautif ont été présentés dans ce chapitre. Des simulations, basées sur l’injection de pannes dans les différents endroits de circuit cible, ont été faites afin d’étudier le comportement fautif dans chaque circuit que ce soit au niveau du filtre ou au niveau du bloc en aval, le bloc de Hough. Dans tous ces scénarios, les comportements fautifs simulés résultant de l’injection de pannes sur chacun des paramètres externes sont représentés. Certains scénarios ont créés des différences significatives de pixels avec le cas de sans pannes, différences qui peuvent par la suite induire des erreurs au niveau de la détection des contours et d’autres au niveau de la détection des lignes pour le bloc en aval. On rappelle qu’une seule panne avec un seul type "stuck at" est faite à la fois.

CHAPITRE 4

ÉTUDE DE LA COUVERTURE DU COMPORTEMENT FAUTIF

4.1 Introduction

Ce chapitre présente les résultats d'une analyse de la couverture du comportement fautif des émulations par les deux saboteurs combinatoires et séquentiels introduits au niveau des ports externes du bloc « Sobel_HW » du premier modèle Matlab/Simulink « Sobel Edge Detection». Cette étude consiste à rappeler ou présenter l'effet de l'injection, par émulation, d'inversion de bits de configuration menant à un comportement fautif d'un bloc, à regarder ses effets sur le bloc en aval et à comparer chacun de ces comportements avec celui des comportements obtenus par les saboteurs. L'objectif de cette comparaison est de valider, de manière informelle, l'hypothèse selon laquelle l'injection de pannes sur les ports via les saboteurs couvre les comportements fautifs par émulation aussi bien au niveau du bloc cible qu'au niveau du bloc en aval.

4.2 Filtre de Sobel

Nous débutons cette étude avec le bloc filtre de Sobel utilisé dans les chapitres précédents.

4.2.1 Injection de pannes par émulation

Dans le chapitre 3, une introduction d'une procédure d'injection de pannes (les saboteurs) a été faite. Dans ce chapitre, une procédure d'injection automatique de pannes est illustrée par la méthode d'injection de pannes par émulation ciblant le bloc filtre de Sobel implanté dans le FPGA Artix-7.

Cette démarche consiste à faire appel à un outil d'injection, "Injector" (Figure 4.1), dédié au FPGA Artix7 A100T. Cet outil a été développé dans le cadre d'un projet précédent par Simon Pichette pour faciliter la communication avec l'utilitaire SEU et par la suite faciliter la procédure d'injection des pannes défini à la section 2.5.3. Dans cette phase, nous utilisons le même modèle synthétisable « Sobel_HW ». Seul ce modèle synthétisable sera la cible des émulations. Notons que cette procédure est applicable à tout type de circuits (combinatoires et séquentiels).

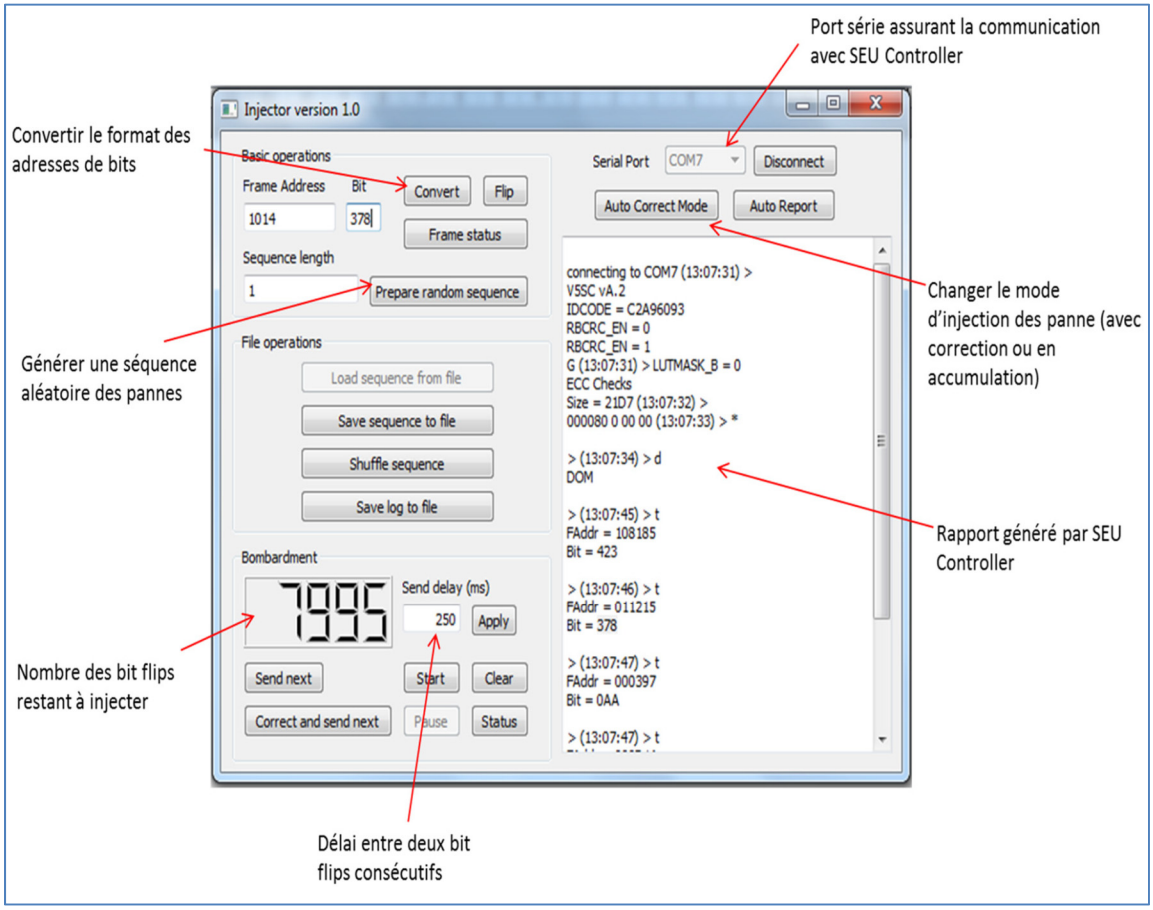

Figure 4.1 Outil « Injector » pour l'injection de pannes par émulation
Tirée de Souari (2016)

4.2.2 Technique d'injection de pannes : par émulation

Le Tableau 4.1 présente les résultats obtenus par émulation, où les pannes sont injectées au niveau du bloc « Sobel_HW ». On y présente le nombre de valeurs différentes à la sortie du bloc « Sobel_HW » par rapport au cas sans pannes NemuSobel et les valeurs des racines carrées de l'erreur quadratique moyenne RemuSobel. Cette comparaison est faite pixel à pixel, sur le vecteur de sortie contenant 16800 pixels. Dans ce projet une centaine d'émulations ont été effectuées.

Sachant que RemuSobel(i) est la racine carrée de l'erreur quadratique moyenne relative de la séquence produite par l'émulation *i* par rapport à une séquence sans pannes, $RELemu_{Sobel}(i)$, RemuSobel(i) est défini :

$$RemuSobel(i)=\sqrt{RELemu_{Sobel(i)}} \tag{4.1}$$

Où $RELemu_{Sobel}(i)$ est égal à :

$$RELemu_{Sobel(i)}=\frac{ABSemu_{Sobel(i)}}{MoyPixel_{Sobel}} \tag{4.2}$$

Et où $ABSemu_{Sobel(i)}$ est l'erreur quadratique moyenne (absolue) de la séquence produite par l'émulation *i* par rapport à une séquence sans pannes, et où $MoyPixel_{Sobel}$ est égal à la valeur moyenne des pixels sans pannes pour le filtre de Sobel (4282599092).

Finalement, ABSemuSobel(i*)* est définie par :

$$ABS_{emuSobel}(i)=\frac{\sum_{k=1}^{16800}\left(Pixel_{Emulations}(k)- Pixel_{sans\ pannes}(k)\right)^2}{PtotSobel} \tag{4.3}$$

Où PtotSobel est le nombre total des pixels pour le filtre de Sobel (16800).

Les résultats du Tableau 4.1 montrent que les valeurs de NemuSobel (i) varient de 1 (émulation 78) à 8255 (émulation 7) et que celles de RemuSobel (i) varient entre 1.98 (émulation 78) à 15374.74 (émulation 22). Il est intéressant de constater que les pires valeurs de NemuSobel (i) et RemuSobel (i) ne proviennent pas de la même émulation, d'où l'intérêt de ces deux métriques.

Tableau 4.1 Nombre de valeurs différentes par rapport au cas sans pannes, NemuSobel (i), et les valeurs de la racine carrée de l'erreur quadratique moyenne relative, RemuSobel (i), pour le filtre de Sobel.

émulation no, i	sobel out NemuSobel(i)	RemuSobel(i)		émulation no, i	sobel out NemuSobel(i)	RemuSobel(i)
1	3	3,43		51	4	3,96
2	182	26,68		52	25	9,89
3	598	48,37		53	476	42,99
4	118	21,49		54	38	12,19
5	3038	109,02		55	128	22,38
6	65	15,95		56	203	28,18
7	8255	179,71		57	105	20,27
8	2904	106,59		58	4181	127,89
9	718	53		59	4089	3,57
10	1467	75,76		60	1421	1,78
11	41	12,66		61	5934	152,36
12	1975	78,9		62	197	27,76
13	3985	124,43		63	16	7,91
14	140	2,43		64	27	10,28
15	153	24,47		65	1061	64,43
16	479	43,29		66	2632	24,19
17	139	23,32		67	788	55,52
18	4849	137,74		68	7404	170,19
19	1113	65,98		69	6	4,84
20	3034	108,95		70	612	48,93
21	4317	129,96		71	27	10,28
22	1168	15374,74		72	163	25,25
23	4565	133,64		73	2390	32,34
24	41	12,66		74	153	24,47
25	4249	128,39		75	1335	72,27
26	544	46,13		76	5934	152,37
27	1113	65,96		77	74	7,31
28	2280	3714,77		78	1	1,98
29	158	24,86		79	33	11,36
30	435	41,25		80	357	37,37
31	2014	88,76		81	3770	93,98
32	25	9,89		82	210	28,66
33	3441	116,03		83	895	59,17
34	4540	133,27		84	4507	132,79
35	4442	131,83		85	2144	91,58
36	15	7,66		86	1362	72,99
37	11	6,56		87	246	31,02
38	1988	88,19		88	35	11,7
39	1185	68,09		89	443	41,63
40	4735	136,1		90	2515	99,19
41	134	22,9		91	1245	69,79
42	2333	95,54		92	20	8,85
43	5615	148,21		93	767	54,78
44	4	3,96		94	2420	97,3
45	210	28,66		95	2616	39,57
46	1366	73,1		96	1061	64,43
47	38	12,19		97	1245	69,79
48	72	16,78		98	105	20,27
49	4744	119,12		99	74	7,31
50	247	31,09		100	35	11,7

4.2.3 Couverture des émulations par les saboteurs sur les nœuds externes au niveau de Sobel

Les résultats résumés au Tableau 4.1 sont comparés à ceux obtenus à l'aide des saboteurs placés, un à la fois, aux entrées et sorties du modèle « Sobel_HW », comme indiqué au chapitre 3 section 3.4.2. Nous comparons les pixels de la sortie à celle de la sortie sans pannes (pixel par pixel) et nous analysons par la suite la couverture des différentes émulations tout en suivant les métriques décrites ci-dessous.

4.2.3.1 Définition de la couverture au niveau du bloc filtre de Sobel

Rappelons ici que l'objectif est de montrer que les pannes induites par les saboteurs peuvent être représentatives de celles induites par émulation (qui sont elles-mêmes considérées comme représentatives de celles induites par les radiations). De ce fait, l'objectif ici est de montrer que, pour chaque émulation effectuée, il y au moins un saboteur menant à une séquence en sortie du module ciblé qui soit au moins aussi erronée que celle de l'émulation effectuée. Dans ce cas, l'émulation en question sera considérée comme complètement couverte. Notons ici qu'il ne s'agit pas ici de vouloir générer des séquences qui soient identiques ou de générer de manière exhaustive toutes les séquences fautives possibles pouvant être causées par les pannes.

De manière plus précise, nous considérons qu'une séquence erronée d'un saboteur couvre complètement une séquence d'émulation si les 2 critères suivants sont respectés:1) elle contient au moins autant de valeurs de pixels différentes (par rapport au cas sans pannes), et 2) l'intensité de ces différences, mesurée à l'aide de la racine carrée de l'erreur quadratique moyenne, est au moins aussi élevée. Dans ce contexte, nous définissons quatre métriques de couverture :

Une couverture de nombre, $C_{nom\_Sobel}(i, j)$, de l'émulation *i* par rapport au saboteur *j*, pour quantifier le premier critère;
Une couverture d'intensité, $C_{int\_Sobel}(i, j)$, de l'émulation *i* par rapport au saboteur *j*, pour quantifier le second critère;

Une couverture globale, $C_{glob\_Sobel}\ (i, j)$, de l'émulation *i* par rapport au saboteur *j*, qui fait l'amalgame des deux premières;
Une couverture résultante, $C_{res\_Sobel}\ (i)$, de l'émulation *i* par rapport au saboteur le plus représentatif, choisi selon les critères décrits plus loin;

Pour commencer la couverture de nombre est représentée par la formule suivante :

$$C_{nom\_Sobel}(i, j) = \min\left(100\%, \frac{NsabSobel(j)}{NemuSobel(i)}\%\right) \tag{4.4}$$

où NsabSobel(j) est le nombre de valeurs de pixels différentes de la séquence produite par le saboteur *j* par rapport à une séquence sans pannes. Rappelons que NemuSobel(i) est le nombre de valeurs de pixels différentes de la séquence produite par l'émulation *i* par rapport à une séquence sans pannes.

La couverture d'intensité est exprimée par la formule suivante :

$$C_{int\_Sobel}(i, j) = \min\left(100\%, \frac{RsabSobel(j)}{RemuSobel(j)}\%\right) \tag{4.5}$$

où RsabSobel(j) est la racine carrée de l'erreur quadratique moyenne de la séquence produite par le saboteur *j* par rapport à une séquence sans pannes (détaillé dans l'annexe VI). Rappelons que RemuSobel(i) est la racine carrée de l'erreur quadratique moyenne de la séquence produite par l'émulation *i* par rapport à une séquence sans pannes.

Enfin la couverture globale est définie comme suit :

$$C_{glob\_Sobel}(i, j) = \min\left(Cnom_{\_Sobel}(i,j), Cint_{\_Sobel}(i,j)\right) \tag{4.6}$$

Pour chaque émulation, nous choisissons le saboteur avec la séquence erronée la plus représentative possible. Ce saboteur est celui qui offre la métrique $C_{glob\_Sobel}\ (i, j)$ la plus importante.

La couverture résultante, $C_{res\_Sobel}\ (i)$, prend alors cette valeur. Dans le cas où plus d'un saboteur mène à une valeur de $C_{glob\_Sobel}\ (i, j)$ égale à 100%, le saboteur choisi est celui pour lequel la distance entre le saboteur et l'émulation,

$$D(i, j)= \sqrt{(RsabSobel(j)-RemuSobel(i))^2+(NsabSobel(j)-NemuSobel(i))^2} \qquad (4.7)$$

est la plus petite.

4.2.3.2 Algorithme de la couverture au niveau du bloc filtre de Sobel

Le test et la vérification de la couverture des pannes sont réalisés par un programme automatisé représenté dans l'annexe VII. C'est un programme développé sous Python et qui fait appel à chaque fois au fichier de résultats de simulation sans pannes, au fichier de résultats des saboteurs et au fichier de résultats de différentes émulations. Un calcul des différentes métriques et une comparaison des résultats des différentes émulations sont présentés. L'exercice est répété avec les résultats de simulation obtenus avec chacun des saboteurs. À la fin une identification, pour chaque émulation, le saboteur avec le nombre de valeurs pixels différentes se rapprochant le plus de tout en étant supérieur à celui de l'émulation ciblée.

Algorithme 4.1 Algorithme de la couverture pour le fltre de Sobel

Début

Ouvrir le fichier des émulations pour Sobel

Ouvrir le fichier des saboteurs pour Sobel

Ouvrir le fichier sans pannes pour Sobel

Boucle1 : Pour **colonne1** dans fichier **emulation. Columns** // Parcourir le fichier colonne par colonne

Var <= 0 // Variable a valeur initiale égale à 0

NemuSobel <= Calculer le nombre de différence du fichier émulation par rapport à sans pannes

ABSemuSobel <= Calculer l'erreur quadratique moyenne absolue pour les émulations

RemuSobel <= Calculer la racine carrée de l'erreur quadratique moyenne relative pour les émulations

Boucle2 : Pour **colonne2** dans fichier **saboteur. Columns** // Parcourir le fichier colonne par colonne

NsabSobel <= Calculer le nombre de différence du fichier saboteur par rapport à sans pannes

ABSsabSobel <= Calculer l'erreur quadratique moyenne absolue pour les saboteurs

RsabSobel <= Calculer la racine carrée de l'erreur quadratique moyenne relative pour les saboteurs

Si **NemuSobel** = **NsabSobel** et **RemuSobel** = **RsabSobel**

Afficher (L'emulation et le saboteur sont identiques)

Sinon Si **NemuSobel** <= **NsabSobel** et **RemuSobel** <= **RsabSobel** // Pour que les saboteurs couvrent les émulations, il faut que le nombre de différence des saboteurs soit supérieur ou égal a celui des emulation et la racine carrée des saboteurs est supérieure ou égale à celle de l'emulation.

