Ecole Publique d'Ingénieurs de Caen

Rapport

PROTECTION DES SYSTEMES EMBARQUES

Date de soutenance : 27/08/2024

stage 2A 2023/2024

Yassine HMIMOU,
Année Universitaire 2023/2024
Génie Physiques et systèmes embarqués

Tuteur entreprise : **Vincent BEROULLE**
Tuteur ENSICAEN : **Miloud FRIKEL**

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

LCIS
Laboratoire de Conception et d'Intégration des Systèmes

www.ensicaen.fr

REMERCIEMENTS

Je tiens à exprimer ma plus profonde gratitude à toutes les personnes qui ont contribué au bon déroulement de mon stage au sein de LCIS à Valence.

Tout d'abord, je remercie chaleureusement M. Vincent BEROULLE, mon tuteur au LCIS, pour son encadrement, ses conseils avisés, et son soutien tout au long de ces 17 semaines. Je souhaite également exprimer ma gratitude à M. David HELY, dont l'expertise et l'assistance m'ont été précieuses pour mener à bien mes missions.

Je souhaite également remercier l'ensemble de l'équipe du LCIS pour leur accueil, leur esprit d'équipe, et leur aide précieuse. Travailler à leurs côtés a été une expérience des plus enrichissantes, tant sur le plan professionnel que personnel.

Un grand merci au monsieur Miloud FRIKEL, mon tuteur à l'ENSICAEN, pour ses encouragements et son accompagnement tout au long de ce stage.

Enfin, je remercie mes collègues de stage et les autres étudiants de LCIS pour leur soutien, leur camaraderie, et les nombreux échanges constructifs qui ont jalonné cette expérience.

TABLE DES MATIERES

I. Introduction

I.1. Sujet

Les systèmes embarqués jouent un rôle crucial dans notre vie quotidienne, s'intégrant de manière invisible mais omniprésente dans divers aspects de notre environnement technologique. Ces systèmes sont intégrés dans des dispositifs variés allant des appareils domestiques aux véhicules, en passant par les équipements industriels et médicaux. Dans le domaine de la sécurité et de la défense, les circuits intégrés sont particulièrement essentiels. Ils sont utilisés pour piloter des drones, gérer les systèmes de communication, contrôler les armes, et assurer la surveillance et la reconnaissance. Leur fiabilité et leur performance sont critiques pour la sécurité nationale et la réussite des opérations militaires.

Cependant, l'omniprésence des circuits intégrés les rend également vulnérables aux attaques matérielles. Deux types d'attaques matérielles courantes sont Les attaques par canal auxiliaire (Side-Channel Attacks : SCA) et les attaques par injection de fautes (Fault Injection Attacks : FIA).

Les attaques par canal auxiliaire se concentrent sur les caractéristiques physiques de l'appareil pendant son fonctionnement, telles que la consommation électrique, le temps d'exécution, les émissions électromagnétiques et/ou acoustiques. En analysant ces signaux, les attaquants peuvent obtenir des informations sur les processus internes de l'appareil, leur donnant accès à des informations sensibles ou à des clés cryptographiques. Le principe fondamental des SCA est que, malgré une sécurité logicielle robuste, le comportement physique des appareils varie en fonction des calculs effectués et des processus réalisés. Par exemple, le temps nécessaire à l'exécution de certaines tâches peut fournir des indications sur les données en cours de traitement.

À l'inverse, les attaques par injection de fautes consistent à perturber le fonctionnement normal du système. Ces fautes peuvent être introduites de différentes manières, comme en modifiant les entrées du système ou en manipulant ses paramètres externes. Le résultat est un écart par rapport au comportement attendu, permettant aux attaquants d'obtenir un accès non autorisé ou de révéler des faiblesses cachées. Dans le cadre de ce travail, nous nous concentrerons exclusivement sur les attaques par injection de fautes et, plus précisément, sur les capteurs permettant de détecter ces attaques afin de renforcer la sécurité des systèmes embarqués.

I.2. Présentation du laboratoire

[1]

Le **Laboratoire de Conception et d'Intégration des Systèmes** (LCIS) est un laboratoire de recherche de l'Université Grenoble Alpes associé à Grenoble INP, situé sur le campus UGA Valence dans le Drôme. Hébergé par Grenoble INP sur le site de Grenoble INP Esisar à Valence.

Fondé en octobre 1996, le LCIS est reconnu par le ministère de l'Éducation français comme équipe d'accueil depuis janvier 2003. C'est le premier laboratoire de recherche universitaire sur le site de Valence, témoignant de son rôle pionnier et de son importance dans le paysage académique et scientifique de la région.

Le LCIS rassemble plus de 60 chercheurs spécialisés en informatique, électronique, et automatique, concentrant leurs efforts autour des systèmes embarqués et communicants. Les recherches menées au laboratoire couvrent des thématiques variées, incluant :

- ***Sûreté et sécurité des systèmes embarqués et distribués*** : Développement de techniques avancées pour protéger les systèmes contre les cybermenaces et garantir l'intégrité des données.
- ***Modélisation, analyse et supervision des systèmes complexes ouverts*** : Conception de méthodes pour analyser et superviser des systèmes dynamiques et interconnectés.
- ***Systèmes radiofréquences sans fil communicants*** : Études et innovations dans le domaine des communications sans fil.

[2]

Le LCIS collabore étroitement avec diverses entreprises et institutions académiques, tant au niveau national qu'international, pour mener à bien ses projets de recherche. En plus de ses activités scientifiques, le laboratoire joue un rôle actif dans la formation des étudiants et des jeunes chercheurs, offrant des stages, des thèses, et des opportunités de collaboration.

Durant mon stage au LCIS, j'ai eu l'opportunité de travailler au sein de l'équipe CTSYS spécialisée en protéction contre les attaques, où j'ai pu participer activement à plusieurs projets liés à la protection des systèmes embarqués. Cette expérience m'a permis de contribuer de manière significative aux avancées du laboratoire dans ce domaine.

I.3. Objectif de stage

Pour protéger les systèmes embarqués contre les différents types d'attaques matérielles, plusieurs capteurs sont mis en place de différentes manières [3]. Ces dispositifs jouent un rôle crucial dans la détection et la prévention des tentatives d'intrusion et des manipulations malveillantes visant les circuits intégrés. Les attaques matérielles, telles que les attaques par injection de fautes et les attaques par canal auxiliaire, peuvent compromettre la sécurité des systèmes en permettant aux attaquants d'accéder à des informations sensibles ou de modifier le comportement du système de manière malveillante.

Les capteurs déployés pour contrer ces attaques sont divers et sophistiqués [3]. Ils peuvent inclure des dispositifs de détection de fautes, qui surveillent les anomalies dans le fonctionnement du système, et des capteurs de sécurité, qui mesurent des paramètres tels que la consommation d'énergie ou les émissions électromagnétiques pour détecter des comportements suspects. Cependant, malgré ces précautions, certaines attaques, comme les Power-Off Attacks, exploitent des périodes où les capteurs sont hors tension pour éviter d'être détectées.

Le but de ce stage est de :

- ***Analyser les mécanismes de fonctionnement de ces capteurs :*** Comprendre comment les capteurs actuels détectent et réagissent aux attaques, notamment les Fault Injection Attacks.
- ***Proposer des améliorations pour la protection des circuits intégrés :*** Sur la base des analyses et des études effectuées, suggérer des modifications ou de nouvelles approches pour renforcer la robustesse des capteurs. Ces améliorations doivent viser à accroître la sécurité des systèmes embarqués contre les attaques matérielles.

En réalisant ces objectifs, ce stage contribuera à développer des solutions plus efficaces et robustes pour la protection des systèmes embarqués, assurant ainsi la sécurité et l'intégrité des informations qu'ils traitent.

II. Etat de l'art

II.1. Mise en situation

II.1.1. Les attaques par injection de fautes (FIA)

Plusieurs méthodes ont été proposées pour protéger les dispositifs contre les attaques par injection de fautes (FIA). Ces techniques peuvent être basées sur la redondance à différents niveaux ou sur l'utilisation de capteurs. L'avantage de la redondance est qu'elle permet de détecter les fautes indépendamment de la technique d'injection utilisée, mais son principal inconvénient est qu'elle ne peut pas capturer toutes les fautes possibles. La deuxième technique consiste à utiliser des capteurs de détection de fautes. Ces capteurs peuvent être divisés en deux catégories : les capteurs numériques et les capteurs analogiques.

II.1.2. Les capteurs numériques et les capteurs analogiques

Les capteurs analogiques, comme leur nom l'indique, utilisent des sources analogiques pour détecter les attaques par injection de faute (FIA). Mais, ces capteurs analogiques sont plus difficiles à calibrer que les capteurs numériques et nécessitent une consommation d'énergie plus élevée, c'est pourquoi les capteurs numériques sont largement utilisés aujourd'hui.

[3]

Un des designs les plus populaires, appelé détecteur basé sur le retard (Delay-based detector), a été suggéré dans plusieurs études. Ce capteur est basé sur les contraintes de synchronisation des systèmes synchrones. Les détecteurs basés sur le retard peuvent détecter diverses attaques par injection de fautes, telles que les glitchs d'horloge, la sous-alimentation ou la surchauffe.

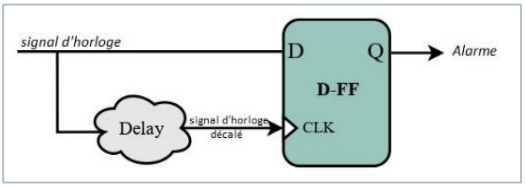


Figure 1 : Schéma du détecteur basé sur le retard (Delay-based detector)

Pour garantir que les processeurs fonctionnent correctement, la période d'horloge doit être supérieure à la somme du délai de propagation et du temps de mise en place. Car sinon, le processeur n'aura pas assez de temps pour effectuer ses opérations. Ce qui est ullistré dans l'inéquation suivante :

$$T_{clock} \geq T_{délai\ de\ propagation} + T_{mise\ en\ sistuation}$$

Avec T_{clock} : C'est le temps total alloué pour une opération complète de traitement et c'est un cycle d'horloge.

$T_{délai\ de\ propagation}$: C'est le temps nécessaire pour qu'un signal se propage à travers un circuit.

$T_{mise\ en\ situation}$: C'est le temps nécessaire pour que les données soient stabilisées et prêtes à être traitées avant le prochain cycle d'horloge.

[3]

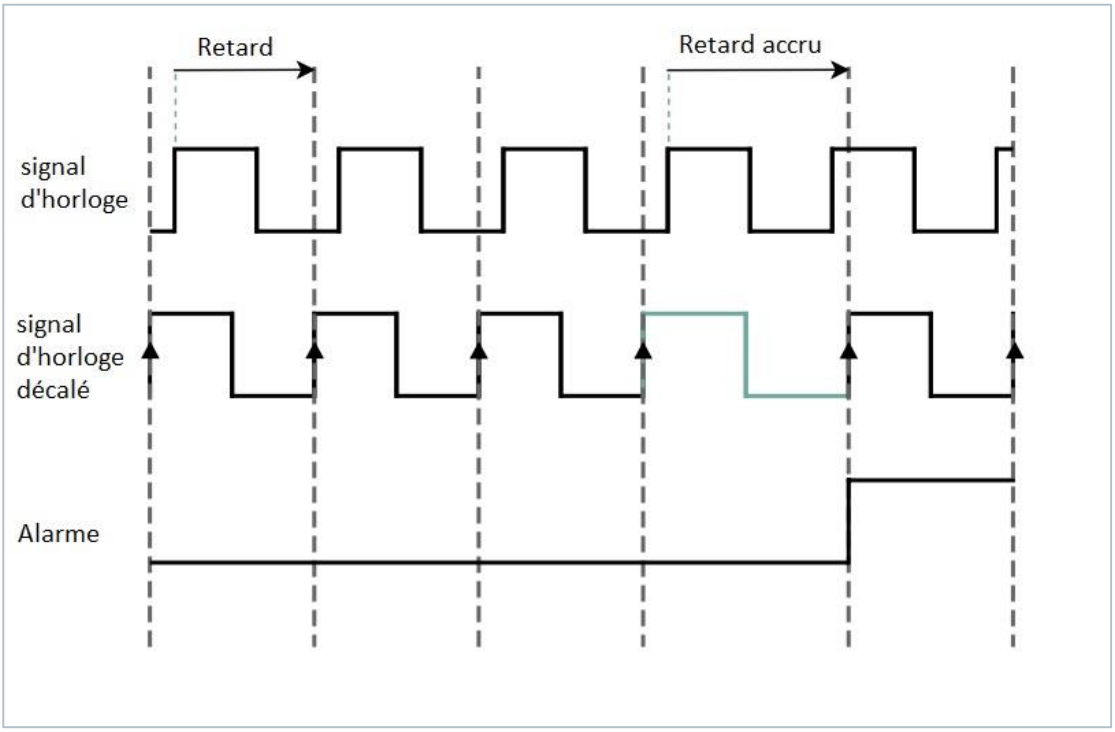
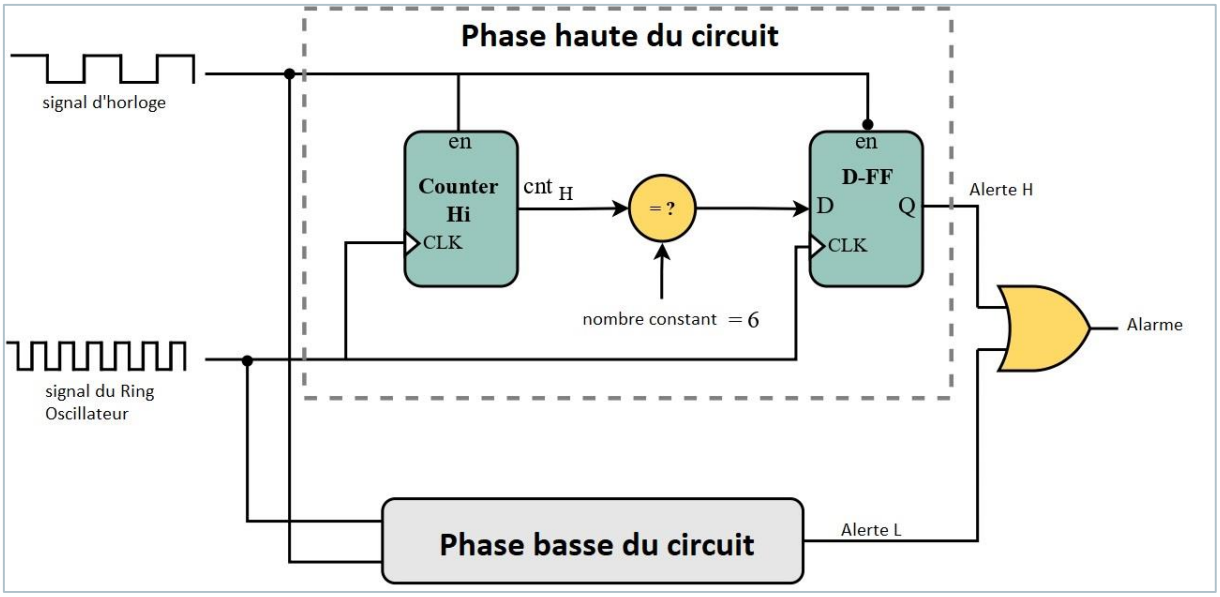


Figure 2 : Fonctionnement du détecteur à retard

La figure ci-dessus montre comment le détecteur compare le signal d'horloge décalé avec le premier signal d'horloge et déclenche l'alarme une fois une différence est detectée.

Bien que les capteurs basés sur le retard soient simples et efficaces contre certaines attaques par injection de fautes (FIA), ils sont impratiques lorsqu'il s'agit de traiter des attaques ayant un effet local, comme les FIA par laser ou électromagnétiques. En conséquence, d'autres designs ont été introduits pour améliorer les taux de détection contre les FIA. Une autre catégorie de capteurs proposés est basée sur les oscillateurs en anneau (Ring Oscillator :RO). Les oscillateurs en anneau peuvent être implémentés en utilisant une chaîne fermée de portes NOT en nombre impair. Dans cette structure, le RO alterne entre zéro et un, il peut donc être un générateur de fréquence, dont la fréquence de sortie dépend du nombre de portes NOT et des délais de propagation.