Cnom_Sobel <= Calculer la couverture de nombre

Cint_Sobel <= Calculer la couverture d'intensité

Cglob_Sobel <= Calculer la couverture globale

Si **Cglob_Sobel** = 100

D <= Calculer la distance entre le saboteur et l'emulation pour le filtre de Sobel

Si **var** > D Ou **var** = 0 // On cherche le saboteur qui couvre l'emulation avec la distance la plus faible

Var <= **D** // La variable var prend la valeur de la distance du saboteur adéquat

Colonne2min= colonne2 // La colonne Colonne2min prend le nom du saboteur adéquat

Afficher (**Le numéro de l'emulation + RemuSobel + Colonne2min + RsabSobel**)

Enregistrer les résultats dans un fichier texte

Fin

4.2.3.3 Algorithme de la couverture au niveau du bloc filtre de Sobel

Le résultat de la totalité de l'analyse est présenté dans le tableau-A VIII-2 de l'annexe VIII. 100 injections de pannes par émulations sont considérées. Ce résumé présente la couverture des signaux fautifs et la métrique de couverture de chaque émulation.

La première colonne (à partir de la gauche) indique le numéro des émulations, la deuxième présente le nombre de différences entre les émulations et sans pannes observées pour chacune d'émulation NemuSobel, la troisième affiche la racine carrée de l'erreur quadratique moyenne RemuSobel, la quatrième présente les saboteurs couvrants les différentes émulations, la cinquième correspond au nombre de différences entre les saboteurs et sans pannes NsabSobel, la sixième affiche la racine carrée de l'erreur quadratique moyenne RsabSobel (détaillée dans l'annexe VI), les dernières colonnes représentent respectivement la couverture de nombre, la couverture d'intensité et la couverture résultante.

L'intérêt de ce résumé, outre son utilisation à des fins de vérification, réside dans la couverture obtenue. On obtient dans ce cas des pannes qui sont toujours couvertes (figure 4.2).
On parle ici d'une couverture totale des différentes émulations (toutes les émulations sont couvertes par les saboteurs).

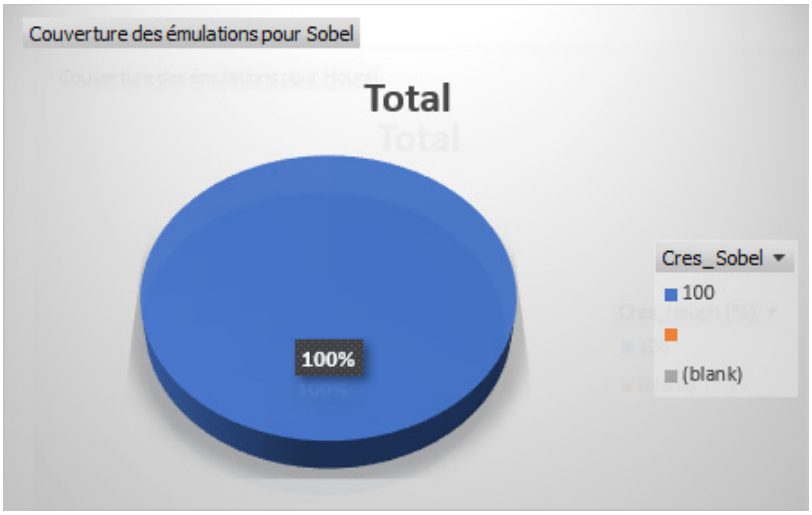

Figure 4.2 Résumé de la couverture pour le filtre de Sobel

4.3 Effet d'injection de panne par émulation sur le bloc en aval : Le bloc de détection des voies de la route

Nous touchons par cette étude le bloc en aval du filtre de Sobel présenté dans les chapitres précédents.

Les comportements fautifs générés lors de l'injection de pannes par émulation au niveau du bloc de filtre de Sobel et en particulier au niveau du bloc synthétisable « Sobel_HW » présente un effet sur le bloc en aval, le bloc de la transformée de Hough. Un récapitulatif des effets de chacune des pannes par émulation sur le bloc en aval est illustrés dans le Tableau 4.2. On y présente le nombre de valeurs différentes à la sortie du bloc « Hough_Transform » par rapport au cas sans pannes NemuHough et les valeurs des racines carrées de l'erreur quadratique moyenne RemuHough.

Pour commencer, RemuHough *(i)* est la racine carrée de l'erreur quadratique moyenne relative de la séquence produite par l'émulation *i* par rapport à une séquence sans pannes, $RELemu_{Hough}(i)$,

$$\text{RemuHough (i)} = \sqrt{\text{RELemu}_{\text{Hough(i)}}} \tag{4.7}$$

Où $RELemu_{Hough}(i)$ est égal à :

$$\text{RELemu}_{\text{Sobel(i)}} = \frac{\text{ABSemu}_{\text{Hough(i)}}}{\text{MoyPixel}_{\text{Hough}}} \tag{4.8}$$

Et où $ABSemu_{Hough}(i)$ est l'erreur quadratique moyenne (absolue) de la séquence produite par l'émulation *i* par rapport à une séquence sans pannes, et où $\text{MoyPixel}_{\text{Hough}}$ est égal à la valeur moyenne des pixels sans pannes pour la transformée de Hough (5,5).

Finalement, $ABSemu_{Hough}(i)$ est définie par :

$$\text{ABS}_{\text{emuHough}}(\text{i}) = \frac{\sum_{\text{k=1}}^{68580} \left(\text{Pixel}_{\text{Emulations}}(\text{k}) - \text{Pixel}_{\text{sans pannes}}(\text{k})\right)^2}{\text{PtotSobel}} \tag{4.9}$$

Où PtotSobel est le nombre total des pixels pour la transformée de Hough (68580).

Tableau 4.2 Nombre de valeurs différentes par rapport au cas sans pannes, NemuHough(i), et les valeurs de la racine carrée de l'erreur quadratique moyenne relative, RemuHough(i), des différentes émulations pour la transformée de Hough

émulatio	Hough out			émulatio	Hough out	
no, i	NemuHough(i)	RemuHough(i)		no, i	NemuHough(i)	NemuHough(i)
1	412	0,03		51	700	0,03
2	13327	0,22		52	2829	0,08
3	20544	0,4		53	15359	0,63
4	11175	0,24		54	22058	2,14
5	22763	5,07		55	10706	0,23
6	8319	0,12		56	15871	0,48
7	29790	9,89		57	12198	0,28
8	23450	2,85		58	24096	3,04
9	24506	1,11		59	21546	6,79
10	24442	2,42		60	21828	2,43
11	5806	0,1		61	23669	8,92
12	28816	2,65		62	10024	0,43
13	25529	4,7		63	2301	0,07
14	6996	0,23		64	3772	0,08
15	13864	0,19		65	18075	1,52
16	19311	0,41		66	27482	3,69
17	13037	0,18		67	21498	1,45
18	25555	3,12		68	29763	6,61
19	23563	1,69		69	780	0,03
20	22756	4,73		70	20820	0,45
21	24579	5,75		71	3673	0,08
22	14556	0,43		72	10711	0,17
23	24969	4,29		73	14010	0,2
24	6012	0,1		74	10516	0,39
25	24079	6,64		75	25283	1,12
26	12535	1,21		76	23669	8,92
27	14556	0,43		77	2618	0,06
28	20297	2,51		78	180	0,02
29	9050	0,37		79	4630	0,09
30	15359	0,63		80	16609	0,54
31	20810	2,46		81	21201	6,12
32	2766	0,08		82	15802	0,49
33	23451	3,96		83	22343	0,76
34	29651	5,81		84	24395	3,18
35	25007	3,09		85	26323	2,28
36	1932	0,07		86	26000	1,96
37	1752	0,05		87	16258	0,24
38	21780	2,64		88	3917	0,11
39	20315	1,42		89	17908	0,61
40	26443	6,76		90	21768	2,43
41	12229	0,18		91	24192	1,53
42	24483	1,54		92	3342	`0,07
43	27720	7,66		93	19238	4,37
44	700	0,03		94	23272	1,22
45	15802	0,49		95	26241	1,86
46	22058	2,14		96	18075	1,52
47	6996	0,23		97	24192	1,53
48	11175	0,24		98	12198	0,28
49	19998	1,62		99	2618	0,06
50	15233	0,57		100	3917	0,11

De gauche à droite, la table contient les numéros des émulations (les mêmes émulations que précédemment) injectées au niveau du filtre de Sobel, la somme des valeurs différentes par rapport au cas sans pannes des émulations et enfin la racine carrée de l'erreur quadratique moyenne des différentes émulations. Les résultats montrent que les valeurs de NemuHough(i) varient de 0.02 (émulation 78) à 29790 (émulation 7) et que les valeurs de RemuHough(i) varient de 180 (émulation 78) à 9.89 (émulation 7). Il est à noter que l'émulation 78 présentait également les plus faibles valeurs de NemuSobel (i) et RemuSobel (i), alors que l'émulation 7 présentait la plus forte valeur pour NemuSobel (i).

4.3.1 Étude de la couverture au niveau du bloc en aval

Similairement au bloc de la détection des bords, cette analyse de couverture consiste à vérifier si les séquences de sortie de la transformée de Hough dues aux injections de pannes par émulation sont couvertes par celles dues aux injections de pannes faites sur les nœuds externes du filtre de Sobel par les saboteurs. Il ne s'agit pas ici de générer des séquences au niveau de la transformée de Hough qui détectent de manière optimale les pannes ou de générer de manière exhaustive toutes les séquences fautives possibles pouvant être causées par les pannes. Il s'agit plutôt de comparer des séquences causées par des pannes au niveau du filtre de Sobel, et de regarder leurs effets sur le bloc en aval.

Dans cette partie les mêmes métriques de couverture sont utilisées :

Une couverture de nombre, $C_{nom\_Hough}\ (i, j)$, de l'émulation *i* par rapport au saboteur *j*, pour quantifier le premier critère;
Une couverture d'intensité, $C_{int\_Hough}\ (i, j)$, de l'émulation *i* par rapport au saboteur *j*, pour quantifier le second critère;
Une couverture globale, $C_{glob\_Hough}\ (i, j)$, de l'émulation *i* par rapport au saboteur *j*, qui fait l'amalgame des deux premières;

Une couverture résultante, $C_{res\_Hough}(i)$, de l'émulation *i* par rapport au saboteur le plus représentatif, choisi selon les critères décrits plus loin;

4.3.2 Algorithme de la couverture au niveau du bloc de Hough

L'algorithme de la couverture au niveau de Hough est semblable à celui de Sobel. Une différence réside dans le traitement des données. Pour le filtre de Sobel, nous travaillons avec des vecteurs mais ici avec la transformée de Hough, nous traitons des matrices. Nous vérifions la couverture au niveau de Hough tout en comparant les matrices colonnes par colonnes.
Le code python du test et de la vérification de la couverture des pannes est représenté dans la deuxième partie de l'annexe VII .

Figure 4.3 Algorithme de la couverture pour la transformée de Hough

```
Début
Ouvrir le fichier des émulations pour Sobel
Ouvrir le fichier des saboteurs pour Sobel
Ouvrir le fichier sans pannes pour Sobel

Boucle1 : Pour colonne1 dans fichier emulation. Columns // Parcourir le fichier colonne par colonne
  Var <= 0 // Variable a valeur initiale égale à 0
  NemuHough <= Calculer le nombre de différence du fichier émulation par rapport à sans pannes
  ABSemuHough <= Calculer l'erreur quadratique moyenne absolue pour les émulations
  RemuHough <= Calculer la racine carrée de l'erreur quadratique moyenne relative pour les émulations
  Boucle2 : Pour colonne2 dans fichier saboteur. Columns // Parcourir le fichier colonne par colonne
    NsabHough<= Calculer le nombre de différence du fichier saboteur par rapport à sans pannes
    ABSsabHough <= Calculer l'erreur quadratique moyenne absolue pour les saboteurs
    RsabHough <= Calculer la racine carrée de l'erreur quadratique moyenne relative pour les saboteurs
    Si NemuHough= NsabHough et RemuHough = RsabHough
                                  Afficher (L'emulation et le saboteur sont identiques)
    Sinon Si NemuHough <= NsabHough et RemuHough <= RsabHough // Pour que les saboteurs couvrent les émulations, il faut que
le nombre de différence des saboteurs soit supérieur ou égal a celui des emulation et la racine carrée des saboteurs est supérieure ou égale
à celle de l'emulation.
      Cnom_Hough <= Calculer la couverture de nombre
      Cint_Hough <= Calculer la couverture d'intensité
```

```
    Cglob_Hough <= Calculer la couverture globale
    Si Cglob_Hough = 100
      D <= Calculer la distance entre le saboteur et l'emulation pour le filtre de Sobel

    Si var > D Ou var = 0 // On cherche le saboteur qui couvre l'emulation avec la distance la plus faible
      Var <= D // La variable var prend la valeur de la distance du saboteur adéquat
      Colonne2min= colonne2 // La colonne Colonne2min prend le nom du saboteur adéquat
Afficher (Le numéro de l'emulation + RemuHough + Colonne2min + RsabHough)
Enregistrer les résultats dans un fichier texte
Fin
```

4.3.3 Résultats de la couverture au niveau du bloc de Hough

Un résumé de la totalité de l'analyse est présenté dans tableau-A IX-3 de l'annexe IX. Les 100 injections de pannes par émulation au niveau de filtre de Sobel sont considérées (nous travaillons avec les même émulations). Ce résumé présente la couverture des signaux fautifs et la métrique de couverture (couverture résultante) de chaque émulation.

La première colonne (à partir de la gauche) indique le numéro des émulations, la deuxième présente le nombre de différences entre les émulations et sans pannes observées pour chacune d'émulation NemuHough, la troisième affiche la racine carrée de l'erreur quadratique moyenne RemuHough, la quatrième présente les saboteurs couvrants les différentes émulations, la cinquième correspond au nombre de différences entre les saboteurs et sans pannes NsabHough, la sixième affiche la racine carrée de l'erreur quadratique moyenne RsabHough (détaillée dans l'annexe VI), les dernières colonnes représentent respectivement la couverture de nombre, la couverture d'intensité et la couverture résultante.

L'utilité de ce résumé, outre son utilisation à des fins de vérification, réside dans la couverture obtenue (figure 4.3). On obtient dans ce cas des pannes qui sont toujours couvertes. Nous parlons encore ici d'une couverture totale des différentes émulations au niveau de Hough.

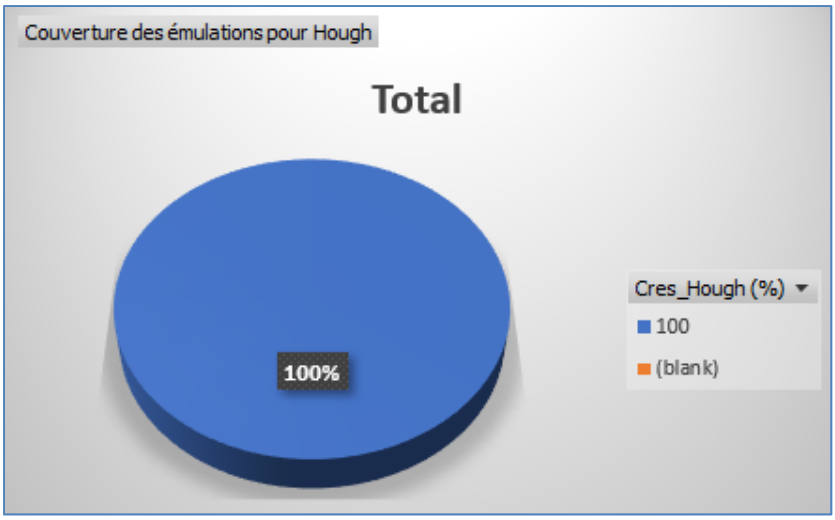

Figure 4.3 Résumé de la couverture pour la transformée de Hough

4.4 Conclusion

Dans ce chapitre, nous avons validé l'hypothèse de la couverture des comportements fautifs des émulations, et ceci pour le bloc de la détection des contours et pour le bloc de la détection des voies de la route. Cette étude a été basée sur la comparaison des sorties données par l'injection de pannes sur les ports externes par rapport aux injections de pannes par émulation, via la représentation des métriques et le calcul du taux de la couverture résultante.