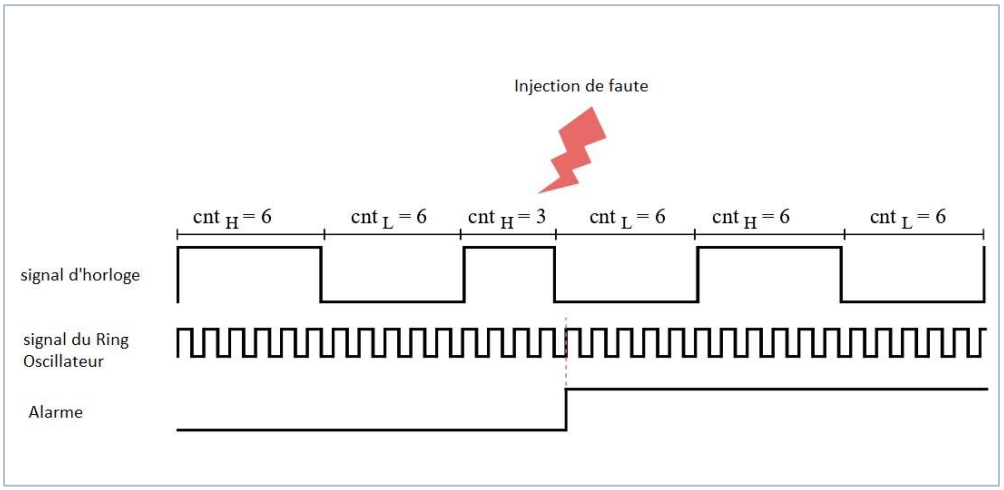
Les oscillateurs en anneau implémentés peuvent être utilisés dans la conception de capteurs. Par exemple, un détecteur consiste en deux circuits de phase haute et basse, qui sont utilisés respectivement pour les niveaux un et zéro du signal d'horloge. Comme illustré sur la figure ci-dessous :



[4]

Figure 3 : Schéma du détetcteur basé sur le Ring Oscillateur [4]

Chacun de ces circuits compte le nombre d'oscillations des oscillateurs en anneau (RO) à chaque niveau d'horloge, puis les compare à une valeur constante. En mode normal, le nombre d'oscillations des RO reste toujours le même, mais en cas d'attaque, ce nombre change et ne correspond plus à la valeur constante. Par conséquent, l'alarme est activée. Ce qui est illustré sur le chronogramme qui suit pour une valeur constante de 6 :



[4]

Figure 4 : Fonctionnement du détecteur basé sur le Ring Oscillateur [4]

Comme ce détecteur utilise deux circuits séparés pour le niveau zéro et un de l'horloge, il peut accélérer la détection des fautes, ce qui lui confère une précision supérieure à celle des détecteurs basés sur le retard. Ainsi, il peut être utilisé pour détecter les fautes locales telles que les attaques par laser et électromagnétiques.

II.2. Problématique

Les détecteurs eux-mêmes sont des cibles potentielles pour les attaques, ce qui soulève la nécessité de les protéger en priorité. Les attaquants peuvent viser ces dispositifs en les manipulant physiquement, en injectant des fautes pour altérer leurs mesures, ou en compromettant les logiciels qui les contrôlent. Ces attaques peuvent perturber les capteurs de plusieurs façons : les manipulations physiques peuvent désactiver ou endommager les capteurs, les injections de fautes peuvent fausser les données collectées, et les attaques logicielles peuvent modifier le fonctionnement des capteurs ou désactiver leurs capacités de détection.

II.3. Contribution

Pour assurer le bon fonctionnement des capteurs et garantir leur efficacité contre les attaques, il est nécessaire de développer des mécanismes de validation robustes. La redondance, qui consiste à utiliser plusieurs capteurs pour surveiller le même paramètre, est une première mesure de sécurité. Elle permet de détecter les anomalies en comparant les données des différents capteurs. Cependant, cette approche peut se révéler insuffisante si les attaquants parviennent à compromettre simultanément tous les capteurs redondants.

Par conséquent, une approche plus efficace implique une validation continue de la fréquence des détecteurs sur une période prolongée. En surveillant constamment le comportement des capteurs et en comparant sa fréquence d'oscillation à des valeurs de référence, il est possible de détecter toute anomalie indicative d'une attaque. Par exemple, dans des systèmes utilisant des oscillateurs en anneau (RO), le nombre d'oscillations à chaque niveau d'horloge est compté et comparé à une valeur constante. En cas d'attaque, le nombre d'oscillations change et ne correspond plus à la valeur attendue, déclenchant ainsi une alarme.

Cette méthode permet non seulement de détecter les attaques locales, telles que les attaques par laser et électromagnétiques, mais aussi de bien vérifier le bon fonctionnement des capteurs avec une plus grande précision. De cette manière, la robustesse des systèmes contre les attaques matérielles est significativement renforcée, assurant une meilleure protection des informations sensibles et du fonctionnement global des systèmes embarqués.

III. Etude théorique du dispositif de protection proposé

III.1. Objectif

L'objectif principal de cette étude théorique est de développer un mécanisme de test robuste qui valide la stabilité de la fréquence des capteurs de sécurité sur une période prolongée. Cette stabilité est cruciale pour garantir que les capteurs fonctionnent correctement et détectent les anomalies de manière précise.

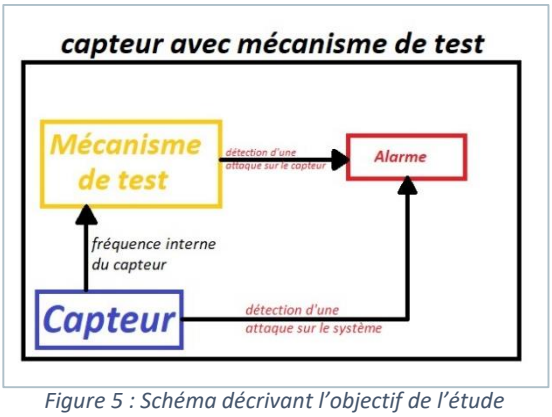


Figure 5 : Schéma décrivant l'objectif de l'étude

III.2. Exigences

La vérification de la fréquence des capteurs peut se réaliser de manière efficace en comptant le nombre de cycles du capteur présents dans un intervalle de temps défini. Cette approche permet une précision élevée grâce à une large plage de compte, garantissant ainsi que toute variation anormale de la fréquence soit détectée de manière fiable.

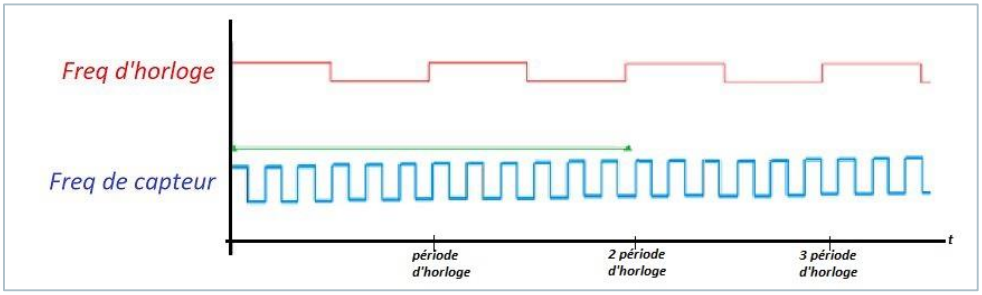


Figure 6 : Chronogramme d'un exemple de fonctionnement du mécanisme

Par exemple, sur la figure ci-dessus on compte le nombre de période du capteur dans une plage de deux cycles d'horloge.

III.3. Schéma fonctionnel

Pour accomplir un bon compte de cycles du capteur, il est nécessaire de mettre en place un mécanisme sous forme d'un système de comptage précis qui surveille en continu les cycles générés par le capteur. Ce qui est montré sur la figure 7 ci-après.

Ce système est composé de plusieurs fonctions essentielles :

- **Première fonction : Compter les cycles du capteur**

La première étape consiste à compter le nombre de cycles du capteur présents dans un intervalle de temps précis. Pour cela, un composant est nécessaire pour compter les cycles du capteur durant le temps spécifique de mesure.

- **Deuxième fonction : Spécifier le temps de mesure**

Il est nécessaire d'avoir un deuxième composant qui spécifie au premier composant le temps de mesure.

- **Troisième fonction : Récupération de la valeur de compte**

Comme le premier composant effectue des cycles de comptage à chaque fois, il sera essentiel de stocker la valeur atteinte à chaque fin de temps de mesure pour pouvoir la comparer ultérieurement avec une valeur de référence.

- **Quatrième fonction : Comparaison**

Après avoir capturé la valeur des cycles présents sur une période définie, il est nécessaire de valider la fréquence du capteur afin de détecter toute anomalie.

Pour mieux illustrer toutes les étapes mentionnées précédemment, nous pouvons les regrouper dans un schéma fonctionnel, où l'on observe bien la présence des quatre composants précédemment mentionnés. Le premier composant compte le nombre de cycles à partir de la fréquence du capteur, notée F_{cap}, en utilisant le signal de réinitialisation généré par le deuxième composant. Le deuxième composant spécifie le temps de mesure en se basant sur la fréquence de l'horloge principale, notée F_{clk}, et le nombre de cycles choisi, N. Le troisième composant stocke la valeur provenant du compteur, notée Q, et la transfère au dernier composant, qui compare Q à une référence X. En se basant sur le résultat de cette comparaison, il déclenche ou non l'alarme.

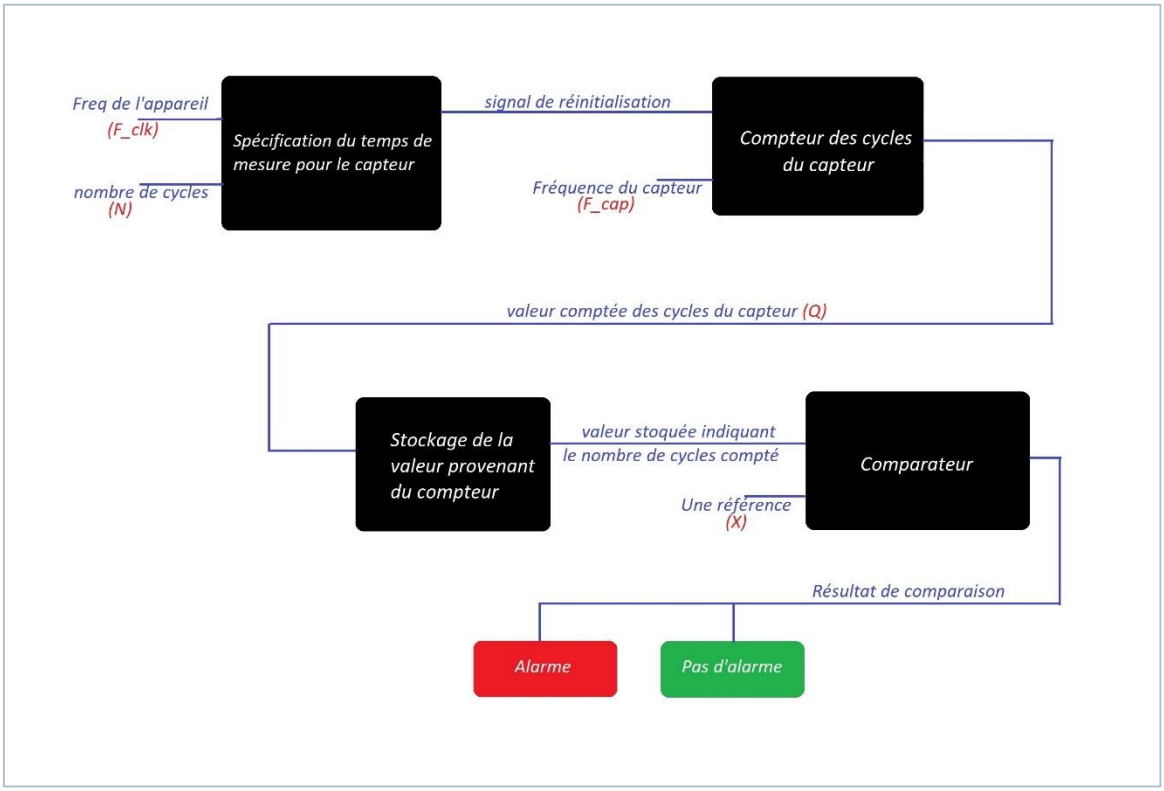


Figure 7 : Schéma fonctionnel du mécanisme

III.4. Schéma structurel

III.4.1. Spécification du temps de mesure

Le premier composant de notre système de comptage est chargé de spécifier le temps de mesure. Ce composant prend en entrée la fréquence de l'horloge principale F_{clk} et un nombre N choisi de cycles d'horloge de l'appareil. À chaque N cycles, il envoie un signal de réinitialisation au composant suivant. Cela définit ainsi un intervalle de temps précis pour le comptage des cycles du capteur. Ce mécanisme permet de segmenter la période de comptage en intervalles réguliers, assurant une surveillance continue et prolongée de la fréquence du capteur.

III.4.2. Comptage des cycles du capteur

Le deuxième composant de notre système de comptage prend en entrée le signal de réinitialisation provenant du premier composant ainsi que la fréquence du capteur F_{cap}. Ce composant est responsable de compter le nombre de cycles générés par le capteur dans chaque période de mesure définie par le signal de réinitialisation. Une fois le comptage effectué pour chaque intervalle, il transmet la valeur comptée au composant suivant. Cela permet de capturer de manière précise le comportement du capteur sur chaque période de mesure, facilitant ainsi l'analyse et la détection d'éventuelles anomalies.

III.4.3. Capture de la valeur comptée

Le troisième composant de notre système de comptage est chargé de capturer la valeur comptée. Ce composant reçoit en entrée la valeur des cycles comptés provenant du deuxième composant. À la fin de chaque période de mesure, il stocke cette valeur comptée (notée Q). Cette valeur est ensuite conservée jusqu'à ce qu'elle soit transmise au composant suivant pour la comparaison. En stockant les valeurs comptées, ce composant permet d'assurer une trace continue et précise des cycles du capteur, ce qui est essentiel pour détecter toute déviation par rapport à un comportement normal attendu.

III.4.4. Comparaison à une référence

Le quatrième composant de notre système de comptage est responsable de la comparaison des valeurs comptées à une référence. Ce composant prend en entrée la valeur des cycles comptés (notée Q) provenant du troisième composant et la compare à une valeur de référence prédéfinie (notée X). Si la valeur Q diffère de manière significative de X, cela indique une anomalie potentielle, et le composant déclenche une alarme. Cette étape est cruciale pour valider la fréquence du capteur et détecter toute irrégularité, garantissant ainsi la sécurité et le bon fonctionnement du système.

III.5. Le fonctionnement global

Après avoir compris les détails du mécanisme, il est nécessaire de connaître les interactions avec les autres composants du système. Si nous disposons d'une fréquence de capteur et d'une fréquence d'horloge connues comment choisir les valeurs appropriées pour le nombre de cycles N et la référence X ?

III.5.1. Détermination du nombre de cycles N

Dans un état de fonctionnement idéal, comme présenté dans la figure ci-dessouspour N=2 , où les deux horloges sont bien synchronisées, on peut établir la relation suivante :$F_{cap} = \frac{Q * F_{clk}}{N}$

Avec Q : le nombre compté des cycles du capteur présents dans N cycles d'horloge

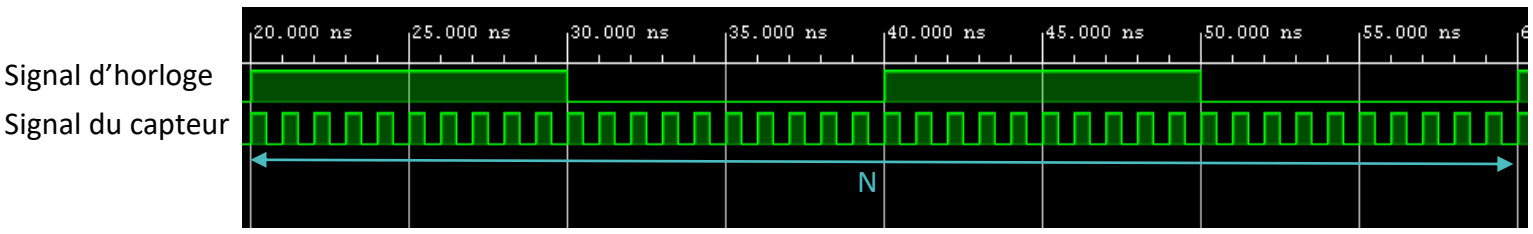


Figure 8 : Chronogramme du fonctionnement idéal pour N = 2

En revanche, en cas de non-synchronisation ou de décalage entre les deux fréquences, comme illustré dans la figure suivante, la relation change en fonction du décalage :

Donc $F_{cap} = \frac{Q * F_{clk}}{N}$ ou $F_{cap} = \frac{(Q+1) * F_{clk}}{N}$

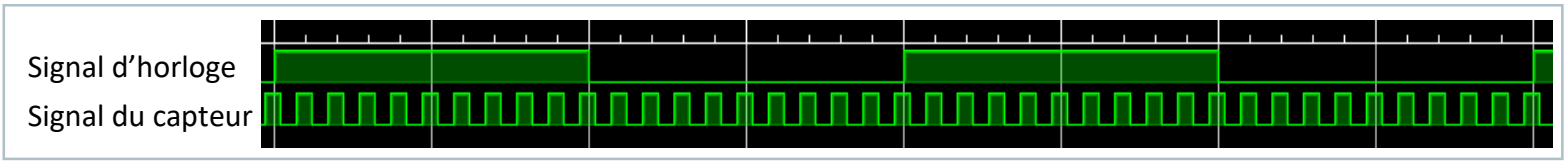


Figure 9 : Chronogramme du fonctionnement avec décalage

Ainsi, on obtient une erreur qui est la différence entre la fréquence obtenue en cas de décalage et la fréquence obtenue en fonctionnement normal :

$$\Delta f = f_{obtenue\ sur\ décalage} - f_{obtenue\ sur\ normal}$$

Finalement, cette marge d'erreur s'exprime par :

$$\Delta f = \frac{(Q+1)*F_{clk}}{N} - \frac{Q*F_{clk}}{N} = \frac{F_{clk}}{N}$$

Cela signifie que la marge d'erreur est égale à la fréquence de l'horloge divisée par le nombre de cycles choisi. Par conséquent, le nombre de cycles N peut être déterminé en fonction de la fréquence de l'horloge et de la marge d'erreur acceptable :

$$\mathrm{N} > \frac{F_{clk}}{\Delta f}$$

III.5.2. Détermination de la référence X

Dans un fonctionnement idéal, la relation entre la fréquence du capteur et le nombre de cycles comptés est donnée par :

$$F_{cap} = \frac{Q * F_{clk}}{N}$$

Lors d'un changement de fréquence de k%, la nouvelle fréquence du capteur devient $f_{cap} \pm k\%$. La relation correspondante pour cette nouvelle fréquence est alors :

$$F_{cap} \pm k\% = \frac{Q_{new} * F_{clk}}{N}$$

Où Q_{new} est la valeur correspondant à la nouvelle fréquence.