CONCLUSION

L'objectif de ce mémoire était la validation plus poussée de modèles de pannes à haut niveau d'abstraction (Matlab/Simulink) représentant l'effet des radiations cosmiques sur des fonctions usuelles du domaine de l'automobile lorsqu'implémentées dans des circuits FPGA à base de mémoire SRAM. Ces modèles, une fois validés, permettent de simuler et mesurer très tôt dans le processus de conception l'effet de ces radiations, et ainsi juger de la pertinence d'appliquer ou non des méthodes de mitigation. Afin de répondre à cet objectif, nous avons procédé comme suit.

Nous avons présenté, dans un premier temps, les modèles de pannes de haut niveau à investiguer, à savoir les saboteurs combinatoires, et les saboteurs séquentiels, issus d'un projet précédent. Ces blocs ont été insérés aux différentes broches d'entrées sorties d'un module synthétisable du filtre de Sobel , inclus dans la librairie de Matlab/Simulink, à la sortie duquel nous avons placé un module de la transformée de Hough. Toujours dans l'environnement Matlab/Simulink, nous avons fait les simulations relatives à chaque cas, une seule panne à la fois pour chaque simulation, afin de recueillir et d'analyser les sorties erronées des modules de Sobel et de Hough.

Nous avons présenté, dans un deuxième temps, un autre type d'injection d'erreur, l'injection par émulation, et nous avons précédé à l'introduction des pannes par l'intermédiaire d'un module développé par la compagnie Xilinx, le SEM Controller, et d'un outil existant, à savoir l'outil « Injector7 » également issu d'un projet précédent. Cette procédure permet de reproduire plus facilement des résultats semblables à ceux obtenus par les radiations cosmiques. L'injection de pannes par émulation se fait au niveau du module synthétisable de Sobel, dans un environnement de type matériel dans la boucle, où le module de Sobel et le SEM Controller sont implémentés dans un FPGA et où le reste du modèle demeure sous l'environnement Matlab/Simulink. Encore ici, les sorties erronées des modules de Sobel et de

Hough ont été recueillies et analysées afin de la comparer à celles produites par les modèles de pannes à haut niveau.

Comme dernière étape de ce travail, nous avons présenté cette comparaison, sous la forme d'une étude de couverture du comportement fautif du modèle Matlab/Simulink du filtre de Sobel suivie de la transformée de Hough. Cette étude est basée sur une comparaison des séquences du circuit défectueux entre l'injection de pannes sur ses nœuds externes (les saboteurs) face aux injections de pannes par émulation. Ce sont les résultats de cette étude qui ont permis de valider l'hypothèse selon laquelle l'injection de pannes par les saboteurs, lorsqu'utilisés aux entrées/sorties d'un module, couvre la totalité des comportements fautifs par émulation non seulement sur le module cible, mais aussi le module en aval. Pour effectuer cette étude, nous avons utilisé les résultats des simulations d'injection de pannes avec saboteurs sur les nœuds externes du module de Sobel Matlab/Simulink, ceux des simulations d'injection de pannes par émulation au niveau du module de Sobel synthétisé dans un FPGA, et un programme automatisé pour la comparaison et la vérification de la couverture au niveau du bloc bombardé et le bloc en aval. Une panne émulée sur FPGA a été considérée couverte s'il existait une panne de saboteurs au moins aussi perturbatrice sur les sorties des modules de Sobel et de Hough. Des métriques d'évaluation ont été attribuées à chaque comparaison et un taux de couverture résultante a été calculé. Les résultats obtenus ont permis de monter que tous les comportements fautifs liés aux pannes par émulation étant couverts par les pannes de saboteurs sur les ports du module de Sobel. Les modèles structuraux de pannes de haut niveau d'abstraction insérés au niveau des entrées/sortiea du module de Sobel ont donc mené à des séquences au moins aussi erronées que celle des émulations effectuées aussi bien au niveau du module cible qu'au niveau de celui en en aval. En plus de permettre une évaluation plus tôt dans le processu de conception, cette validation permet de réduire le nombre d'injections de pannes à effectuer pour étudier le comportement des circuits face aux radiations.

RECOMMANDATIONS

Dans ce mémoire, nous avons étudié le comportement fautif de certains blocs Matlab/Simulink face aux radiations cosmiques. Cette étude couvre différents modèles en rapport avec le domaine de l'automobile tels que le filtre de Sobel et la transformée de Hough avec le degré de complexité présenté surtout au niveau de la combinaison des deux blocs. Dans cet axe, il serait intéressant de faire des tests d'injection de pannes sur d'autres modèles employés pour les véhicules autonomes afin d'avoir une étude complète qui estime le comportement fautif de chaque composant et puis le circuit au complet face aux radiations.

Une autre ouverture dans ce projet consiste à améliorer la qualité de couverture des émulations par les saboteurs présentés dans le chapitre 4, afin d'avoir une très bonne couverture, vu que les véhicules autonomes font partie des applications les plus critiques de nos jours. Cette amélioration consiste à développer des combinaisons de scénarios bien déterminées pour les différentes tâches. Ceci revient à dire que chaque combinaison tient en compte plusieurs saboteurs qui fassent l'injection de pannes au même temps (c'est-à-dire d'activer plusieurs saboteurs dans la même simulation).

ANNEXE I

CALCUL DE LA SENSIBILITÉ DES FPGA

Ce calcul est basé sur les données sur l'effet des radiations fournies par Xilinx (UG116, Xilinx, April 23, 2021). On commence par calculer la sensibilité des différents FPGA tout en supposant que tous les bits de configurations sont des bits critiques.

Classification du niveau ASIL :

	ASIL-A	ASIL-B	ASIL-C	ASIL-D
SPF (Single Point fault) Metric	Not Applicable	> 90%	> 97%	> 99%
LF (Latent Fault) Metric	Not Applicable	> 60%	> 80%	> 90%
Failure rate	10^{-6}/hour	10^{-7}/hour	10^{-7}/hour	10^{-8}/hour
FIT (failure in time)	< 1,000 FIT	< 100 FIT	< 100 FIT	< 10 FIT

Calcul de la sensibilité :

Product Family	**Real-Time SER per Event**(FIT/Mb)	**Nombre de bits**	**Nombre de FIT**	**Niveau ASIL**
Kintex UltraScale KU025	31	128055264	3969,713184	
Kintex UltraScale KU035	31	128055265	3969,713215	
Kintex UltraScale KU040	31	128055266	3969,713246	
Kintex UltraScale KU060	31	192999264	5982,977184	

Kintex UltraScale KU085	31	386012288	11966,38093	
Kintex UltraScale KU095	31	286746912	8889,154272	
Kintex UltraScale KU0115	31	386012288	11966,38093	
Virtex UltraScale VU065	31	200713824	6222,128544	
Virtex UltraScale VU080	31	286746912	8889,154272	
Virtex UltraScale VU095	31	286746913	8889,154303	
Virtex UltraScale VU125	31	401441408	12444,68365	
Virtex UltraScale VU160	31	602156064	18666,83798	
Virtex UltraScale VU190	31	602156064	18666,83798	
Virtex UltraScale VU440	31	1031731104	31983,66422	
Kintex UltraScale+ KU3P	5	123449056	617,24528	ASIL A
Kintex UltraScale+ KU5P	5	123449056	617,24528	ASIL A
Kintex UltraScale+ KU9P	5	212086240	1060,4312	
Kintex UltraScale+ KU11P	5	188647264	943,23632	ASIL A
Kintex UltraScale+ KU13P	5	229605952	1148,02976	
Kintex UltraScale+ KU15P	5	290744896	1453,72448	
Virtex UltraScale+ VU3P	5	213752800	1068,764	
Virtex UltraScale+ VU5P	5	427519232	2137,59616	
Virtex UltraScale+ VU7P	5	427519232	2137,59616	
Virtex UltraScale+ VU9P	5	641272864	3206,36432	

Virtex UltraScale+ VU11P	5	679913248	3399,56624	
Virtex UltraScale+ VU13P	5	906547008	4532,73504	
Virtex UltraScale+ VU27P	5	906547008	4532,73504	
Virtex UltraScale+ VU29P	5	906547008	4532,73504	
Virtex UltraScale+ VU31P	5	226632928	1133,16464	
Virtex UltraScale+ VU33P	5	226632928	1133,16464	
Virtex UltraScale+ VU35P	5	453279488	2266,39744	
Virtex UltraScale+ VU37P	5	679913248	3399,56624	
Virtex-5 XC5VLX20T	165	6251200	1031,448	
Virtex-5 XC5VLX30T	165	9371136	1546,23744	
Virtex-5 XC5VLX50T	165	14052352	2318,63808	
Virtex-5 XC5VLX85T	165	23341312	3851,31648	
Virtex-5 XC5VLX110T	165	31118848	5134,60992	
Virtex-5 XC5VLX155T	165	43042304	7101,98016	
Virtex-5 XC5VLX220T	165	55133696	9097,05984	
Virtex-5 XC5VLX330T	165	82696192	13644,87168	
Virtex-5 XC5VSX35T	165	13349120	2202,6048	
Virtex-5 XC5VSX50T	165	20019328	3303,18912	
Virtex-5 XC5VSX95T	165	35716096	5893,15584	
Spartan-7 7S6	76	4310752	327,617152	ASIL A
Spartan-7 7S15	76	4310752	327,617152	ASIL A
Spartan-7 7S25	76	9934432	755,016832	ASIL A

Spartan-7 7S50	76	17536096	1332,743296	
Spartan-7 7S75	76	29494496	2241,581696	
Spartan-7 7S100	76	29494496	2241,581696	
Artix -7 7A12T	76	9934432	755,016832	ASIL A
Artix -7 7A15T	76	17536096	1332,743296	
Artix -7 7A25T	76	9934432	755,016832	ASIL A
Artix -7 7A35T	76	17536096	1332,743296	
Artix -7 7A50T	76	17536096	1332,743296	
Artix -7 7A75T	76	30606304	2326,079104	
Artix -7 7A100T	76	30606305	2326,07918	
Artix -7 7A200T	76	77845216	5916,236416	
Kintex-7 7K70T	50	24090592	1204,5296	
Kintex-7 7K160T	50	53540576	2677,0288	
Kintex-7 7K325T	50	91548896	4577,4448	
Kintex-7 7K355T	50	112414688	5620,7344	
Kintex-7 7K410T	50	127023328	6351,1664	
Kintex-7 7K420T	50	149880032	7494,0016	
Kintex-7 7K480T	50	149880032	7494,0016	
Virtex-7 7V585T	50	161398880	8069,944	
Virtex-7 7V2000T	50	447337216	22366,8608	
Virtex-7 7VX330T	50	111238240	5561,912	
Virtex-7 7VX485T	50	137934560	6896,728	
Virtex-7 7VX550T	50	162187488	8109,3744	
Virtex-7 7VX550T	50	229878496	11493,9248	
Spartan-3S200A/AN	190	1196128	227,26432	ASIL A
Spartan-3S400A/AN	190	1886560	358,4464	ASIL A
Spartan-3S700A/AN	190	2732640	519,2016	ASIL A
Spartan-3S1400A/AN	190	4755296	903,50624	ASILA
Spartan-3SD1800A	190	8197280	1557,4832	
Spartan-3SD3400A	190	11718304	2226,47776	
Spartan-3S100E	190	581344	110,45536	ASIL A
Spartan-3S250E	190	1353728	257,20832	ASIL A
Spartan-3S500E	190	2270208	431,33952	ASIL A
Spartan-3S1200E	190	3841184	729,82496	ASIL A

Spartan-3S1600E	190	5969696	1134,24224	
Spartan-3S200	190	1047616	199,04704	ASIL A
Spartan-3S400	190	1699136	322,83584	ASIL A
Spartan-3S1000	190	3223488	612,46272	ASIL A
Spartan-3S1500	190	5214784	990,80896	ASIL A
Spartan-3S2000	190	7673024	1457,87456	
Spartan-3S4000	190	11316864	2150,20416	
Spartan-3S5000	190	13271936	2521,66784	
Spartan-6 6SLX4	177	2731488	483,473376	ASIL A
Spartan-6 6SLX9	177	2742528	485,427456	ASIL A
Spartan-6 6SLX16	177	3731264	660,433728	ASIL A
Spartan-6 6SLX25	177	6440432	1139,956464	
Spartan-6 6SLX25T	177	6440432	1139,956464	

Hypothèse 1 : 25% des bits de configurations sont des bits critiques

Product Family	Real-Time SER per Event	Nombre de bits	Nombre de FIT	Niveau ASIL
Kintex UltraScale KU025	31	128055264	992,428296	ASIL A
Kintex UltraScale KU035	31	128055265	992,4283038	ASIL A
Kintex UltraScale KU040	31	128055266	992,4283115	ASIL A
Kintex UltraScale KU060	31	192999264	1495,744296	
Kintex UltraScale KU085	31	386012288	2991,595232	
Kintex UltraScale KU095	31	286746912	2222,288568	
Kintex UltraScale KU0115	31	386012288	2991,595232	
Virtex UltraScale VU065	31	200713824	1555,532136	

Virtex UltraScale VU080	31	286746912	2222,288568	
Virtex UltraScale VU095	31	286746913	2222,288576	
Virtex UltraScale VU125	31	401441408	3111,170912	
Virtex UltraScale VU160	31	602156064	4666,709496	
Virtex UltraScale VU190	31	602156064	4666,709496	
Virtex UltraScale VU440	31	1031731104	7995,916056	
Kintex UltraScale+ KU3P	5	123449056	154,31132	ASIL A
Kintex UltraScale+ KU5P	5	123449056	154,31132	ASIL A
Kintex UltraScale+ KU9P	5	212086240	265,1078	ASIL A
Kintex UltraScale+ KU11P	5	188647264	235,80908	ASIL A
Kintex UltraScale+ KU13P	5	229605952	287,00744	ASIL A
Kintex UltraScale+ KU15P	5	290744896	363,43112	ASIL A
Virtex UltraScale+ VU3P	5	213752800	267,191	ASIL A
Virtex UltraScale+ VU5P	5	427519232	534,39904	ASIL A
Virtex UltraScale+ VU7P	5	427519232	534,39904	ASIL A
Virtex UltraScale+ VU9P	5	641272864	801,59108	ASIL A
Virtex UltraScale+ VU11P	5	679913248	849,89156	ASIL A
Virtex UltraScale+ VU13P	5	906547008	1133,18376	
Virtex UltraScale+ VU27P	5	906547008	1133,18376	
Virtex UltraScale+ VU29P	5	906547008	1133,18376	

Virtex UltraScale+ VU31P	5	226632928	283,29116	ASIL A
Virtex UltraScale+ VU33P	5	226632928	283,29116	ASIL A
Virtex UltraScale+ VU35P	5	453279488	566,59936	ASIL A
Virtex UltraScale+ VU37P	5	679913248	849,89156	ASIL A
Virtex-5 XC5VLX20T	165	6251200	257,862	ASIL A
Virtex-5 XC5VLX30T	165	9371136	386,55936	ASIL A
Virtex-5 XC5VLX50T	165	14052352	579,65952	ASIL A
Virtex-5 XC5VLX85T	165	23341312	962,82912	ASIL A
Virtex-5 XC5VLX110T	165	31118848	1283,65248	
Virtex-5 XC5VLX155T	165	43042304	1775,49504	
Virtex-5 XC5VLX220T	165	55133696	2274,26496	
Virtex-5 XC5VLX330T	165	82696192	3411,21792	
Virtex-5 XC5VSX35T	165	13349120	550,6512	ASIL A
Virtex-5 XC5VSX50T	165	20019328	825,79728	ASIL A
Virtex-5 XC5VSX95T	165	35716096	1473,28896	
Spartan-7 7S6	76	4310752	81,904288	ASIL B-C
Spartan-7 7S15	76	4310752	81,904288	ASIL B-C
Spartan-7 7S25	76	9934432	188,754208	ASIL A
Spartan-7 7S50	76	17536096	333,185824	ASIL A
Spartan-7 7S75	76	29494496	560,395424	ASIL A
Spartan-7 7S100	76	29494496	560,395424	ASIL A
Artix -7 7A12T	76	9934432	188,754208	ASIL A
Artix -7 7A15T	76	17536096	333,185824	ASIL A