Finalement, la valeur limite $X_{k\%}$ pour un changement de k% de la fréquence est donnée par :

$$X_{-k\%} = \left\lceil \frac{N*(F_{cap}-k\%)}{F_{clk}} \right\rceil - 1 \qquad \text{Et} \qquad X_{+k\%} = \left\lceil \frac{N*(F_{cap}+k\%)}{F_{clk}} \right\rceil + 1$$

Cela signifie que la limite pour un changement de k% de la fréquence est la valeur $X_{k\%}$. L'ajout de ± 1 prend en compte un éventuel décalage entre les deux horloges. Ainsi, le résultat obtenu montre que si la valeur Q dépasse les limites de l'intervalle $[X_{-k\%}, X_{+k\%}]$, un changement de fréquence de k% sera détecté, indiquant une variation significative par rapport à la fréquence d'origine du capteur.

Supposons que la fréquence du capteur F_{cap} soit de 100 MHz, la fréquence d'horloge F_{clk} de 50 MHz, et que le nombre de cycles N soit de 10. Si l'on souhaite détecter une variation de fréquence de 5%, on calcule $X_{+5\%}$ et $X_{-5\%}$ pour établir les bornes de l'intervalle. En appliquant la formule, cela donnera des valeurs spécifiques pour $X_{-5\%} = 18$ et $X_{+5\%} = 22$, et toute valeur Q en dehors de cet intervalle indiquerait une variation de la fréquence de plus de 5%, déclenchant ainsi une alarme.

III.6. Conclusion

En conclusion, nous avons mis en place un système de comptage précis composé de plusieurs étapes essentielles : compter les cycles du capteur, spécifier le temps de mesure, récupérer et stocker les valeurs de compte, et comparer ces valeurs à un interval de références pour déclencher des alertes en cas d'anomalie.

Les étapes de détermination de N et X sont basées sur des calculs précis prenant en compte la marge d'erreur, garantissant ainsi une détection efficace des anomalies. Ce processus assure que les capteurs fonctionnent correctement et protègent efficacement les circuits intégrés contre les attaques matérielles.

IV. Implémentation et vérification

IV.1. Mise en œuvre du système

L'implémentation du système s'est basée sur la réalisation des quatre composants clés décrits précédemment :

- Spécifier le temps de mesure : **[Annexe 1]**

Le premier composant, qui spécifie le temps de mesure, a été implémenté sous forme d'un compteur. Ce compteur prend en entrée la fréquence de l'horloge principale F_{clk} et le nombre de cycles d'horloge N choisi. À chaque N cycles, ce compteur génère un signal de réinitialisation, déterminant ainsi l'intervalle de temps pour le comptage des cycles du capteur.

- Compter les cycles du capteur : **[Annexe 2]**

Le deuxième composant, chargé de compter les cycles du capteur, est également un compteur. Il reçoit en entrée le signal de réinitialisation provenant du premier composant ainsi que la fréquence du capteur F_{cap}. Ce compteur suit le nombre de cycles générés par le capteur durant chaque période de mesure définie par le premier compteur, et transmet cette valeur au composant suivant.

- Stocker la valeur de compte : **[Annexe 3]**

Le troisième composant, qui capture la valeur comptée, est réalisé à l'aide d'un registre. Ce registre stocke la valeur du compteur à chaque fin de période de mesure, garantissant ainsi que les données capturées peuvent être comparées ultérieurement sans être affectées par les cycles en cours.

- Comparer la valeur de compte à une référence : **[Annexe 4]**

Enfin, le dernier composant, un comparateur, reçoit la valeur stockée par le registre et la compare avec un intervalle de valeurs définie $[X_{-k\%}, X_{+k\%}]$. En fonction du résultat de cette comparaison, le comparateur déclenche ou non une alarme, signalant une variation potentielle de la fréquence du capteur en dehors des limites acceptables.

L'ensemble de ces composants a été intégré pour former un système cohérent capable de surveiller en continu la stabilité de la fréquence du capteur et de détecter toute anomalie en temps réel.

IV.2. Configuration de test

Les étapes de mise en œuvre précédemment décrites ont été réalisées en VHDL. Les différents composants, à savoir les deux compteurs, le registre et le comparateur, ont été codés en VHDL. Le code source complet de chaque composant est présenté en annexe.

Après l'assemblage des composants en un système fonctionnel, un test a été mené pour vérifier le bon fonctionnement du système dans des conditions normales, c'est-à-dire sans anomalies ni changements de fréquence. Pour ce test, une fréquence d'horloge de 50 MHz, couramment utilisée comme standard sur les cartes FPGA, a été choisie. Différentes fréquences de capteur ont été testées, notamment 100 MHz, 200 MHz et 1 GHz. Ce test a permis de valider que les interactions entre les composants se déroulent correctement et que le système fonctionne comme prévu, sans déclenchement injustifié de l'alarme.

IV.3. Résultats expérimentaux

IV.3.1. Validation de N

Dans cette section, nous validons la relation théorique établie pour le paramètre N en comparant les résultats expérimentaux avec les prévisions théoriques. Pour cela, nous avons testé plusieurs fréquences de capteur en utilisant une fréquence d'horloge fixe de 50 MHz, standard sur les cartes FPGA. Nous avons évalué la relation suivante, déjà mentionnée dans la partie théorique:

$$N > \frac{F_{clk}}{\Delta f}$$

Cette relation indique que la marge d'erreur Δf est inversement proportionnelle à N. Ainsi, en augmentant N, la marge d'erreur diminue, ce qui améliore la précision du système.

Pour valider cette relation, nous avons effectué des tests sur plusieurs fréquences de capteur (100 MHz, 200 MHz, 1 GHz, etc.) et pour diverses valeurs de N (2,4,7,10…). Les résultats obtenus montrent que les valeurs expérimentales sont en accord avec les valeurs théoriques, confirmant la validité de la relation.

Le tableau suivant présente un échantillon des valeurs testées, où l'on observe qu'en fixant une marge d'erreur maximale de 10 Mhz, on obtient une valeur $N_{min} = 5$ telle que, tant que $N > N_{min}$, l'erreur ne dépasse pas 10 Mhz :

F_{clk} (Mhz)	F_{cap} (Mhz)	N	Fréquence calculée par le mécanisme (Mhz)	Δf (Mhz)	N_{min} calculée théoriquement
50	100	2	125	25	5
		4	112,5	12,5	
		7	107,143	7,143	
		10	105	5	
50	200	2	225	25	5
		4	212,5	12,5	
		7	207,143	7,143	
		10	205	5	
50	250	2	275	25	5
		4	262,5	12,5	
		7	257,143	7,143	
		10	255	5	
50	1000	2	1025	25	5
		4	1012,5	12,5	
		7	1007,143	7,143	
		10	1005	5	

Table 1 : Résultats obtenus pour une marge d'érreur de 10 Mhz

Les résultats confirment que la théorie est bien validée par l'expérimentation, et que la précision du système de détection peut être optimisée en ajustant le paramètre N.

IV.3.2. Validation de X

Dans cette section, nous validons la relation théorique pour le paramètre $X_{k\%}$ en comparant les résultats obtenus expérimentalement avec les prévisions théoriques. Le paramètre $X_{k\%}$ détermine l'intervalle de valeurs autour de la fréquence de référence du capteur, permettant de détecter des variations de fréquence égales ou supérieures à un certain pourcentage k%. La relation théorique utilisée est la suivante :

$$X_{k\%} = \lceil \frac{N*(F_{cap} \pm k\%)}{F_{clk}} \rceil \pm 1$$

Pour valider cette relation, nous avons réalisé des tests en prenant différentes fréquences de capteur (100 MHz, 200 MHz, 1 GHz, etc.) avec une fréquence d'horloge fixe de 50 MHz. Les résultats expérimentaux ont été comparés aux valeurs théoriques de $X_{k\%}$, en évaluant la capacité du système à détecter un changement de fréquence k% lorsque la valeur comptée Q dépasse les limites de l'intervalle $X_{k\%}$.