Artix -7 7A25T	76	9934432	188,754208	ASIL A
Artix -7 7A35T	76	17536096	333,185824	ASIL A
Artix -7 7A50T	76	17536096	333,185824	ASIL A
Artix -7 7A75T	76	30606304	581,519776	ASIL A
Artix -7 7A100T	76	30606305	581,519795	ASIL A
Artix -7 7A200T	76	77845216	1479,059104	
Kintex-7 7K70T	50	24090592	301,1324	ASIL A
Kintex-7 7K160T	50	53540576	669,2572	ASIL A
Kintex-7 7K325T	50	91548896	1144,3612	
Kintex-7 7K355T	50	112414688	1405,1836	
Kintex-7 7K410T	50	127023328	1587,7916	
Kintex-7 7K420T	50	149880032	1873,5004	
Kintex-7 7K480T	50	149880032	1873,5004	
Virtex-7 7V585T	50	161398880	2017,486	
Virtex-7 7V2000T	50	447337216	5591,7152	
Virtex-7 7VX330T	50	111238240	1390,478	
Virtex-7 7VX485T	50	137934560	1724,182	
Virtex-7 7VX550T	50	162187488	2027,3436	
Virtex-7 7VX550T	50	229878496	2873,4812	
Spartan-3S50A/AN	190	437312	20,77232	ASIL B-C
Spartan-3S200A/AN	190	1196128	56,81608	ASIL B-C
Spartan-3S400A/AN	190	1886560	89,6116	ASIL B-C
Spartan-3S700A/AN	190	2732640	129,8004	ASIL A
Spartan-3S1400A/AN	190	4755296	225,87656	ASIL A
Spartan-3SD1800A	190	8197280	389,3708	ASIL A
Spartan-3SD3400A	190	11718304	556,61944	ASIL A
Spartan-3S100E	190	581344	27,61384	ASIL B-C
Spartan-3S250E	190	1353728	64,30208	ASIL B-C
Spartan-3S500E	190	2270208	107,83488	ASIL A
Spartan-3S1200E	190	3841184	182,45624	ASIL A
Spartan-3S1600E	190	5969696	283,56056	ASIL A
Spartan-3S50	190	439264	20,86504	ASIL B-C
Spartan-3S200	190	1047616	49,76176	ASIL B-C

Spartan-3S400	190	1699136	80,70896	ASIL B-C
Spartan-3S1000	190	3223488	153,11568	ASIL A
Spartan-3S1500	190	5214784	247,70224	ASIL A
Spartan-3S2000	190	7673024	364,46864	ASIL A
Spartan-3S4000	190	11316864	537,55104	ASIL A
Spartan-3S5000	190	13271936	630,41696	ASIL A
Spartan-6 6SLX4	177	2731488	120,868344	ASIL A
Spartan-6 6SLX9	177	2742528	121,356864	ASIL A
Spartan-6 6SLX16	177	3731264	165,108432	ASIL A
Spartan-6 6SLX25	177	6440432	284,989116	ASIL A
Spartan-6 6SLX25T	177	6440432	284,989116	ASIL A

Hypothèse 2 : 20% des bits de configurations sont des bits critiques

Product Family	Real-Time SER per Event	Nombre de bits	Nombre de FIT	Niveau ASIL
Kintex UltraScale KU025	31	128055264	793,9426368	ASIL A
Kintex UltraScale KU035	31	128055265	793,942643	ASIL A
Kintex UltraScale KU040	31	128055266	793,9426492	ASIL A
Kintex UltraScale KU060	31	192999264	1196,595437	
Kintex UltraScale KU085	31	386012288	2393,276186	
Kintex UltraScale KU095	31	286746912	1777,830854	
Kintex UltraScale KU0115	31	386012288	2393,276186	

Virtex UltraScale VU065	31	200713824	1244,425709	
Virtex UltraScale VU080	31	286746912	1777,830854	
Virtex UltraScale VU095	31	286746913	1777,830861	
Virtex UltraScale VU125	31	401441408	2488,93673	
Virtex UltraScale VU160	31	602156064	3733,367597	
Virtex UltraScale VU190	31	602156064	3733,367597	
Virtex UltraScale VU440	31	1031731104	6396,732845	
Kintex UltraScale+ KU3P	5	123449056	123,449056	ASIL A
Kintex UltraScale+ KU5P	5	123449056	123,449056	ASIL A
Kintex UltraScale+ KU9P	5	212086240	212,08624	ASIL A
Kintex UltraScale+ KU11P	5	188647264	188,647264	ASIL A
Kintex UltraScale+ KU13P	5	229605952	229,605952	ASIL A
Kintex UltraScale+ KU15P	5	290744896	290,744896	ASIL A
Virtex UltraScale+ VU3P	5	213752800	213,7528	ASIL A

Virtex UltraScale+ VU5P	5	427519232	427,519232	ASIL A
Virtex UltraScale+ VU7P	5	427519232	427,519232	ASIL A
Virtex UltraScale+ VU9P	5	641272864	641,272864	ASIL A
Virtex UltraScale+ VU11P	5	679913248	679,913248	ASIL A
Virtex UltraScale+ VU13P	5	906547008	906,547008	ASIL A
Virtex UltraScale+ VU27P	5	906547008	906,547008	ASIL A
Virtex UltraScale+ VU29P	5	906547008	906,547008	ASIL A
Virtex UltraScale+ VU31P	5	226632928	226,632928	ASIL A
Virtex UltraScale+ VU33P	5	226632928	226,632928	ASIL A
Virtex UltraScale+ VU35P	5	453279488	453,279488	ASIL A
Virtex UltraScale+ VU37P	5	679913248	679,913248	ASIL A
Virtex-5 XC5VLX20T	165	6251200	206,2896	ASIL A
Virtex-5 XC5VLX30T	165	9371136	309,247488	ASIL A
Virtex-5 XC5VLX50T	165	14052352	463,727616	ASIL A
Virtex-5 XC5VLX85T	165	23341312	770,263296	ASIL A

Virtex-5 XC5VLX110T	165	31118848	1026,921984	
Virtex-5 XC5VLX155T	165	43042304	1420,396032	
Virtex-5 XC5VLX220T	165	55133696	1819,411968	
Virtex-5 XC5VLX330T	165	82696192	2728,974336	
Virtex-5 XC5VSX35T	165	13349120	440,52096	ASIL A
Virtex-5 XC5VSX50T	165	20019328	660,637824	ASIL A
Virtex-5 XC5VSX95T	165	35716096	1178,631168	
Spartan-7 7S6	76	4310752	65,5234304	ASIL B-C
Spartan-7 7S15	76	4310752	65,5234304	ASIL B-C
Spartan-7 7S25	76	9934432	151,0033664	ASIL A
Spartan-7 7S50	76	17536096	266,5486592	ASIL A
Spartan-7 7S75	76	29494496	448,3163392	ASIL A
Spartan-7 7S100	76	29494496	448,3163392	ASIL A
Artix -7 7A12T	76	9934432	151,0033664	ASIL A
Artix -7 7A15T	76	17536096	266,5486592	ASIL A
Artix -7 7A25T	76	9934432	151,0033664	ASIL A
Artix -7 7A35T	76	17536096	266,5486592	ASIL A
Artix -7 7A50T	76	17536096	266,5486592	ASIL A
Artix -7 7A75T	76	30606304	465,2158208	ASIL A
Artix -7 7A100T	76	30606305	465,215836	ASIL A
Artix -7 7A200T	76	77845216	1183,247283	
Kintex-7 7K70T	50	24090592	240,90592	ASIL A
Kintex-7 7K160T	50	53540576	535,40576	ASIL A
Kintex-7 7K325T	50	91548896	915,48896	ASIL A
Kintex-7 7K355T	50	112414688	1124,14688	
Kintex-7 7K410T	50	127023328	1270,23328	
Kintex-7 7K420T	50	149880032	1498,80032	
Kintex-7 7K480T	50	149880032	1498,80032	

Virtex-7 7V585T	50	161398880	1613,9888	
Virtex-7 7V2000T	50	447337216	4473,37216	
Virtex-7 7VX330T	50	111238240	1112,3824	
Virtex-7 7VX485T	50	137934560	1379,3456	
Virtex-7 7VX550T	50	162187488	1621,87488	
Virtex-7 7VX550T	50	229878496	2298,78496	
Spartan-3S50A/AN	190	437312	16,617856	ASIL B-C
Spartan-3S200A/AN	190	1196128	45,452864	ASIL B-C
Spartan-3S400A/AN	190	1886560	71,68928	ASIL B-C
Spartan-3S700A/AN	190	2732640	103,84032	ASIL A
Spartan-3S1400A/AN	190	4755296	180,701248	ASIL A
Spartan-3SD1800A	190	8197280	311,49664	ASIL A
Spartan-3SD3400A	190	11718304	445,295552	ASIL A
Spartan-3S100E	190	581344	22,091072	ASIL B-C
Spartan-3S250E	190	1353728	51,441664	ASIL B-C
Spartan-3S500E	190	2270208	86,267904	ASIL B-C
Spartan-3S1200E	190	3841184	145,964992	ASIL A
Spartan-3S1600E	190	5969696	226,848448	ASIL A
Spartan-3S50	190	439264	16,692032	ASIL B-C
Spartan-3S200	190	1047616	39,809408	ASIL B-C
Spartan-3S400	190	1699136	64,567168	ASIL B-C
Spartan-3S1000	190	3223488	122,492544	ASIL A
Spartan-3S1500	190	5214784	198,161792	ASIL A
Spartan-3S2000	190	7673024	291,574912	ASIL A
Spartan-3S4000	190	11316864	430,040832	ASIL A
Spartan-3S5000	190	13271936	504,333568	ASIL A

Spartan-6 6SLX4	177	2731488	96,6946752	ASIL B-C
Spartan-6 6SLX9	177	2742528	97,0854912	ASIL B-C
Spartan-6 6SLX16	177	3731264	132,0867456	ASIL A
Spartan-6 6SLX25	177	6440432	227,9912928	ASIL A
Spartan-6 6SLX25T	177	6440432	227,9912928	ASIL A

Hypothèse 3 : 15% des bits de configurations sont des bits critiques

Product Family	Real-Time SER per Event	Nombre de bits	Nombre de FIT	Niveau ASIL
Kintex UltraScale KU025	31	128055264	595,4569776	ASIL A
Kintex UltraScale KU035	31	128055265	595,4569823	ASIL A
Kintex UltraScale KU040	31	128055266	595,4569869	ASIL A
Kintex UltraScale KU060	31	192999264	897,4465776	ASIL A
Kintex UltraScale KU085	31	386012288	1794,957139	
Kintex UltraScale KU095	31	286746912	1333,373141	
Kintex UltraScale KU0115	31	386012288	1794,957139	
Virtex UltraScale VU065	31	200713824	933,3192816	ASIL A

Virtex UltraScale VU080	31	286746912	1333,373141	
Virtex UltraScale VU095	31	286746913	1333,373145	
Virtex UltraScale VU125	31	401441408	1866,702547	
Virtex UltraScale VU160	31	602156064	2800,025698	
Virtex UltraScale VU190	31	602156064	2800,025698	
Virtex UltraScale VU440	31	1031731104	4797,549634	
Kintex UltraScale+ KU3P	5	123449056	92,586792	ASIL B-C
Kintex UltraScale+ KU5P	5	123449056	92,586792	ASIL B-C
Kintex UltraScale+ KU9P	5	212086240	159,06468	ASIL A
Kintex UltraScale+ KU11P	5	188647264	141,485448	ASIL A
Kintex UltraScale+ KU13P	5	229605952	172,204464	ASIL A
Kintex UltraScale+ KU15P	5	290744896	218,058672	ASIL A
Virtex UltraScale+ VU3P	5	213752800	160,3146	ASIL A

Virtex UltraScale+ VU5P	5	427519232	320,639424	ASIL A
Virtex UltraScale+ VU7P	5	427519232	320,639424	ASIL A
Virtex UltraScale+ VU9P	5	641272864	480,954648	ASIL A
Virtex UltraScale+ VU11P	5	679913248	509,934936	ASIL A
Virtex UltraScale+ VU13P	5	906547008	679,910256	ASIL A
Virtex UltraScale+ VU27P	5	906547008	679,910256	ASIL A
Virtex UltraScale+ VU29P	5	906547008	679,910256	ASIL A
Virtex UltraScale+ VU31P	5	226632928	169,974696	ASIL A
Virtex UltraScale+ VU33P	5	226632928	169,974696	ASIL A
Virtex UltraScale+ VU35P	5	453279488	339,959616	ASIL A
Virtex UltraScale+ VU37P	5	679913248	509,934936	ASIL A
Virtex-5 XC5VLX20T	165	6251200	154,7172	ASIL A
Virtex-5 XC5VLX30T	165	9371136	231,935616	ASIL A
Virtex-5 XC5VLX50T	165	14052352	347,795712	ASIL A
Virtex-5 XC5VLX85T	165	23341312	577,697472	ASIL A
Virtex-5 XC5VLX110T	165	31118848	770,191488	ASIL A

Virtex-5 XC5VLX155T	165	43042304	1065,297024	
Virtex-5 XC5VLX220T	165	55133696	1364,558976	
Virtex-5 XC5VLX330T	165	82696192	2046,730752	
Virtex-5 XC5VSX35T	165	13349120	330,39072	ASIL A
Virtex-5 XC5VSX50T	165	20019328	495,478368	ASIL A
Virtex-5 XC5VSX95T	165	35716096	883,973376	ASIL A
Spartan-7 7S6	76	4310752	49,1425728	ASIL B-C
Spartan-7 7S15	76	4310752	49,1425728	ASIL B-C
Spartan-7 7S25	76	9934432	113,2525248	ASIL A
Spartan-7 7S50	76	17536096	199,9114944	ASIL A
Spartan-7 7S75	76	29494496	336,2372544	ASIL A
Spartan-7 7S100	76	29494496	336,2372544	ASIL A
Artix -7 7A12T	76	9934432	113,2525248	ASIL A
Artix -7 7A15T	76	17536096	199,9114944	ASIL A
Artix -7 7A25T	76	9934432	113,2525248	ASIL A
Artix -7 7A35T	76	17536096	199,9114944	ASIL A
Artix -7 7A50T	76	17536096	199,9114944	ASIL A
Artix -7 7A75T	76	30606304	348,9118656	ASIL A
Artix -7 7A100T	76	30606305	348,911877	ASIL A
Artix -7 7A200T	76	77845216	887,4354624	ASIL A
Kintex-7 7K70T	50	24090592	180,67944	ASIL A
Kintex-7 7K160T	50	53540576	401,55432	ASIL A
Kintex-7 7K325T	50	91548896	686,61672	ASIL A
Kintex-7 7K355T	50	112414688	843,11016	ASIL A
Kintex-7 7K410T	50	127023328	952,67496	ASIL A
Kintex-7 7K420T	50	149880032	1124,10024	
Kintex-7 7K480T	50	149880032	1124,10024	
Virtex-7 7V585T	50	161398880	1210,4916	

Virtex-7 7V2000T	50	447337216	3355,02912	
Virtex-7 7VX330T	50	111238240	834,2868	ASIL A
Virtex-7 7VX485T	50	137934560	1034,5092	
Virtex-7 7VX550T	50	162187488	1216,40616	
Virtex-7 7VX550T	50	229878496	1724,08872	
Spartan-3S50A/AN	190	437312	12,463392	ASIL B-C
Spartan-3S200A/AN	190	1196128	34,089648	ASIL B-C
Spartan-3S400A/AN	190	1886560	53,76696	ASIL B-C
Spartan-3S700A/AN	190	2732640	77,88024	ASIL B-C
Spartan-3S1400A/AN	190	4755296	135,525936	ASIL A
Spartan-3SD1800A	190	8197280	233,62248	ASIL A
Spartan-3SD3400A	190	11718304	333,971664	ASIL A
Spartan-3S100E	190	581344	16,568304	ASIL B-C
Spartan-3S250E	190	1353728	38,581248	ASIL B-C
Spartan-3S500E	190	2270208	64,700928	ASIL B-C
Spartan-3S1200E	190	3841184	109,473744	ASIL A
Spartan-3S1600E	190	5969696	170,136336	ASIL A
Spartan-3S50	190	439264	12,519024	ASIL B-C
Spartan-3S200	190	1047616	29,857056	ASIL B-C
Spartan-3S400	190	1699136	48,425376	ASIL B-C
Spartan-3S1000	190	3223488	91,869408	ASIL B-C
Spartan-3S1500	190	5214784	148,621344	ASIL A
Spartan-3S2000	190	7673024	218,681184	ASIL A
Spartan-3S4000	190	11316864	322,530624	ASIL A
Spartan-3S5000	190	13271936	378,250176	ASIL A
Spartan-6 6SLX4	177	2731488	72,5210064	ASIL B-C

Spartan-6 6SLX9	177	2742528	72,8141184	ASIL B-C
Spartan-6 6SLX16	177	3731264	99,0650592	ASIL B-C
Spartan-6 6SLX25	177	6440432	170,9934696	ASIL A
Spartan-6 6SLX25T	177	6440432	170,9934696	ASIL A

Hypothèse 4 : 10% des bits de configurations sont des bits critiques

Product Family	Real-Time SER per Event	Nombre de bits	Nombre de FIT	Niveau ASIL
Kintex UltraScale KU025	31	128055264	396,9713184	ASIL A
Kintex UltraScale KU035	31	128055265	396,9713215	ASIL A
Kintex UltraScale KU040	31	128055266	396,9713246	ASIL A
Kintex UltraScale KU060	31	192999264	598,2977184	ASIL A
Kintex UltraScale KU085	31	386012288	1196,638093	
Kintex UltraScale KU095	31	286746912	888,9154272	ASIL A
Kintex UltraScale KU0115	31	386012288	1196,638093	
Virtex UltraScale VU065	31	200713824	622,2128544	ASIL A
Virtex UltraScale VU080	31	286746912	888,9154272	ASIL A
Virtex UltraScale VU095	31	286746913	888,9154303	ASIL A

Virtex UltraScale VU125	31	401441408	1244,468365	
Virtex UltraScale VU160	31	602156064	1866,683798	
Virtex UltraScale VU190	31	602156064	1866,683798	
Virtex UltraScale VU440	31	1031731104	3198,366422	
Kintex UltraScale+ KU3P	5	123449056	61,724528	ASIL B-C
Kintex UltraScale+ KU5P	5	123449056	61,724528	ASIL B-C
Kintex UltraScale+ KU9P	5	212086240	106,04312	ASIL A
Kintex UltraScale+ KU11P	5	188647264	94,323632	ASIL B-C
Kintex UltraScale+ KU13P	5	229605952	114,802976	ASIL A
Kintex UltraScale+ KU15P	5	290744896	145,372448	ASIL A
Virtex UltraScale+ VU3P	5	213752800	106,8764	ASIL A
Virtex UltraScale+ VU5P	5	427519232	213,759616	ASIL A
Virtex UltraScale+ VU7P	5	427519232	213,759616	ASIL A
Virtex UltraScale+ VU9P	5	641272864	320,636432	ASIL A
Virtex UltraScale+ VU11P	5	679913248	339,956624	ASIL A

Virtex UltraScale+ VU13P	5	906547008	453,273504	ASIL A
Virtex UltraScale+ VU27P	5	906547008	453,273504	ASIL A
Virtex UltraScale+ VU29P	5	906547008	453,273504	ASIL A
Virtex UltraScale+ VU31P	5	226632928	113,316464	ASIL A
Virtex UltraScale+ VU33P	5	226632928	113,316464	ASIL A
Virtex UltraScale+ VU35P	5	453279488	226,639744	ASIL A
Virtex UltraScale+ VU37P	5	679913248	339,956624	ASIL A
Virtex-5 XC5VLX20T	165	6251200	103,1448	ASIL A
Virtex-5 XC5VLX30T	165	9371136	154,623744	ASIL A
Virtex-5 XC5VLX50T	165	14052352	231,863808	ASIL A
Virtex-5 XC5VLX85T	165	23341312	385,131648	ASIL A
Virtex-5 XC5VLX110T	165	31118848	513,460992	ASIL A
Virtex-5 XC5VLX155T	165	43042304	710,198016	ASIL A
Virtex-5 XC5VLX220T	165	55133696	909,705984	ASIL A
Virtex-5 XC5VLX330T	165	82696192	1364,487168	
Virtex-5 XC5VSX35T	165	13349120	220,26048	ASIL A
Virtex-5 XC5VSX50T	165	20019328	330,318912	ASIL A

Virtex-5 XC5VSX95T	165	35716096	589,315584	ASIL A
Spartan-7 7S6	76	4310752	32,7617152	ASIL B-C
Spartan-7 7S15	76	4310752	32,7617152	ASIL B-C
Spartan-7 7S25	76	9934432	75,5016832	ASIL B-C
Spartan-7 7S50	76	17536096	133,2743296	ASIL A
Spartan-7 7S75	76	29494496	224,1581696	ASIL A
Spartan-7 7S100	76	29494496	224,1581696	ASIL A
Artix -7 7A12T	76	9934432	75,5016832	ASIL A
Artix -7 7A15T	76	17536096	133,2743296	ASIL A
Artix -7 7A25T	76	9934432	75,5016832	ASIL A
Artix -7 7A35T	76	17536096	133,2743296	ASIL A
Artix -7 7A50T	76	17536096	133,2743296	ASIL A
Artix -7 7A75T	76	30606304	232,6079104	ASIL A
Artix -7 7A100T	76	30606305	232,607918	ASIL A
Artix -7 7A200T	76	77845216	591,6236416	ASIL A
Kintex-7 7K70T	50	24090592	120,45296	ASIL A
Kintex-7 7K160T	50	53540576	267,70288	ASIL A
Kintex-7 7K325T	50	91548896	457,74448	ASIL A
Kintex-7 7K355T	50	112414688	562,07344	ASIL A
Kintex-7 7K410T	50	127023328	635,11664	ASIL A
Kintex-7 7K420T	50	149880032	749,40016	ASIL A
Kintex-7 7K480T	50	149880032	749,40016	ASIL A
Virtex-7 7V585T	50	161398880	806,9944	ASIL A
Virtex-7 7V2000T	50	447337216	2236,68608	ASIL A
Virtex-7 7VX330T	50	111238240	556,1912	ASIL A
Virtex-7 7VX485T	50	137934560	689,6728	ASIL A
Virtex-7 7VX550T	50	162187488	810,93744	ASIL A
Virtex-7 7VX550T	50	229878496	1149,39248	
Spartan-3S50A/AN	190	437312	8,308928	ASIL D
Spartan-3S200A/AN	190	1196128	22,726432	ASIL B-C
Spartan-3S400A/AN	190	1886560	35,84464	ASIL B-C

Spartan-3S700A/AN	190	2732640	51,92016	ASIL B-C
Spartan-3S1400A/AN	190	4755296	90,350624	ASIL B-C
Spartan-3SD1800A	190	8197280	155,74832	ASIL A
Spartan-3SD3400A	190	11718304	222,647776	ASIL A
Spartan-3S100E	190	581344	11,045536	ASIL B-C
Spartan-3S250E	190	1353728	25,720832	ASIL B-C
Spartan-3S500E	190	2270208	43,133952	ASIL B-C
Spartan-3S1200E	190	3841184	72,982496	ASIL B-C
Spartan-3S1600E	190	5969696	113,424224	ASIL A
Spartan-3S50	190	439264	8,346016	ASIL D
Spartan-3S200	190	1047616	19,904704	ASIL B-C
Spartan-3S400	190	1699136	32,283584	ASIL B-C
Spartan-3S1000	190	3223488	61,246272	ASIL B-C
Spartan-3S1500	190	5214784	99,080896	ASIL B-C
Spartan-3S2000	190	7673024	145,787456	ASIL A
Spartan-3S4000	190	11316864	215,020416	ASIL A
Spartan-3S5000	190	13271936	252,166784	ASIL A
Spartan-6 6SLX4	177	2731488	48,3473376	ASIL B-C
Spartan-6 6SLX9	177	2742528	48,5427456	ASIL B-C
Spartan-6 6SLX16	177	3731264	66,0433728	ASIL B-C
Spartan-6 6SLX25	177	6440432	113,9956464	ASIL A
Spartan-6 6SLX25T	177	6440432	113,9956464	ASIL A

ANNEXE II

INJECTION DE PANNES SUR L'ENTRÉE "VIDEO_IN =22 " S@1

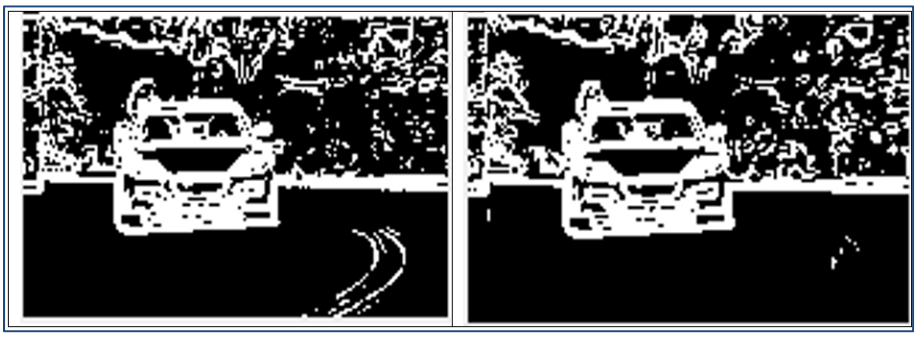

Figure-A II-1 Résultat d'injection de pannes sur Video_in 22 S@1 (droite de la figure) et résultat de la simulation de référence sans panne (à gauche de la figure)

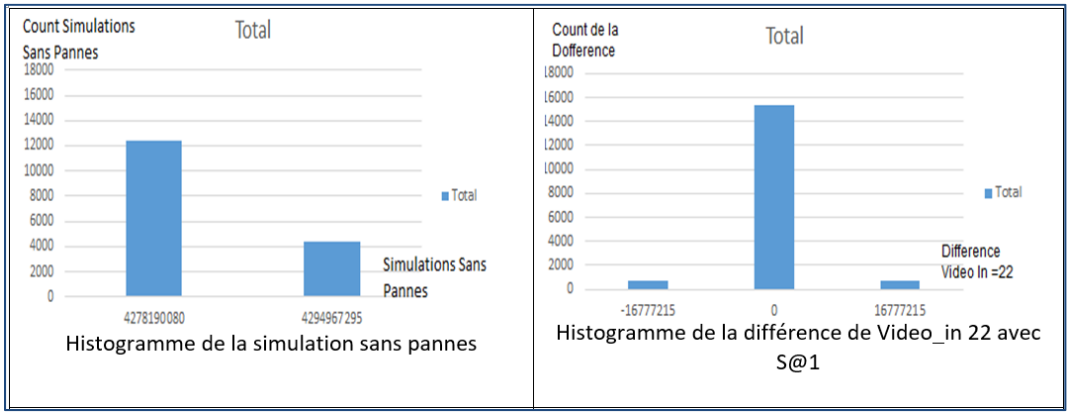

Figure-A II-2 Histogramme de la différence d'injection de pannes (droite de la figure) et Histogramme de la simulation de référence sans panne (à gauche de la figure)

INJECTION DE PANNES SUR L'ENTRÉE "VIDEO_IN =23 " S@1

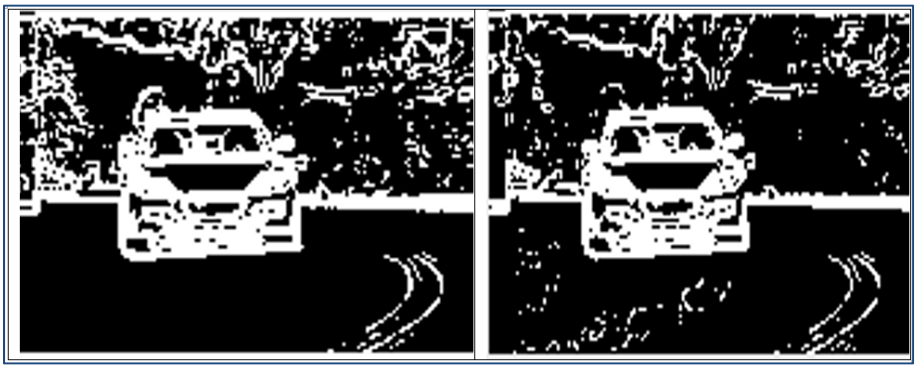

Figure-A II-3 Résultat d'injection de pannes sur Video_in=23 S@1 (droite de la figure) et résultat de la simulation de référence sans panne (à gauche de la figure)

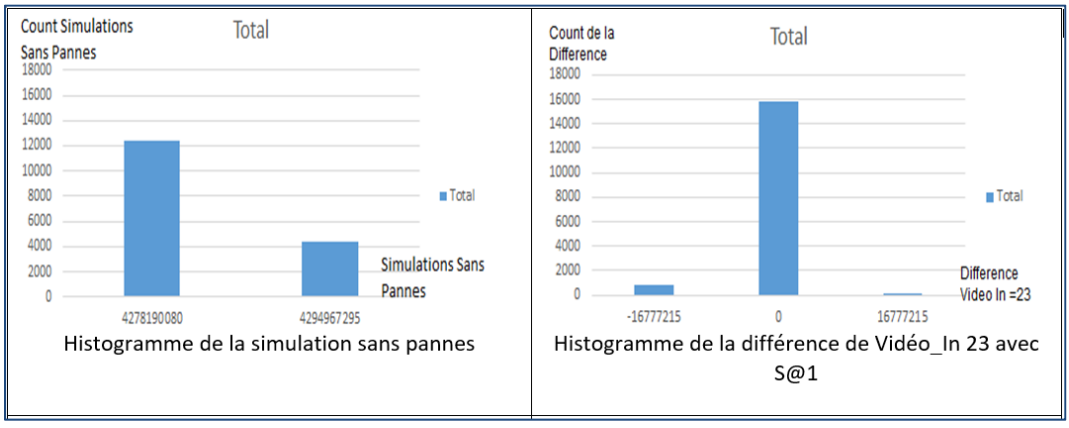

Figure-A II-4 Histogramme de la différence d'injection de pannes (droite de la figure) et Histogramme de la simulation de référence sans panne (à gauche de la figure)

ANNEXE III

INJECTION DE PANNES SUR L'ENTRÉE "THRESHOLD =5 " S@1

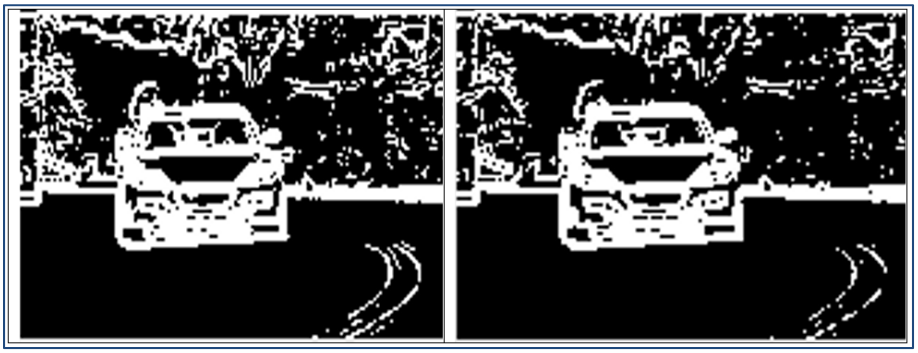

Figure-A III-1 Résultat d'injection de pannes sur Threshold =5 S@1 (droite de la figure) et résultat de la simulation de référence sans panne (à gauche de la figure)

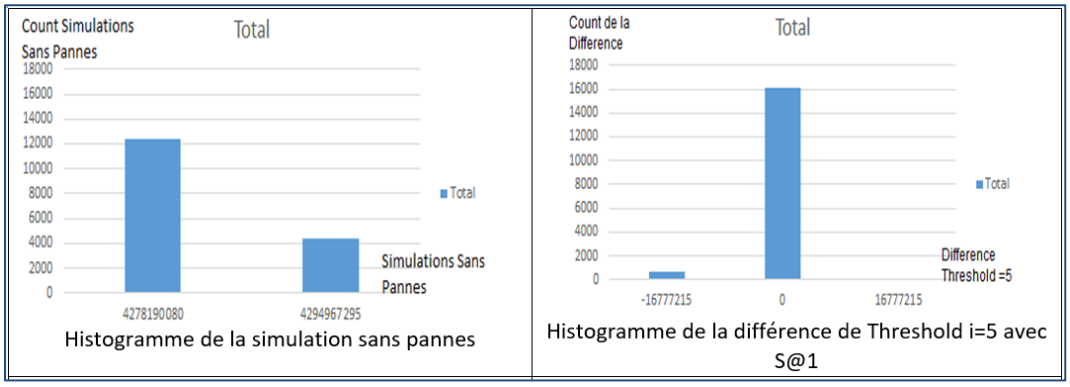

Figure-A III-2 Histogramme de la différence d'injection de pannes (droite de la figure) et Histogramme de la simulation de référence sans panne (à gauche de la figure)

INJECTION DE PANNES SUR L'ENTRÉE "THRESHOLD =7 " S@0

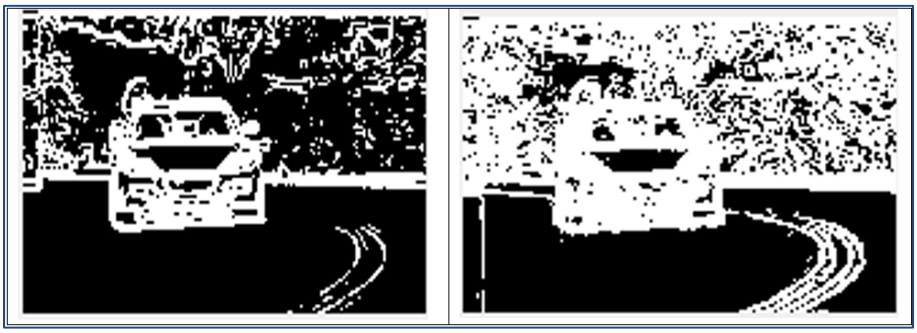

Figure-A III-3 Résultat d'injection de pannes sur Threshold= 7 S@0 (droite de la figure) et résultat de la simulation de référence sans panne (à gauche de la figure)