Les résultats expérimentaux confirment que la valeur Q reste dans les limites de l'intervalle $[X_{-k\%}, X_{+k\%}]$ en l'absence de perturbation, et dépasse ces limites lorsqu'un changement de fréquence est introduit. Ce comportement valide la relation théorique, en montrant que le système est capable de détecter des variations de fréquence avec une précision conforme aux attentes.

Le tableau suivant illustre quelques résultats, montrant la correspondance entre les valeurs théoriques et expérimentales pour différentes configurations. En fixant une variation de 10% de la fréquence du capteur, on obtient un intervalle $[X_{-k\%}, X_{+k\%}]$ dont lequel toutes les valeurs incluses conviennent à cette variation. En revanche, toute valeure qui dépasse les extrémités de cet intervalle correspond à une variation supérieure à 10%. Cela est bien visible pour les valeurs de « Q avec variation » mésurées dans le tableau suivant, qui sont comprises dans l'intervalle $[X_{-k\%}, X_{+k\%}]$:

F_{clk} (Mhz)	F_{cap} (Mhz)	types de mesure	N	Q sans variation	Q avec variation	$X_{k\%}$ mésurée théoriquement	
50	250	Avec variation de - 10 %	2	10	10	9	$X_{-k\%}$
			4	20	19	18	
			7	35	33	31,5	
			10	50	46	45	
50	250	Avec variation de + 10 %	2	10	12	12	$X_{+k\%}$
			4	20	23	23	
			7	35	37	39,5	
			10	50	56	56	
50	1000	Avec variation de - 10 %	2	40	37	36	$X_{-k\%}$
			4	80	73,5	72	
			7	140	128	126	
			10	200	182	180	
50	1000	Avec variation de + 10 %	2	40	43	44	$X_{+k\%}$
			4	80	87	88	
			7	140	152	154	
			10	200	220	220	

Table 2: Résultats obtenus pour un changement de fréquence

Les résultats confirment que le système est capable de détecter efficacement des variations de fréquence validant ainsi l'approche théorique.

IV.3.3. Validation du fonctionnement global

Pour évaluer le bon fonctionnement global du système, nous avons effectué une série de tests visant à valider son comportement en conditions normales et en présence de perturbations. L'objectif est de vérifier que le système réagit correctement aux variations de la fréquence du capteur, conformément aux spécifications définies.

Dans un premier temps, nous avons testé le système en l'absence de toute anomalie ou changement de fréquence, en utilisant une fréquence d'horloge de 50 MHz et différentes fréquences de capteur (100 MHz, 200 MHz, 1000 MHz). Le système a correctement compté les cycles du capteur, et la valeur Q obtenue est restée dans les limites de l'intervalle $[X_{-k\%}, X_{+k\%}]$, confirmant le bon fonctionnement du système en mode normal.

Ensuite, nous avons introduit des variations de fréquence pour simuler des perturbations, avec pour objectif de vérifier si le système pouvait détecter ces changements en comparant la valeur Q aux limites calculées de l'intervalle $[X_{-k\%}, X_{+k\%}]$. Les résultats montrent que, pour des variations de fréquence exactement égales au seuil de k%, la valeur Q atteint les limites de l'intervalle, déclenchant ainsi l'alarme comme attendu en cas de dépassement. Cependant, pour des variations supérieures au seuil, la capacité du système à détecter ces anomalies dépend de la marge d'erreur inhérente au mécanisme de détection. Cette marge d'erreur doit être prise en compte, car elle peut influencer la précision du déclenchement de l'alarme en cas de variations de fréquence plus importantes. **[Annexe 5]**

Les résultats expérimentaux démontrent que le système fonctionne correctement dans toutes les conditions testées mais il faut tenir compte de la marge d'erreur. Il est capable de surveiller en continu les cycles du capteur, de détecter des anomalies de fréquence, et de réagir en temps réel en cas de détection d'une perturbation. Cette validation confirme la robustesse du système et son aptitude à protéger les circuits intégrés contre les attaques basées sur la manipulation des fréquences.

V. Conclusion et perspectives

Au terme de ce projet, nous avons développé et validé un système capable de surveiller en continu la fréquence des capteurs embarqués, en détectant et réagissant efficacement aux variations qui pourraient indiquer une tentative d'attaque. En s'appuyant sur un mécanisme de comptage précis et une comparaison rigoureuse avec une référence préétablie, le système a démontré sa robustesse à travers divers tests expérimentaux, validant ainsi les relations théoriques définies au préalable.

Les résultats obtenus montrent que notre approche permet de détecter avec précision les variations de fréquence, et d'alerter en cas d'anomalie, ce qui est essentiel pour garantir la sécurité des systèmes embarqués. En outre, la configuration testée, avec une fréquence d'horloge standard de 50 MHz, s'est révélée adaptée à diverses fréquences de capteurs, soulignant la flexibilité du système.

V.1. Perspectives

Plusieurs pistes d'amélioration peuvent être envisagées pour renforcer encore davantage le système développé. Tout d'abord, il serait pertinent de considérer les cas où l'on souhaite détecter des variations de fréquence plus petites ou égales à notre marge d'erreur, afin d'évaluer la capacité du système à réagir face à des anomalies plus prononcées. De plus, bien que les simulations réalisées sur Vivado aient montré des résultats encourageants, une validation plus concrète sur une carte FPGA serait nécessaire pour garantir le bon fonctionnement du système dans des conditions réelles.

Enfin, pour aller plus loin dans l'évaluation de l'efficacité du dispositif, il serait crucial de tester le système avec un capteur réel soumis à une attaque, afin de vérifier sa capacité à détecter et à réagir face à de véritables tentatives de manipulation. Cette approche permettrait de déceler d'éventuels aspects non pris en compte lors des simulations, et ainsi d'apporter les ajustements nécessaires pour renforcer la fiabilité et la sécurité du système dans un contexte pratique.

ANNEXES

TABLE DES FIGURES

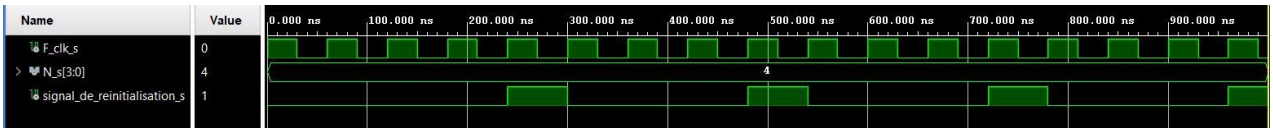
TABLE DES TABLEAUX

ANNEXES

Annexe 1.1 : Code source de la fonction « Specification de temps de mesure »

```
-- Importation des bibliothèques nécessaires
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;
-- Déclaration de l'entité et des ports entrées et sorties
entity specification_de_temps_de_mesure is Port (
        F_clk : in STD_LOGIC;                                   -- Horloge principale
        N : in STD_LOGIC_VECTOR (3 downto 0);                   -- Nombre de cycles d'horloge
        signal_de_reinitialisation : out STD_LOGIC := '0');     -- Signal de réinitialisation
end specification_de_temps_de_mesure;
-- Définition du fonctionnement
architecture Behavioral of specification_de_temps_de_mesure is
    signal cnt : STD_LOGIC_VECTOR (3 downto 0) := "0001";       -- Signal pour le compteur
begin
    process(F_clk)
    begin
        if rising_edge(F_clk) then
            if cnt = N then
                signal_de_reinitialisation <= '1';              -- Générer le signal de réinitialisation
                cnt <= "0001";                                  -- Réinitialiser le compteur
            else
                cnt <= cnt + 1;                                 -- Incrémenter le compteur
                signal_de_reinitialisation <= '0';              -- Maintenir le signal de réinitialisation
            end if;
        end if;
    end process;
end Behavioral;
```

Annexe 2.2 : Simulation de la fonction « Specification de temps de mesure »

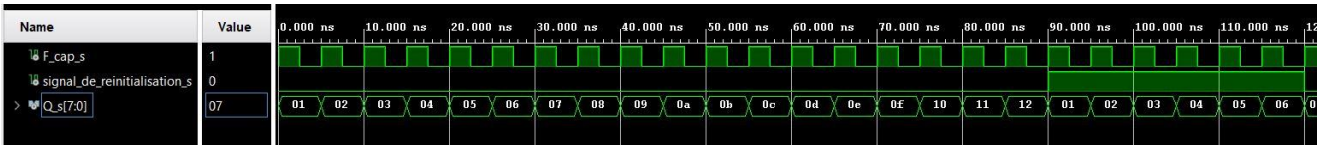


Dans cette simulation, une fréquence d'horloge de 16.67 MHz a été utilisée, correspondant à une période d'horloge de 60 ns (30 ns pour chaque état haut et bas de l'horloge). La valeur d'entrée pour le nombre de cycles N a été définie à 4, ce qui signifie que le système est configuré pour générer un signal de réinitialisation après 4 cycles d'horloge. Le signal de réinitialisation signal_de_reinitialisation_s est observé pour vérifier qu'il se déclenche correctement après que le compteur ait atteint la valeur spécifiée par N.