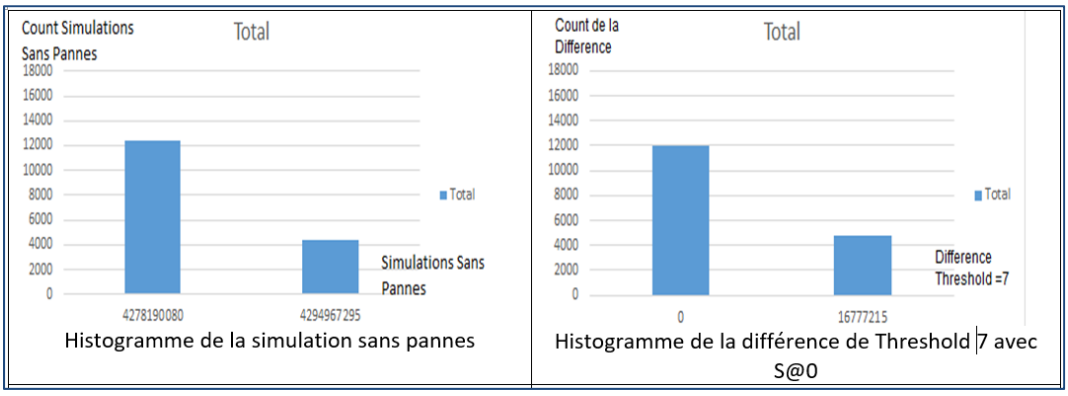

Figure-A III-4 Histogramme de la différence d'injection de pannes (droite de la figure) et Histogramme de la simulation de référence sans panne (à gauche de la figure)

ANNEXE IV

INJECTION DE PANNES AU NIVEAU DE LA SORTIE "VIDEO_OUT =8 " S@0

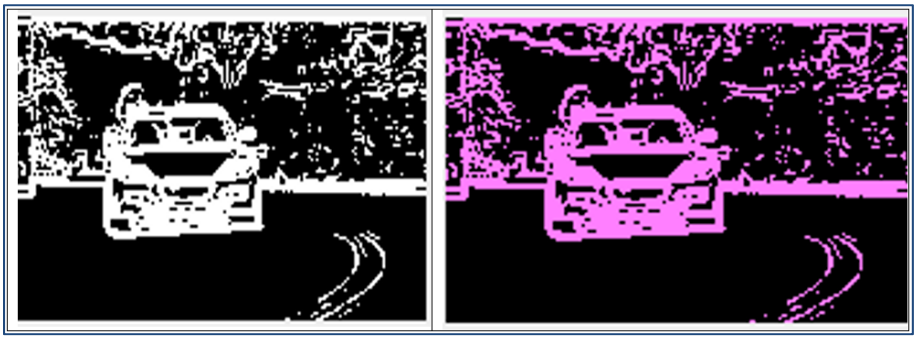

Figure-A IV-1 Résultat d'injection de pannes sur Video_out 8 S@0 (droite de la figure) et résultat de la simulation de référence sans panne (à gauche de la figure)

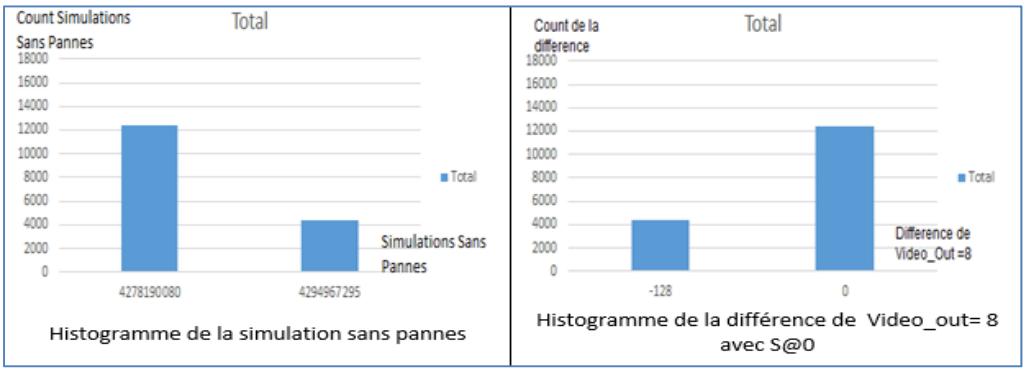

Figure-A IV-2 Histogramme de la différence d'injection de pannes (droite de la figure) et Histogramme de la simulation de référence sans panne (à gauche de la figure)

INJECTION DE PANNES AU NIVEAU DE LA SORTIE "VIDEO_OUT =9 " S@0

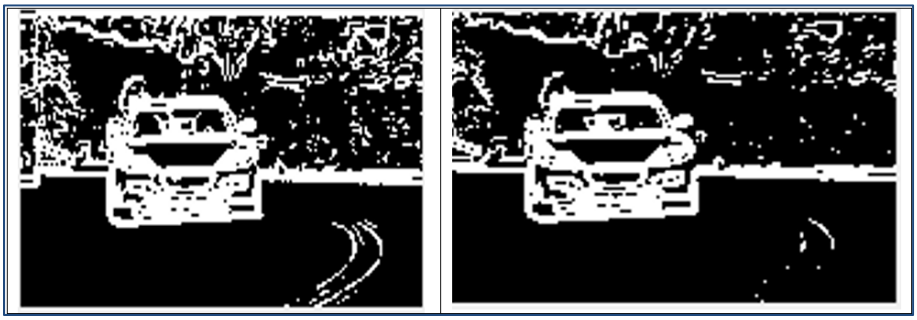

Figure-A IV-3 Résultat d'injection de pannes sur Video_out 9 S@0 (droite de la figure) et résultat de la simulation de référence sans panne (à gauche de la figure)

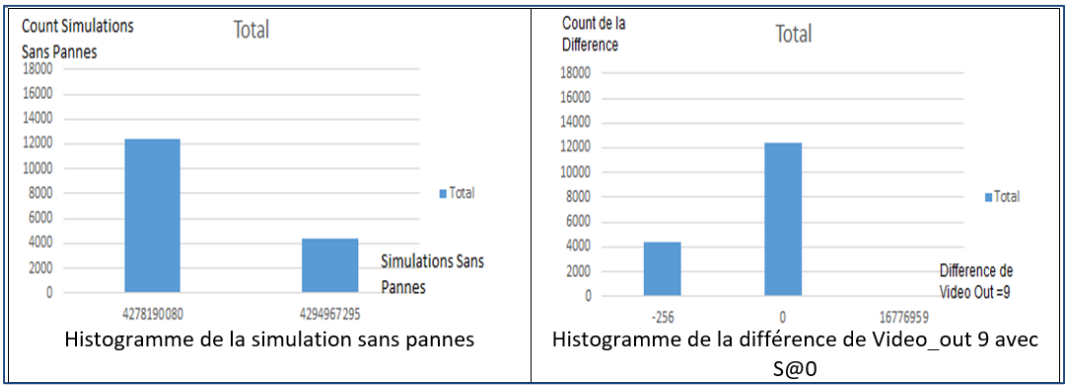

Figure-A IV-4 Histogramme de la différence d'injection de pannes (droite de la figure) et Histogramme de la simulation de référence sans panne (à gauche de la figure)

ANNEXE V

EFFET DES SABOTEURS SUR LA TRANSFORMENT DE HOUGH

Le tableau ci-dessous illustre les effets des saboteurs, ajoutés aux entrées et sortie du bloc synthétisable « Sobel_HW », sur le bloc en aval.

Tableau-A V-4 L'effet des saboteurs sur le bloc en aval

Saboteurs	**Le nombre de différences**
Sobel_Enable	30052
Background_Color	30062
Show_Gradient	30079
Threshold 1 S@1	12796
Threshold 2 S@1	12591
Threshold 3 S@0	18352
Threshold 4 S@1	22481
Threshold 5 S@1	25022
Threshold 6 S@0	26799
Threshold 7 S@0	28387
Threshold 8 S@1	26482
Video_in 1 S@0	10498
Video_in 1 S@1	11505
Video_in 2 S@0	12338
Video_in 2 S@1	12521
Video_in 3 S@0	15460
Video_in 3 S@1	15487
Video_in 4 S@0	19690
Video_in 4 S@1	19883
Video_in 5 S@0	24583
Video_in 5 S@1	24559
Video_in 6 S@0	27056
Video_in 6 S@1	27062
Video_in 7 S@0	26396

Table-A V-5 L'effet des saboteurs sur le bloc en aval (suite)

Video_in 7 S@1	26307
Video_in 8 S@0	19634
Video_in 8 S@1	20976
Video_in 9 S@0	4788
Video_in 9 S@1	4130
Video_in 10 S@0	6483
Video_in 10 S@1	7107
Video_in 11 S@0	10180
Video_in 11 S@1	9498
Video_in 12 S@0	10157
Video_in 12 S@1	10548
Video_in 13 S@0	16499
Video_in 13 S@1	16592
Video_in 14 S@0	22303
Video_in 14 S@1	22285
Video_in 15 S@0	22285
Video_in 15 S@1	23038
Video_in 16 S@0	7708
Video_in 16 S@1	12263
Video_in 17 S@0	8783
Video_in 17 S@1	6779
Video_in 18 S@0	8978
Video_in 18 S@1	9542
Video_in 19 S@0	11854
Video_in 19 S@1	12566
Video_in 20 S@0	16894
Video_in 20 S@1	16624
Video_in 21 S@0	21103
Video_in 21 S@1	21065
Video_in 22S@0	23971
Video_in 22 S@1	24142
Video_in 23 S@0	25401
Video_in 23 S@1	25369

Video_in 24 S@1	27017

Table-A V-4 L'effet des saboteurs sur le bloc en aval (suite)

Video_in 25 S@1	16894
Video_out 1 S@1	30052
Video_out 2 S@1	30052
Video_out 3 S@1	30052
Video_out 4 S@1	30052
Video_out 5 S@1	30052
Video_out 6 S@1	30052
Video_out 7 S@1	30052
Video_out 8 S@1	30052
Video_out 8 S@1	30052
Video_out 8 S@1	30052
Video_out 9 S@1	30052
Video_out 10 S@1	30052
Video_out 11 S@1	30052
Video_out 12 S@1	30052
Video_out 13 S@1	30052
Video_out 14 S@1	30052
Video_out 15 S@1	30052
Video_out 16 S@1	30052
Video_out 17 S@1	30052
Video_out 18 S@1	30052
Video_out 19 S@1	30052
Video_out 20 S@1	30052
Video_out 21 S@1	30052
Video_out 22 S@1	30052
Video_out 23 S@1	30052
Video_out 24 S@1	30052
Video_out 25 S@0	27017
Video_out 26 S@0	27017
Video_out 27 S@0	27017
Video_out 28 S@0	27017
Video_out 29 S@0	27017
Video_out 30 S@0	27017

Table-A V-6 L'effet des saboteurs sur le bloc en aval (suite)

Video_out 31 S@0	27017
Reset	27017
Clock	27017

ANNEXE VI

CALCUL DE Rsab POUR LE FILTRE DE SOBEL ET LA TRANSFORMEE DE HOUGH

Filtre de Sobel :

Similairement, RsabSobel (j) est la racine carrée de *RELsab$_{Sobel(j)}$,* qui représente l'erreur quadratique moyenne relative de la séquence produite par un saboteur j par rapport à une séquence sans pannes. RsabSobel *(j)* est égal à :

$$Rsab_{Sobel(j)} = \sqrt{RELsab_{Sobel(j)}} \qquad \text{(A VI-1)}$$

Où :

$$RELsab_{Sobel(j)} = \frac{ABSsab_{Sobel}(j)}{MoyPixel_{Sobel}} \qquad \text{(A VI-2)}$$

Et où *ABSsab$_{Sobel(j)}$,* l'erreur quadratique moyenne(absolue) de la séquence produite par un saboteur *j* par rapport à une séquence sans pannes, est égale à :

$$ABSsab_{Sobel(j)} = \frac{\sum_{k=1}^{16800} \left(Pixel_{Sobel}(k) - Pixel_{SansPannes}(k)\right)^2}{Ptot_{Sobel}} \qquad \text{(A VI-3)}$$

La transformée de Hough :

RsabHough(j) est la racine carrée de *RELsab$_{Hough(j)}$,* qui représente l'erreur quadratique moyenne relative de la séquence produite par un saboteur j par rapport à une séquence sans pannes. RsabHough (j) est égal à :

$$\mathrm{Rsab}_{\mathrm{Hough(j)}}=\sqrt{\mathrm{RELsab}_{\mathrm{Hough(j)}}} \qquad \text{(A VI-4)}$$

Où :

$$\mathrm{RELsab}_{\mathrm{Hough(j)}}=\frac{\mathrm{ABSsab}_{\mathrm{Hough}}(\mathrm{j})}{\mathrm{MoyPixel}_{\mathrm{Hough}}} \qquad \text{(A VI-5)}$$

Et où *ABSsab$_{Hough(j)}$,* l'erreur quadratique moyenne(absolue) de la séquence produite par un saboteur *j* par rapport à une séquence sans pannes, est égale à :

$$\mathrm{ABSsab}_{\mathrm{Hough(j)}}=\frac{\sum_{k=1}^{68580}\left(\mathrm{Pixel}_{\mathrm{Hough}}(k)-\mathrm{Pixel}_{\mathrm{SansPannes}}(k)\right)^{2}}{\mathrm{Ptot}_{\mathrm{Hough}}} \qquad \text{(A VI-6)}$$

ANNEXE VII

CODE DE VÉRIFICATION DE LA COUVERTURE DES ÉMULATIONS PAR LES SABOTEURS

Ce code est dédié pour la vérification de la couverture des différentes émulations par les saboteurs. Il est composé d'un programme permettant principalement d'ouvrir les différents fichiers Excel des résultats en question. Cet algorithme fait ensuite le calcul de la différence entre le fichier des émulations par rapport au fichier sans pannes et le fichier des saboteurs par rapport au fichier sans pannes et la racine carrée de ces différentes émulations. Enfin une comparaison est faite et le choix du saboteur adéquat est pris.

```
import pandas as pd
import numpy as np
import math

fichier_emulation = pd.read_csv ('C:/compa/emulation_sobel.csv', sep=',')
fichier_saboteur = pd.read_csv("C:/compa/ saboteur_sobel.csv", sep=',')
fichier_sansPannes = pd.read_csv("C:/compa/sans pannes.csv", sep=',')
text_file = open ("resultats.txt", "w")

for col1 in fichier_emulation.columns:
    var = -1
 NemuSobel= list(fichier_emulation[((fichier_emulation[col1] ! = fichier_sansPannes['Sans Pannes']) & (fichier_emulation[col1] > fichier_sansPannes['Sans Pannes'])) | ((fichier_emulation[col1] ! = fichier_sansPannes['Sans Pannes']) & (fichier_emulation[col1] < fichier_sansPannes['Sans Pannes'] ))].index)
        Diff_emulation_carree = (fichier_emulation[col1] - fichier_sansPannes['Sans Pannes']) * (fichier_emulation [col1] - fichier_sansPannes ['Sans Pannes'])
```

```
a = sum (Diff_emulation_carree) /16800
a1 = a/4282599092
RemuSobel = np.sqrt(a1)
        for col2 in f2.columns:
        NsabSobel= list(fichier_saboteur[((fichier_saboteur[col2] ! = fichier_sansPannes['Sans Pannes']) & (fichier_saboteur [col2] > fichier_sansPannes['Sans Pannes'])) | ((fichier_saboteur [col2] != fichier_sansPannes['Sans Pannes']) & (fichier_saboteur [col2] < fichier_sansPannes['Sans Pannes']))].index)
        Diff_saboteur_carree = (fichier_saboteur[col2] - fichier_sansPannes['Sans Pannes']) * (fichier_saboteur [col2] - fichier_sansPannes['Sans Pannes'])
        b = sum (Diff_saboteur_carree)/16800
        b1 = b/4282599092
        RsabSobel= np.sqrt(b1)

        if len (NemuSobel) == len (NsabSobel) and RemuSobel == RsabSobel:
            print(fichier_emulation[col1].name +' \t'+ fichier_saboteur[col2].name + '\t'+'identique')
            text_file. write (fichier_emulation[col1].name +' \t'+ fichier_saboteur [col2].name +' \t' + 'identique' + '\n' )
            break
        elif len (NemuSobel) > len (NsabSobel) and RemuSobel >= RsabSobel:
    Cnom_Sobel = (len (NsabSobel) / len (NemuSobel)) *100
            if Cnom_Sobel >= 100:
                Cnom_Sobel = 100
            else:
                Cnom_Sobel = (len (NsabSobel)/len (NemuSobel)) *100

     Cint_Sobel = (RsabSobel / RemuSobel) *100
```

```
        if Cint_Sobel >= 100:
          Cint_Sobel = 100
        else:
            Cint_Sobel = Cint_Sobel

        Cglob_Sobel = min (Cnom _Sobel, Cint_Sobel)

        if Cglob_Sobel < 100:
            Cres_Sobel = Cglob_Sobel
        elif Cglob_Sobel == 100:
            D= np.sqrt((pow (Rsab-Remu,2)) +pow(len (sabo)-len (emu),2))
        if var > D or var == -1:
            var = D
            col2mini = col2
            print(fichier_emulation[col1].name+'\t'+str(RemuSobel)+'\t'+        fichier_saboteur
[col2mini].name+'\t'+ str(RsabSobel)+'\t'+str(len(D))+'\n')
            text_file.write(fichier_emulation[col1].name+'\t'+str(RemuSobel)+'\t'+
fichier_saboteur [col2mini].name+'\t'+ str(RsabSobel)+'\t'+str(len(D))+'\n')

text_file.close ()
```

CODE DE VÉRIFICATION DE LA COUVERTURE DES ÉMULATIONS PAR LES SABOTEURS POUR LA TRANSFORMEE DE HOUGH

```
import pandas as pd
import numpy as np
import math

fichier_emulation = pd.read_csv ('C:/compa/emulation_Hough.csv', sep=',')
fichier_saboteur = pd.read_csv("C:/compa/ saboteur_Hough.csv", sep=',')
fichier_sansPannes = pd.read_csv("C:/compa/sans pannes.csv", sep=',')
text_file = open ("resultats.txt", "w")

for col1 in fichier_emulation.columns:
    var = -1
 NemuHough = list(fichier_emulation[((fichier_emulation[col1] ! = fichier_sansPannes['Sans Pannes']) & (fichier_emulation[col1] > fichier_sansPannes['Sans Pannes'])) | ((fichier_emulation[col1] ! = fichier_sansPannes['Sans Pannes']) & (fichier_emulation[col1] < fichier_sansPannes['Sans Pannes'] ))].index)
        Diff_emulation_carree = (fichier_emulation[col1] - fichier_sansPannes['Sans Pannes']) * (fichier_emulation [col1] - fichier_sansPannes ['Sans Pannes'])
   a = sum (Diff_emulation_carree) /68580
   a1 = a/11.59
   RemuHough = np.sqrt(a1)
        for col2 in f2.columns:
       NsabHough = list(fichier_saboteur[((fichier_saboteur[col2] ! = fichier_sansPannes['Sans Pannes']) & (fichier_saboteur [col2] > fichier_sansPannes['Sans Pannes'])) | ((fichier_saboteur [col2] != fichier_sansPannes['Sans Pannes']) & (fichier_saboteur [col2] < fichier_sansPannes['Sans Pannes']))].index)
```

```
        Diff_saboteur_carree = (fichier_saboteur[col2] - fichier_sansPannes['Sans Pannes']) *
(fichier_saboteur [col2] - fichier_sansPannes['Sans Pannes'])
        b = sum (Diff_saboteur_carree)/68580
        b1 = b/11.59
        RsabHough= np.sqrt(b1)

        if len (NemuHough) == len (NsabHough) and RemuHough == RsabHough:
           print(fichier_emulation[col1].name     +'     \t'+     fichier_saboteur[col2].name     +
'\t'+'identique')
           text_file. write (fichier_emulation[col1].name +' \t'+ fichier_saboteur
[col2].name +' \t' + 'identique' + '\n' )
           break
        elif len (NemuHough) > len (NsabHough) and RemuHough>= RsabHough:
    Cnom_Hough = (len (NsabHough) / len (NemuHough)) *100
           if Cnom_Hough >= 100:
               Cnom_Hough = 100
           else:
              Cnom_Hough = (len (NsabHough)/len (NemuHough)) *100

     Cint_Hough = (RsabHough / RemuHough) *100
        if Cint_Hough >= 100:
           Cint_Hough = 100
        else:
            Cint_Hough = Cint_Hough

        Cglob_Hough = min (Cnom _Hough, Cint_Hough)

        if Cglob_Hough < 100:
            Cres_Hough = Cglob_Hough
```

```
        elif Cglob_Hough == 100:
            D= np.sqrt((pow (Rsab-Remu,2)) +pow(len (sabo)-len (emu),2))
        if var > D or var == -1:
            var = D
            col2mini = col2
            print(fichier_emulation[col1].name+'\t'+str(RemuHough)+'\t'+      fichier_saboteur
[col2mini].name+'\t'+ str(RsabHough)+'\t'+str(len(D))+'\n')
            text_file.write(fichier_emulation[col1].name+'\t'+str(RemuSobel)+'\t'+
fichier_saboteur [col2mini].name+'\t'+ str(RsabHough)+'\t'+str(len(D))+'\n')

text_file.close ()
```

ANNEXE VIII

RÉSULTATS DE LA COUVERTURE AU NIVEAU DU FILTRE DE SOBEL

Tableau-A VIII-7 Couverture globale des émulations au niveau de Sobel avec des saboteurs

No d'émulations (i)	Nemu Sobel (i)	Remu Sobel (i)	Saboteurs couvrants les émulations	Nsab Sobel (j)	Rsab Sobel (j)	Cnom_ Sobel (%)	Cint _ Sobel (%)	Cres_ Sobel (%)
1	3	3,43	Video_in 25 S@1	3	3,43	100	100	100
2	182	26,68	Threshold 3 S@0	200	27,97	100	100	100
3	598	48,37	Video_in 21 S@1	616	49,09	100	100	100
4	118	21,49	Video_in 19 S@0	124	22,03	100	100	100
5	3038	109,02	Threshold 7 S@0	4755	136,39	100	100	100
6	65	15,95	Video_in 17 S@0	74	17,01	100	100	100
7	8255	179,71	Background_Color	16797	256,35	100	100	100
8	2904	106,59	Threshold 7 S@0	4755	136,39	100	100	100
9	718	53	Video_in 23 S@1	1028	63,42	100	100	100
10	1467	75,76	Threshold 6 S@0	1915	86,56	100	100	100
11	41	12,66	Video_in 10 S@0	49	13,85	100	100	100
12	1975	78,9	Video_in 7 S@1	2076	90,12	100	100	100
13	3985	124,43	Threshold 7 S@0	4755	136,39	100	100	100
14	140	23.4	Video_in 16 S@1	146	23,9	100	100	100
15	153	24,47	Threshold 3 S@0	200	27,97	100	100	100
16	479	43,29	Video_in 14 S@0	481	43,38	100	100	100
17	139	23,32	Video_in 16 S@1	146	23,9	100	100	100
18	4849	137,74	Background_Color	16797	256,35	100	100	100
19	1113	65,98	Video_in 22S@0	1280	70,76	100	100	100
20	3034	108,95	Threshold 7 S@0	4755	136,39	100	100	100
21	4317	129,96	Threshold 7 S@0	4755	136,39	100	100	100
22	1168	15374,74	Video_out 31 S@0	16800	16407,6	100	100	100
23	4565	133,64	Threshold 7 S@0	4755	136,39	100	100	100
24	41	12,66	Video_in 10 S@0	49	13,85	100	100	100
25	4249	128,39	Threshold 7 S@0	4755	136,39	100	100	100
26	544	46,13	Video_in 21 S@1	616	49,09	100	100	100
27	1113	65,98	Video_in 22S@0	1280	70,76	100	100	100
28	2280	3714,77	Video_out 29 S@0	16800	4101,87	100	100	100
29	158	24,86	Threshold 3 S@0	200	27,97	100	100	100
30	435	41,25	Video_in 4 S@0	446	41,77	100	100	100

Tableau-A VIII-5 Couverture globale des émulations au niveau de Sobel avec des saboteurs (suite)

31	2014	88,76	Video_in 7 S@1	2076	90,12	100	100	100
32	25	9,89	Video_in 9 S@1	30	10,83	100	100	100
33	3441	116,03	Threshold 7 S@0	4755	136,39	100	100	100
34	4540	133,27	Threshold 7 S@0	4755	136,39	100	100	100
35	4442	131,83	Threshold 7 S@0	4755	136,39	100	100	100
36	15	7,66	Video_in 9 S@1	30	10,83	100	100	100
37	11	6,56	Video_in 9 S@1	30	10,83	100	100	100
38	1988	88,19	Video_in 7 S@1	2076	90,12	100	100	100
39	1185	68,09	Video_in 22S@0	1280	70,76	100	100	100
40	4735	136,1	Threshold 7 S@0	4755	136,39	100	100	100
41	134	22,9	Video_in 16 S@1	146	23,9	100	100	100
42	2333	95,54	Threshold 8 S@1	2493	98,76	100	100	100
43	5615	148,21	Background_Color	16797	256,35	100	100	100
44	4	3,96	Video_in 9 S@1	30	10,83	100	100	100
45	210	28,66	Video_in 3 S@0	217	29,14	100	100	100
46	1366	73,1	Threshold 6 S@0	1915	86,56	100	100	100
47	38	12,19	Video_in 10 S@0	49	13,85	100	100	100
48	72	16,78	Video_in 17 S@0	74	17,01	100	100	100
49	4744	136.23	Threshold 7 S@0	4755	136,39	100	100	100
50	247	31,09	Video_in 20 S@1	247	31,09	100	100	100
51	4	3,96	Video_in 9 S@1	30	10,83	100	100	100
52	25	9,89	Video_in 9 S@1	30	10,83	100	100	100
53	476	42,99	Video_in 14 S@1	478	43,24	100	100	100
54	38	12,19	Video_in 10 S@0	49	13,85	100	100	100
55	128	22,38	Video_in 19 S@1	130	22,55	100	100	100
56	203	28,18	Video_in 3 S@0	217	29,14	100	100	100
57	105	20,27	Threshold 1 S@1	106	20,36	100	100	100
58	4181	127,89	Threshold 7 S@0	4755	136,39	100	100	100
59	4089	3,57	Threshold 7 S@0	4755	136,39	100	100	100
60	1421	1,78	Threshold 6 S@0	1915	86,56	100	100	100
61	5934	152,36	Background_Color	16797	256,35	100	100	100
62	197	27,76	Threshold 3 S@0	200	27,97	100	100	100
63	16	7,91	Video_in 9 S@1	30	10,83	100	100	100
64	27	10,28	Video_in 9 S@1	30	10,83	100	100	100
65	1061	64,43	Video_in 22S@0	1280	70,76	100	100	100
66	2632	24,19	Threshold 7 S@0	4755	136,39	100	100	100
67	788	55,52	Video_in 23 S@1	1028	63,42	100	100	100
68	7404	170,19	Background_Color	16797	256,35	100	100	100
69	6	4,84	Video_in 9 S@1	30	10,83	100	100	100
70	612	48,93	Video_in 21 S@1	616	49,09	100	100	100

Tableau-A VIII-5 Couverture globale des émulations au niveau de Sobel avec des saboteurs (suite)

71	27	10,28	Video_in 9 S@1	30	10,83	100	100	100
72	163	25,25	Threshold 3 S@0	200	27,97	100	100	100
73	2390	32,34	Threshold 8 S@1	2493	98,76	100	100	100
74	153	24,47	Threshold 3 S@0	200	27,97	100	100	100
75	1335	72,27	Threshold 6 S@0	1915	86,56	100	100	100
76	5934	152,37	Background_Color	16797	256,35	100	100	100
77	74	7,31	Video_in 17 S@0	74	17,01	100	100	100
78	1	1,98	Video_in 25 S@1	3	343	100	100	100
79	33	11,36	Video_in 9 S@0	35	11,7	100	100	100
80	357	37,37	Threshold 4 S@1	370	38,05	100	100	100
81	3770	93,98	Threshold 7 S@0	4755	136,39	100	100	100
82	210	28,66	Video_in 3 S@0	217	29,14	100	100	100
83	895	59,17	Video_in 23 S@1	1028	63,42	100	100	100
84	4507	132,79	Threshold 7 S@0	4755	136,39	100	100	100
85	2144	91,58	Video_in 6 S@1	2315	95,17	100	100	100
86	1362	72,99	Threshold 6 S@0	1915	86,56	100	100	100
87	246	31,02	Video_in 20 S@1	247	31,09	100	100	100
88	35	11,7	Video_in 9 S@0	35	11,7	100	100	100
89	443	41,63	Video_in 4 S@0	446	41,77	100	100	100
90	2515	99,19	Threshold 7 S@0	4755	136,39	100	100	100
91	1245	69,79	Video_in 22S@0	1280	70,76	100	100	100
92	20	8,85	Video_in 9 S@1	30	10,83	100	100	100
93	767	54,78	Video_in 23 S@1	1028	63,42	100	100	100
94	2420	97,3	Threshold 8 S@1	2493	98,76	100	100	100
95	2616	39,57	Threshold 7 S@0	4755	136,39	100	100	100
96	1061	64,43	Video_in 22S@0	1280	70,76	100	100	100
97	1245	69,79	Video_in 22S@0	1280	70,76	100	100	100
98	105	20,27	Threshold 1 S@1	106	20,36	100	100	100
99	74	7,31	Video_in 17 S@0	74	17,01	100	100	100
100	35	11,7	Video_in 9 S@0	35	11,7	100	100	100

ANNEXE IX

RÉSULTATS DE LA COUVERTURE AU NIVEAU DU BLOC EN AVAL : LA TRANSFORME DE HOUGH

Tableau-A IX-8 Une récapitulation de la couverture des émulations au niveau de la transformée de Hough

No d'émulations (i)	Nemu Hough (i)	Remu Hough (i)	Saboteurs couvrants les émulations	Nsab Hough (j)	Rsab Hough (j)	Cnom Hough (%)	Cint_ Hough (%)	Cres_ Hough (%)
1	412	0,03	Video_in 9 S@1	4130	0,08	100	100	100
2	13327	0,22	Video_in 3 S@0	15460	0,23	100	100	100
3	20544	0,4	Video_in 8 S@1	20976	0,64	100	100	100
4	11175	0,24	Video_in 13 S@0	16499	0,25	100	100	100
5	22763	5,07	Video_in 24 S@1	27017	6,36	100	100	100
6	8319	0,12	Video_in 17 S@0	8783	0,13	100	100	100
7	29790	9,89	Sobel_Enable	30052	15,91	100	100	100
8	23450	2,85	Threshold 8 S@1	26482	3,57	100	100	100
9	24506	1,11	Video_in 23 S@1	25369	1,28	100	100	100
10	24442	2,42	Threshold 8 S@1	26482	3,57	100	100	100
11	5806	0,1	Video_in 10 S@0	6483	0,11	100	100	100
12	28816	2,65	Sobel_Enable	30052	15,91	100	100	100
13	25529	4,7	Video_in 24 S@1	27017	6,36	100	100	100
14	6996	0,23	Video_in 16 S@1	12263	0,23	100	100	100
15	13864	0,19	Video_in 3 S@0	15460	0,23	100	100	100
16	19311	0,41	Video_in 8 S@0	19634	0,58	100	100	100
17	13037	0,18	Video_in 3 S@0	15460	0,23	100	100	100
18	25555	3,12	Threshold 8 S@1	26482	3,57	100	100	100
19	23563	1,69	Threshold 8 S@1	26482	3,57	100	100	100
20	22756	4,73	Video_in 24 S@1	27017	6,36	100	100	100
21	24579	5,75	Video_in 24 S@1	27017	6,36	100	100	100
22	14556	0,43	Video_in 8 S@0	19634	0,58	100	100	100
23	24969	4,29	Video_in 24 S@1	27017	6,36	100	100	100
24	6012	0,1	Video_in 10 S@0	6483	0,11	100	100	100
25	24079	6,64	Threshold 7 S@0	28387	6,64	100	100	100
26	12535	1,21	Video_in 23 S@1	25369	1,28	100	100	100

27	14556	0,43	Video_in 8 S@0	19634	0,58	100	100	100
28	20297	2,51	Threshold 8 S@1	26482	3,57	100	100	100

Tableau-A IX-6 Une récapitulation de la couverture des émulations au niveau de la transformée de Hough (suite)