Annexe 2.1 : Code source de la fonction « Compteur des cycles du capteur »

```vhdl
-- Importation des bibliothèques nécessaires
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;
-- Déclaration de l'entité et des ports entrées et sorties
entity compteur_des_cycles_du_capteur is
    Port ( F_cap : in STD_LOGIC;                                -- Fréquence du capteur
           signal_de_reinitialisation : in STD_LOGIC;           -- Signal de réinitialisation
           Q : out STD_LOGIC_VECTOR (7 downto 0));              -- Valeur comptée des cycles du capteur
end compteur_des_cycles_du_capteur;
-- Définition du fonctionnement
architecture Behavioral of compteur_des_cycles_du_capteur is
    signal cnt : STD_LOGIC_VECTOR (7 downto 0) := "00000001";   -- Signal pour le compteur
    signal signal_de_reinitialisation_dernier : STD_LOGIC := '0';
begin
    process(F_cap, signal_de_reinitialisation)
    begin
        if signal_de_reinitialisation = '1' and signal_de_reinitialisation_dernier ='0' then
            cnt <= "00000001";                                  -- Réinitialiser le compteur
        elsif rising_edge(F_cap) then
            cnt <= cnt + 1;                                     -- Incrémenter le compteur
        end if;
        signal_de_reinitialisation_dernier <= signal_de_reinitialisation ;
    end process;
    Q <= cnt;
end Behavioral;
```

Annexe 2.2 : Simulation de la fonction « Compteur des cycles du capteur »



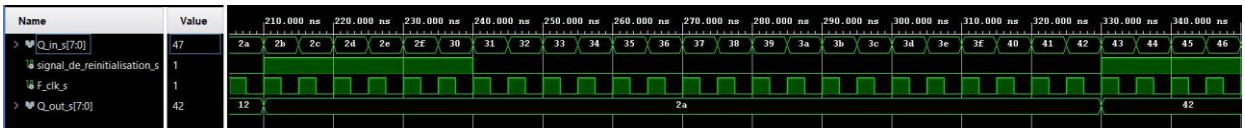
Dans cette simulation, la fréquence du capteur est configurée pour générer une horloge à 200 MHz, ce qui correspond à une période d'horloge de 5 ns (2,5 ns pour chaque état haut et bas de l'horloge). Le signal de réinitialisation signal_de_reinitialisation_s est d'abord maintenu bas, puis activé à des intervalles réguliers pour tester le comportement du compteur. (les nombres sur la figure sont en hexadécimal)

Le compteur Q_s est observé pour s'assurer qu'il compte correctement les cycles de l'horloge du capteur entre chaque signal de réinitialisation. Cette simulation permet de valider que le compteur compte les cycles de manière précise et que le signal de réinitialisation remet bien le compteur à zéro avant de commencer un nouveau comptage.

Annexe 3.1 : Code source de la fonction « Stockage de la valeur du compteur »

```
-- Importation des bibliothèques nécessaires
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Déclaration de l'entité et des ports entrées et sorties
entity stockage_de_la_valeur_du_compteur is
    Port ( Q_in : in STD_LOGIC_VECTOR (7 downto 0);                    -- Valeur comptée des cycles du capteur
           F_clk : in STD_LOGIC;                                       -- Fréquence d'horloge
           signal_de_reinitialisation : in STD_LOGIC;                  -- Signal de réinitialisation
           Q_out : out STD_LOGIC_VECTOR (7 downto 0) :="00000000" );   -- Valeur stockée des cycles du capteur
end stockage_de_la_valeur_du_compteur;
-- Définition du fonctionnement
architecture Behavioral of stockage_de_la_valeur_du_compteur is
    signal Q_reg : STD_LOGIC_VECTOR (7 downto 0);
begin
    process(F_clk, signal_de_reinitialisation)
    begin
        if signal_de_reinitialisation = '1' then
            Q_out <= Q_reg ;
        elsif rising_edge(F_clk) then
            Q_reg <= Q_in;
        end if;
    end process;
end Behavioral;
```

Annexe 3.2 : Simulation de la fonction « Stockage de la valeur du compteur »



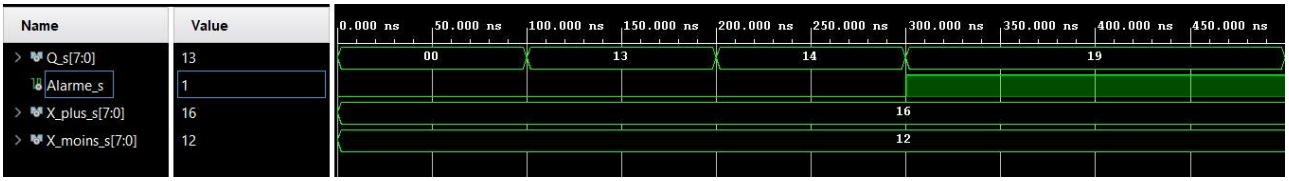
Dans cette simulation, la fréquence de l'horloge est configurée pour fonctionner à 200 MHz, ce qui correspond à une période d'horloge de 5 ns (2,5 ns pour chaque état haut et bas de l'horloge). Le signal d'entrée Q_in_s est incrémenté toutes les 5 ns pour simuler l'évolution des valeurs du compteur. (les nombres sur la figure sont en hexadécimal)

Le signal de réinitialisation signal_de_reinitialisation_s est activé à des intervalles réguliers pour vérifier que la valeur de Q_in_s est correctement stockée dans Q_out_s au moment du déclenchement du signal de réinitialisation. Cette simulation permet de valider que le composant stocke correctement la valeur du compteur lorsque le signal de réinitialisation est activé, et que le système se comporte comme attendu sous l'horloge configurée.

Annexe 4.1 : Code source de la fonction « Comparateur »

```
-- Importation des bibliothèques nécessaires
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Déclaration de l'entité et des ports entrées et sorties
entity comparateur is
    Port ( Q : in STD_LOGIC_VECTOR (7 downto 0);                    -- Valeur comptée
           Alarme : out STD_LOGIC :='0';                            -- Alarme signifiant que le capteur fonctionne mal
           X_plus : in STD_LOGIC_VECTOR (7 downto 0);               -- L'extrémité supérieure de l'intervalle de référence
           X_moins : in STD_LOGIC_VECTOR (7 downto 0));             -- L'extrémité inférieure de l'intervalle de référence
end comparateur;
-- Définition du fonctionnement
architecture Behavioral of comparateur is
begin
    process(Q)
    begin
        if Q = "00000000" then
            Alarme <= '0' ;
        elsif (Q < X_moins or Q > X_plus) then                      -- Vérification que Q appartient à l'intervalle
            Alarme <= '1';                                          -- Déclencher l'alarme
        end if;
    end process;
end Behavioral;
```

Annexe 4.2 : Simulation de la fonction « Comparateur »



Dans cette simulation, la valeur de Q_s, représentant le compteur, est testée contre deux seuils, X_plus_s =18 et X_moins_s =22, qui définissent un intervalle de référence (les nombres sur la figure sont en hexadécimal). Le but est de vérifier si le comparateur peut détecter correctement lorsque Q_s sort de cet intervalle et déclencher une alarme via Alarme_s.

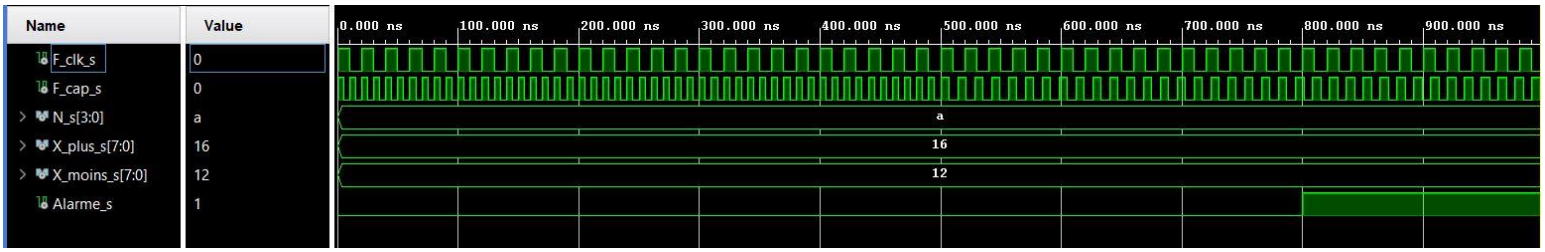
Au début de la simulation, Q_s est défini sur une valeur en dehors de l'intervalle pour observer la réponse immédiate de l'alarme. Ensuite, à chaque intervalle de 100 ns, Q_s est réglé sur différentes valeurs. Dans un premier temps, Q_s est fixé à 19, une valeur légèrement au-dessus de X_moins_s. Puis, Q_s est ajusté à 20, une valeur à l'intérieur de l'intervalle, permettant de vérifier que l'alarme ne se déclenche pas lorsque Q_s reste dans les limites spécifiées. Enfin, Q_s est augmenté à 25, une valeur supérieure à X_plus_s, pour confirmer que l'alarme se déclenche correctement lorsque Q_s dépasse le seuil supérieur.

Cette simulation permet ainsi de valider le bon fonctionnement du comparateur, en s'assurant que l'alarme est activée uniquement lorsque la valeur de Q_s sort des limites définies par X_plus_s et X_moins_s.

Annexe 5.1 : Code source de l'assembalge du mécanisme de test tout entier

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mecanisme_de_test_robuste is Port(
       F_clk : in STD_LOGIC;                                    -- Fréquence de l'horloge
       F_cap : in STD_LOGIC;                                    -- Fréquence du capteur
       N : in STD_LOGIC_VECTOR (3 downto 0);                    -- Nombre de cycles
       X_plus : in STD_LOGIC_VECTOR (7 downto 0);               -- Extrimité supérieure de l'intervalle de réference
       X_moins : in STD_LOGIC_VECTOR (7 downto 0);              -- Extrimité inférieure de l'intervalle de réference
       Alarme : out STD_LOGIC);                                 -- Signal d'alarme
end mecanisme_de_test_robuste;

architecture Structural of mecanisme_de_test_robuste is
    signal signal_de_reinitialisation : STD_LOGIC;              -- Signal de réinitialisation
    signal Q_in : STD_LOGIC_VECTOR (7 downto 0);                -- Le nombre des cycles comptées
    signal Q_out : STD_LOGIC_VECTOR (7 downto 0);               -- La valeur stockée des cycles comptées

component specification_de_temps_de_mesure is Port (
        F_clk : in STD_LOGIC;                                   -- Horloge principale
        N : in STD_LOGIC_VECTOR (3 downto 0);                   -- Nombre de cycles d'horloge
        signal_de_reinitialisation : out STD_LOGIC := '0');     -- Signal de réinitialisation
end component;

component compteur_des_cycles_du_capteur is
    Port ( F_cap : in STD_LOGIC;                                -- Fréquence du capteur
           signal_de_reinitialisation : in STD_LOGIC;           -- Signal de réinitialisation
           Q : out STD_LOGIC_VECTOR (7 downto 0));              -- Valeur comptée des cycles du capteur
end component;

component stockage_de_la_valeur_du_compteur is Port(
   Q_in : in STD_LOGIC_VECTOR (7 downto 0);                     -- Valeur comptée des cycles du capteur
   F_clk : in STD_LOGIC;                                        -- Fréquence d'horloge
   signal_de_reinitialisation : in STD_LOGIC;                   -- Signal de réinitialisation
   Q_out : out STD_LOGIC_VECTOR (7 downto 0) :="00000000" );    -- Valeur stockée des cycles du capteur
end component;


component comparateur is
    Port ( Q : in STD_LOGIC_VECTOR (7 downto 0);                -- Valeur comptée
           Alarme : out STD_LOGIC :='0';                        -- Alarme signifiant que le capteur fonctionne mal
           X_plus : in STD_LOGIC_VECTOR (7 downto 0);           -- L'extrémité supérieure de l'intervalle de référence
           X_moins : in STD_LOGIC_VECTOR (7 downto 0));         -- L'extrémité inférieure de l'intervalle de référence
end component;

begin

inst_specification_de_temps_de_mesure : specification_de_temps_de_mesure PORT MAP(
    F_clk => F_clk,
    N => N,
    signal_de_reinitialisation => signal_de_reinitialisation);

inst_compteur_des_cycles_du_capteur : compteur_des_cycles_du_capteur PORT MAP(
    F_cap => F_cap,
    signal_de_reinitialisation => signal_de_reinitialisation,
    Q => Q_in);

inst_stockage_de_la_valeur_du_compteur : stockage_de_la_valeur_du_compteur PORT MAP(
    Q_in => Q_in,
    F_clk => F_clk,
    signal_de_reinitialisation => signal_de_reinitialisation,
    Q_out => Q_out);

inst_comparateur : comparateur PORT MAP(
    Q => Q_out,
    X_plus => X_plus,
    X_moins => X_moins,
    Alarme => Alarme);

end Structural;
```

Annexe 5.2 : Simulation du fonctionnement du mécanisme de test robuste



Dans cette simulation, deux signaux d'horloge, F_clk_s et F_cap_s, sont utilisés pour modéliser le comportement du système de test robuste. La fréquence d'horloge F_clk_s est configurée pour fonctionner à 50 MHz. Ce signal d'horloge représente l'horloge principale du système. Le signal F_cap_s, représentant la fréquence du capteur, varie au cours de la simulation pour tester la robustesse du mécanisme en présence de fluctuations de fréquence. Pendant les 50 premiers cycles, F_cap_s fonctionne à une fréquence de 100 MHz correspondant à une situation normale sans perturbations. Ensuite, la fréquence de F_cap_s est modifiée à environ 71,4 MHz pour simuler une perturbation.

Les valeurs X_plus_s et X_moins_s définissent l'intervalle de référence pour la détection des anomalies. La simulation vérifie si le signal d'alarme Alarme_s est correctement déclenché lorsque la fréquence du capteur sort de cet intervalle, ce qui indiquerait un changement significatif et potentiellement une attaque par injection de fautes. (Les nombres sur la figure sont en hexadécimal)

REFERENCES

[1] Emplacement de LCIS, https://www.google.fr/maps/place/LCIS.

[2] Photo du batiment D du laboratoire, https://lcis.fr/.

[3] Mariam Esmaili, “On Power-off Attacks Potential against Security Sensors”, page 3.

[4] Mariam Esmaili, “On Power-off Attacks Potential against Security Sensors”, page 4.

Résumé

Ce rapport présente le développement d'un système de protection pour les systèmes embarqués, visant à détecter des variations de fréquence qui pourraient indiquer des tentatives d'attaques physiques. Le projet s'appuie sur un mécanisme de comptage précis des cycles de capteurs, avec des étapes clés incluant la spécification du temps de mesure, le comptage des cycles, la capture des valeurs mesurées et la comparaison avec une référence prédéfinie. Les résultats théoriques ont été validés par des simulations sur Vivado, confirmant l'efficacité du système. Cependant, des tests sur une carte FPGA et avec des capteurs réels soumis à des attaques permettraient d'évaluer davantage la robustesse du dispositif et d'identifier d'autres facteurs influençant son fonctionnement.

Mots Clés : Systèmes embarqués, Sécurité matérielle, Attaques matérielles, Fréquence des capteurs.

Summary

This report presents the development of a protection system for embedded systems, aimed at detecting frequency variations that could indicate attempted physical attacks. The project is based on a precise sensor cycle counting mechanism, with key steps including specification of measurement time, cycle counting, capture of measured values and comparison with a predefined reference. The theoretical results were validated by simulations on Vivado, confirming the system's efficiency. However, tests on an FPGA board and with real sensors under attack would enable us to further assess the robustness of the device and identify other factors influencing its operation.

Keywords : Embedded systems, Hardware security, Hardware attacks, sensors frequency.

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

LCIS
Laboratoire de Conception et d'Intégration des Systèmes

Ecole Publique d'Ingénieurs de Caen

6 boulevard Maréchal Juin, CS 45053
14050 CAEN cedex 04

ministère éducation nationale enseignement supérieur recherche
Normandie Université
Cti Commission des titres d'ingénieur
cdefi Conférence des Directeurs des Écoles Françaises d'Ingénieurs
European Accreditation of Engineering Programmes EUR-ACE®
CONFÉRENCE DES GRANDES ÉCOLES