29	9050	0,37	Video_in 8 S@0	19634	0,58	100	100	100
30	15359	0,63	Video_in 8 S@1	20976	0,64	100	100	100
31	20810	2,46	Threshold 8 S@1	26482	3,57	100	100	100
32	2766	0,08	Video_in 9 S@1	4130	0,08	100	100	100
33	23451	3,96	Video_in 24 S@1	27017	6,36	100	100	100
34	29651	5,81	Sobel_Enable	30052	15,91	100	100	100
35	25007	3,09	Threshold 8 S@1	26482	3,57	100	100	100
36	1932	0,07	Video_in 9 S@1	4130	0,08	100	100	100
37	1752	0,05	Video_in 9 S@1	4130	0,08	100	100	100
38	21780	2,64	Threshold 8 S@1	26482	3,57	100	100	100
39	20315	1,42	Video_in 7 S@1	26307	1,66	100	100	100
40	26443	6,76	Sobel_Enable	30052	15,91	100	100	100
41	12229	0,18	Video_in 16 S@1	12263	0,23	100	100	100
42	24483	1,54	Video_in 7 S@1	26307	1,66	100	100	100
43	27720	7,66	Sobel_Enable	30052	15,91	100	100	100
44	700	0,03	Video_in 9 S@1	4130	0,08	100	100	100
45	15802	0,49	Video_in 8 S@0	19634	0,58	100	100	100
46	22058	2,14	Threshold 8 S@1	26482	3,57	100	100	100
47	6996	0,23	Video_in 16 S@1	12263	0,23	100	100	100
48	11175	0,24	Video_in 13 S@0	16499	0,25	100	100	100
49	19998	1,62	Video_in 7 S@1	26307	1,66	100	100	100
50	15233	0,57	Video_in 8 S@0	19634	0,58	100	100	100
51	700	0,03	Video_in 9 S@1	4130	0,08	100	100	100
52	2829	0,08	Video_in 9 S@1	4130	0,08	100	100	100
53	15359	0,63	Video_in 8 S@1	20976	0,64	100	100	100
54	22058	2,14	Threshold 8 S@1	26482	3,57	100	100	100
55	10706	0,23	Video_in 16 S@1	12263	0,23	100	100	100
56	15871	0,48	Video_in 8 S@0	19634	0,58	100	100	100
57	12198	0,28	Threshold 3 S@0	18352	0,34	100	100	100
58	24096	3,04	Threshold 8 S@1	26482	3,57	100	100	100
59	21546	6,79	Sobel_Enable	30052	15,91	100	100	100
60	21828	2,43	Threshold 8 S@1	26482	3,57	100	100	100
61	23669	8,92	Sobel_Enable	30052	15,91	100	100	100
62	10024	0,43	Video_in 8 S@0	19634	0,58	100	100	100
63	2301	0,07	Video_in 9 S@1	4130	0,08	100	100	100

64	3772	0,08	Video_in 9 S@1	4130	0,08	100	100	100
65	18075	1,52	Video_in 7 S@1	26307	1,66	100	100	100
66	27482	3,69	Threshold 7 S@0	28387	6,64	100	100	100
67	21498	1,45	Video_in 7 S@1	26307	1,66	100	100	100
68	29763	6,61	Sobel_Enable	30052	15,91	100	100	100

Tableau-A IX-6 Une récapitulation de la couverture des émulations au niveau de la transformée de Hough (suite)

69	780	0,03	Video_in 9 S@1	4130	0,08	100	100	100
70	20820	0,45	Video_in 8 S@1	20976	0,64	100	100	100
71	3673	0,08	Video_in 9 S@1	4130	0,08	100	100	100
72	10711	0,17	Video_in 19 S@0	11854	0,17	100	100	100
73	14010	0,2	Video_in 3 S@0	15460	0,23	100	100	100
74	10516	0,39	Video_in 8 S@0	19634	0,58	100	100	100
75	25283	1,12	Video_in 23 S@1	25369	1,28	100	100	100
76	23669	8,92	Sobel_Enable	30052	15,91	100	100	100
77	2618	0,06	Video_in 9 S@1	4130	0,08	100	100	100
78	180	0,02	Video_in 9 S@1	4130	0,08	100	100	100
79	4630	0,09	Video_in 10 S@0	6483	0,11	100	100	100
80	16609	0,54	Video_in 8 S@0	19634	0,58	100	100	100
81	21201	6,12	Video_in 24 S@1	27017	6,36	100	100	100
82	15802	0,49	Video_in 8 S@0	19634	0,58	100	100	100
83	22343	0,76	Video_in 15 S@1	23038	0,93	100	100	100
84	24395	3,18	Threshold 8 S@1	26482	3,57	100	100	100
85	26323	2,28	Threshold 8 S@1	26482	3,57	100	100	100
86	26000	1,96	Threshold 8 S@1	26482	3,57	100	100	100
87	16258	0,24	Video_in 13 S@0	16499	0,25	100	100	100
88	3917	0,11	Video_in 10 S@0	6483	0,11	100	100	100
89	17908	0,61	Video_in 8 S@1	20976	0,64	100	100	100
90	21768	2,43	Threshold 8 S@1	26482	3,57	100	100	100
91	24192	1,53	Video_in 7 S@1	26307	1,66	100	100	100
92	3342	`0,07	Video_in 9 S@1	4130	0,08	100	100	100
93	19238	4,37	Video_in 24 S@1	27017	6,36	100	100	100
94	23272	1,22	Video_in 23 S@1	25369	1,28	100	100	100
95	26241	1,86	Threshold 8 S@1	26482	3,57	100	100	100
96	18075	1,52	Video_in 7 S@1	26307	1,66	100	100	100
97	24192	1,53	Video_in 7 S@1	26307	1,66	100	100	100
98	12198	0,28	Threshold 3 S@0	18352	0,34	100	100	100

99	2618	0,06	Video_in 9 S@1	4130	0,08	100	100	100
100	3917	0,11	Video_in 10 S@0	6483	0,11	100	100	100

LISTE DE RÉFÉRENCES BIBILOGRAPHIQUES

Amaricai, Alexandru, Sergiu Nimara, Oana Boncalo, Jiaoyan Chen & Emanuel Popovici. (2014).Probabilistic gate level fault modeling for near and sub-threshold CMOS circuits. Dans *Digital System Design (DSD), 2014 17th Euromicro Conference on. p. 473-479.* IEEE.

Arlat, Jean. (1990). Validation de la sûreté de fonctionnement par injection de fautes : méthode, mise en oeuvre, application. Dans *Institut national polytechnique de Toulouse, 191 p.*

Artix-7 FPGA Configuration User Guide (v1.25) June 18, 2018.

B.Poornima, Y.Ramadevi & T.Sridevi .(2011). A Novel Threshold Based Edge Detection Algorithm. Dans *International Journal of Engineering Science and Technology.*

Bandarage Shehani Sanketha Rathnayake & Lochandaka Ranathunga. (2018). Lane Detection and Prediction under Hazy Situations for Autonomous Vehicle Navigation. Dans *2018 International Conference on Advances in ICT for Emerging Regions (ICTer)*. IEEE.

Baraza, J. C., Gracia, J., Gil, D., & Gil, P. J. (2000). A prototype of a VHDL-based fault injection tool. Dans *Proceedings IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems* (pp. 396-404). IEEE.

Berrojo, Luis, Isabel González, Fulvio Corno, Matteo Sonza Reorda, Giovanni Squillero, Luis Entrena & Celia Lopez. (2002). New techniques for speeding-up fault-injection campaigns ». Dans *Proceedings of the conference on Design, automation and test in Europe*. p. 847. IEEE Computer Society.

Bilel Bhloul. (2020). Développement d'un modèle fautif à haut niveau d'abstraction simulant l'impact des radiations cosmiques sur les composants électroniques. *(École de technologie supérieure).*

Bingqian Xie & Xinming Huang. (2014). Real-Time Lane Departure and Front Collision Warning System on an FPGA. Dans *2014 IEEE High Performance Extreme Computing Conference (HPEC).*IEEE.

Bombieri, Nicola, Franco Fummi & Valerio Guarnieri. (2011). Accelerating RTL fault simulation through RTL-to-TLM abstraction . Dans *2011 Sixteenth IEEE European Test Symposium. p. 117-122*. IEEE.

Borecky, J., &al. (2011). Fault models usability study for on-line tested fpga. Dans *Digital System Design (DSD), 2011 14th Euromicro Conference on*. IEEE.

Chen, Mingsong, Prabhat Mishra & Dhrubajyoti Kalita. (2012).Automatic RTL test generation from SystemC TLM specifications . Dans *ACM Transactions on Embedded Computing Systems (TECS), vol. 11, no 2, p. 38.*

Di Carlo, Stefano, Paolo Prinetto, Daniele Rolfo & Pascal Trotta. (2014). A fault injection methodology and infrastructure for fast single event upsets emulation on Xilinx SRAM-based FPGAs. Dans *2014 IEEE International Symposium on Defect and Fault.* IEEE.

Entrena, Luis, Celia López-Ongil, Mario García-Valderas, Marta Portela-García & Michael Nicolaidis. (2011). Hardware fault injection . Dans *Soft Errors in Modern Electronic Systems. p. 141-166.* Springer.

Felipe Augusto da Silva, Ahmet Cagri Bagbaba, Sandro Sartoni, Riccardo Cantoro, Matteo Sonza Reorda, Said Hamdioui & Christian Sauer, (2020). Determined-Safe Faults Identification: A step towards ISO26262 hardware compliant designs. Dans *25th IEEE European Test Symposium (ETS)*. IEEE

Foucard, G. (2010). *Taux d'erreurs dues aux radiations pour des applications implémentées dans des FPGAs à base de mémoire SRAM: prédictions versus mesures*.

Farid Bounini, Denis Gingras, Vincent Lapointe & Herve Pollart. (2015). Autonomous Vehicle And Real Time Road Lanes Detection And Tracking. Dans *2015 IEEE Vehicle Power and Propulsion Conference (VPPC).* IEEE.

Ghaffari, Fakhreddine, Fouad Sahraoui, Mohamed El Amine Benkhelifa, Bertrand Granado, Marc Alexandre Kacou & Olivier Romain. (2014). Fast SRAM-FPGA fault injection platform based on dynamic partial reconfiguration. Dans *2014 26th International Conference on Microelectronics (ICM). p. 144-147*. IEEE.

Gosheblagh, Reza Omidi, & Karim Mohammadi. (2013). Dynamic partial based Single Event Upset (SEU) injection platform on FPGA. Dans *International Journal of Computer Applications, vol. 76, no 3.*

Guan Jing, Zhou Yaling & Sun Hanwen. (2018). Research on Electrical Fault Injection Experiments of Hybrid Electric Vehicle Controller. Dans *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*. IEEE

Jamal Raiyn, (2018). Using intelligent cooperative system for travel flow management in autonomous vehicle networks. Dans *2018 UKSim-AMSS 20th International Conference on Modelling & Simulation*. IEEE.

Jenn, E., Arlat, J., Rimen, M., Ohlsson, J., & Karlsson, J. (1995). Fault injection into VHDL models: the MEFISTO tool. Dans *Predictably Dependable Computing Systems* (pp. 329-346). Springer.

Juneau, Marc. (2010). Méthode d'évaluation de la sensibilité aux radiations. (École de technologie supérieure*).*

Junzhao Liu, Mohamed Mohandes & Mohamed Deriche. (2013). A Multi-Classifier Image Based Vacant Parking Detection System. Dans *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS).* IEEE

Karlsson, Johan, Ulf Gunneflo, Peter Lidén et Jan Torin. 1991. « Two fault injection techniques for test of fault handling mechanisms ». Dans *Test Conference, 1991, Proceedings., International. p. 140*. IEEE.

Khatri, A. R., Milde, M., Hayek, A., & Börcsök, J. (2014). Instrumentation technique for FPGA based fault injection tool. Dans *5th International Conference on Design and Product Development (ICDPD 2014), Istanbul, Turkey* (pp. 68-74).

Libiao Jiang, Jingxuan Li & Wandong Ai, (2019). Lane Line Detection Optimization Algorithm based on Improved Hough Transform and R-least Squares with Dual Removal. Dans *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC 2019).* IEEE.

Luc Provencher.(2010). Développement d'une infrastructure pour l'accélération sur matériel de la simulation numérique. (École de technologie supérieure).

Lv Ning, Niu Shuyan & Li Xinran. (2011) .Research on Image Processing Algorithm of Intelligent Car with Visual Navigation. Dans *2011 The 6th International Forum on Strategic Technology*. IEEE

Michel, T, Regis Leveugle, Gabriele Saucier, R Doucet & P Chapier. (1994).Taking advantage of ASICs to improve dependability with very low overheads [PLC].

NASA. (20 February 2013). Repéré à http://www.nasa.gov/mission_pages/rbsp/news/third-belt.html

Ning Xin, Guo-hong Wang & Jing Zhang. (2006). A synthetical pose estimation of SAR imagery using Hough transform and 2-D continuous wavelet transform. Dans *2006 CIE International Conference on Radar.* IEEE.

Parlons science. (2019). Repéré à https://parlonssciences.ca/ressources-pedagogiques/documents-dinformation/quest-ce-que-le-rayonnement-cosmique

Paul V. C. Hough. Hough. (1962). Method and means for recognizing complex patterns.Dans *u.s. patent 3069654.*

Preemon Rattanathammawat & Thanarat H. Chalidabhongse. (2006). A Car Plate Detector using Edge Information. Dans *2006 International Symposium on Communications and Information Technologies.* IEEE.

Qing Li & Nanning Zheng. (2003). Springrobot: A Prototype Autonomous Vehicle and Its Algorithms for Lane Detection. Dans *IEEE Transactions on Intelligent Transportation Systems.* IEEE.

Quentin LAMBERT. (2019). Amélioration du processus de testabilité des circuits intégrés asynchrones dérivés de la topologie de conception d'Octasic. (École de technologie supérieure).

Robache, R. (2013). Mise en œuvre et caractérisation d'une méthode d'injection de pannes à haut niveau d'abstraction. (École de technologie supérieure).

Rongsheng Zhang , Liyi Xiao, Jie Li , Xuebing Cao et Linzhe Li. An Adjustable and Fast Error Repair Scrubbing Method Based on Xilinx Essential Bits Technology for SRAM-Based FPGA. Dans IEEE Transactions on Reliability, vol. 69, no. 2, pp. 430-439.IEEE

R. R. Sabbella & M. Arunachalam. Functional Safety Development of Motor Control Unit for Electric Vehicles. 2019 IEEE Transportation Electrification Conference (ITEC-India).IEEE

Sheetal Israni & Swapnil Jain. (2016). Edge detection of license plate using sobel operator. Dans *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT).* IEEE.

Siegle, F., Vladimirova, T., Ilstad, J., & Emam, O. (2015). Mitigation of radiation effects in SRAM-based FPGAs for space applications. Dans *ACM Computing Surveys (CSUR), 47*(2), 37.

Sieh, V., Tschache, O., & Balbach, F. (1997). VERIFY: Evaluation of reliability using VHDL-models with embedded fault descriptions. Dans *Proceedings of IEEE 27th International Symposium on Fault Tolerant Computing* (pp. 32-36). IEEE.

Somchart Chokchaitam, Phakdee Sukpornsawan, Nutcha Pungpiboon & Saowalak Tharawut. (2019). RGB Compensation based on Background Shadow Subtraction for Low-Luminance Pill Recognition. Dans *2019 4th International Conference on Control, Robotics and Cybernetics (CRC).* IEEE

Sootkaneung, Warin, & Kewal K Saluja. (2010). Gate input reconfiguration for combatingsoft errors in combinational circuits ». Dans *2010 International Conference on Dependable Systems and Networks Workshops (DSN-W). p. 107-112.* IEEE.

Souari, A. (2016). *Stratégies facilitant les tests en pré-certification pour la robustesse à l'égard des radiations* (École de technologie supérieure).

Thibeault, C. (2019). CR analysis on systems. Dans *EPICEA Deliverable 4.2 Report*.

Thibeault, C., & Gagnon, G. (2018). On the analysis and the mitigation of power supply noise and power distribution network impedance variation for scan-based delay testing techniques. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 26*(7), 1377-1390.

Ug116.(Avril 23, 2021).Device Reliability Report. Repéré à https://www.xilinx.com/support/documentation/user_guides/ug116.pdf

Vanhauwaert, P. (2008). Analyse de sûreté par injection de fautes dans un environnement de prototypage à base de FPGA (Institut National Polytechnique de Grenoble-INPG).

Velazco, R., & al. (2007). Radiation effects on embedded systems, Springer.

Wang Bin, Ma Kai, Huang Xin, Ren Shan, Wang Fang &Shi Haotian. (2019). Fault Injection Test for MCU Based On E-Motor Emulator. Dans *2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE).* IEEE.

Wang, L.-T., Wu, C.-W., & Wen, X. (2006). VLSI test principles and architectures: design for testability. Elsevier.

Wei Kaijie, Koki Honda, & Hideharu Amano. (2018). FPGA Design for Autonomous Vehicle Driving Using Binarized Neural Networks. Dans *2018 International Conference on Field-Programmable Technology (FPT).* IEEE.

Wikipédia. (2007). Rayonnement continu de freinage. Repéré à https://fr.wikipedia.org/wiki/Rayonnement_continu_de_freinage

Yi Zhang, Xiaoyuan HAN, Han Zhang & Liming Zhao. (2017). Edge Detection Algorithm of Image Fusion Based on Improved Sobel Operator. Dans *2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC)*. IEEE.

Ying Li & Sihao Ding. (2019). Fast Lane Filtering for Autonomous Vehicle. Dans *2019 IEEE National Aerospace and Electronics Conference (NAECON)*. IEEE.

Ziade, Haissam, Rafic A Ayoubi & Raoul Velazco. (2004). A survey on fault injection techniques. Dans *Int. Arab J. Inf. Technol., vol. 1, no 2, p. 171-186.*