



République Algérienne Populaire et Démocratique
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche
Scientifique
وزارة التعليم العالي والبحث العلمي
Ecole National Supérieur des Technologies Avancées
المدرسة الوطنية العليا للتكنولوجيات المتقدمة



Département : Génie électrique Et Informatique Industrielle

Mémoire de fin d'étude en vue de l'obtention du diplôme
De Master

-Filière-

Télécommunications

-Spécialité-

Systèmes de Télécommunications et Réseaux

- Thème –

Implémentation hardware de la fonction SHA3: Etat de l'art

Réalisé par
MECHITOUA Zina

Les membres de Jury :

M. BOUDIAR Toufik	Président
Mme BOUCHAMA Samira	Promoteur
Mme BELHACHAT Faiza	Co-Promoteur
M. ABBADI Belkacem	Examineur
M. GUELLAL Amar	Examineur

Alger, le 19/09/2024

Année universitaire 2023 –2024

Implémentation hardware de la fonction SHA3: État de l'art

Mechitoua Zina
École Nationale Supérieure
des Technologies Avancées

Bouchama Samira
École Nationale Supérieure
des Technologies Avancées

Belhachat Faiza
École Nationale Supérieure
des Technologies Avancées

Abstract— Une fonction de hachage cryptographique est une fonction qui calcule une empreinte de taille fixe à partir d'un message de taille variable. LE SHA-3 (Secure Hash Algorithm 3) est le dernier membre de la famille SHA, il est basé sur une nouvelle approche appelée la construction éponge. Dans ce travail nous nous intéressons à la fonction SHA3 et les avantages de son implémentation sur différentes plateformes. Un exemple pratique est donné pour l'implémentation de cette fonction sur un processeur de traitement de signal. La carte choisie est le DSP C6713.

Mots clés —FPGA, GPU, DSP, SHA3, implémentation Hardware

I. INTRODUCTION

Implémenter la cryptographie est aussi complexe que de concevoir des systèmes de sécurité. Une implémentation incorrecte peut compromettre la sécurité d'un système pourtant éprouvé. Il est donc crucial de bien choisir la manière dont nous allons mettre en œuvre nos algorithmes cryptographiques, ainsi que les plateformes sur lesquelles ils seront déployés. Les algorithmes cryptographiques modernes peuvent être réalisés de deux façons principales : soit en utilisant du matériel spécialisé dédié à la cryptographie, soit en déployant des logiciels sur du matériel général. Chacune de ces approches présente des avantages et des inconvénients, que nous allons examiner plus en détail ci-après.

Dans le cadre de ce travail nous nous intéressons particulièrement à l'implémentation Hardware de la fonction de hachage SHA3 et nous proposons une implémentation sur une carte DSP.

Dans la section 2 nous donnons un aperçu sur les avantages et inconvénients des solutions de chiffrement basées sur le logiciel et les solutions basées sur hardware. Dans la section 3 nous abordons particulièrement le cas de l'implémentation de la fonction de Hachage SHA3. Dans la section 4 nous présentons un cas pratique qui est l'implémentation de la fonction SHA3 sur un DSP. La plateforme choisie est la carte DSP 6713.

II. IMPLÉMENTATION D'ALGORITHMES CRYPTOGRAPHIQUES

Lors de l'implémentation de systèmes cryptographiques, plusieurs facteurs doivent être pris en compte. La performance inclut le temps d'exécution et l'utilisation des ressources nécessaires pour les opérations cryptographiques. La sécurité assure la résistance aux attaques et la conformité aux normes reconnues. Le coût englobe le prix du matériel spécialisé et les dépenses liées au développement et à la maintenance. La maintenance et le support se réfèrent à la

facilité de mise à jour et à la qualité de la documentation. Enfin, la compatibilité évalue l'intégration et l'interopérabilité avec d'autres systèmes. Ces facteurs sont essentiels pour garantir une implémentation efficace et sécurisée [1].

II.1 Chiffrement basé sur le logiciel

Le chiffrement basé sur logiciel consiste à sécuriser les données via des programmes informatiques. Ces logiciels sont installés sur des ordinateurs hôtes pour chiffrer et déchiffrer les données. Ils partagent souvent les ressources de traitement avec d'autres programmes, ce qui peut affecter les performances globales du système. Développer des solutions logicielles est généralement moins coûteux car il n'y a pas besoin de matériel spécifique. Pour des applications sécurisées moins complexes, les besoins cryptographiques fondamentaux tels que les fonctions de hachage et la vérification des signatures numériques peuvent être réalisés par des logiciels. Cette méthode est plus simple que l'intégration de protections matérielles et n'affecte pas la taille des puces. Les solutions de chiffrement logiciel sont plus courantes que les solutions matérielles car elles peuvent protéger tous les appareils d'une organisation. Disponibles pour tous les principaux systèmes d'exploitation, elles protègent les données au repos, en transit, et sur divers appareils. Le chiffrement logiciel inclut souvent des fonctionnalités de sécurité supplémentaires, impossibles à obtenir directement avec le matériel. De plus, les solutions logicielles offrent une grande flexibilité dans le choix des algorithmes, car l'ajout d'un nouvel algorithme se fait par une simple mise à jour logicielle. Cependant, la sécurité de ces solutions dépend de celle du système d'exploitation de l'appareil. Une faille dans le système d'exploitation peut compromettre la sécurité du chiffrement. Le logiciel de chiffrement peut aussi être complexe à configurer pour des utilisations avancées et pourrait être désactivé par les utilisateurs. La dégradation des performances est également un problème notable. Le niveau de sécurité d'un module cryptographique logiciel est limité par le mécanisme qui protège la confidentialité et l'intégrité de la mémoire qu'il utilise

II.2 Chiffrement matériel et plates-formes d'implémentation

Le chiffrement basé sur le matériel sécurise les données en utilisant un processeur dédié et distinct. Cette méthode est plus rentable pour les grandes entreprises car elle n'exige aucune installation logicielle supplémentaire. Elle offre une capacité de débit et une vitesse supérieure dans les environnements à grande échelle. Une solution matérielle est idéale pour la protection des données sensibles sur des appareils portables comme les ordinateurs portables ou les clés USB, et elle est

également efficace pour protéger les données au repos. Les disques contenant des données sensibles, telles que les applications bancaires, les données privées ou les informations gouvernementales, sont mieux protégés par des clés matérielles, qui restent efficaces même en cas de vol de l'équipement ou si celui-ci est installé sur un autre ordinateur. Le chiffrement matériel offre une meilleure résistance aux attaques courantes. En général, un attaquant ne pourra pas mener d'attaques par force brute ¹ sur un système chiffré matériellement car le module cryptographique arrêtera le système et pourra compromettre les données après un certain nombre de tentatives. En revanche, dans les solutions logicielles, les pirates pourraient localiser et réinitialiser les compteurs ou copier le fichier chiffré sur différents systèmes pour des tentatives de décryptage parallèles. Le chiffrement matériel permet également un traitement plus rapide des algorithmes, un stockage des clés résistant aux altérations et une protection contre le code non autorisé. Cependant, la mise en œuvre matérielle est relativement fixe et prend un temps relativement long à être réalisée. Même une simple implémentation d'un algorithme sur DSP est plus complexe à concevoir, coder et tester que la même tâche réalisée en logiciel. De plus, les solutions matérielles peuvent être coûteuses et sont liées à un appareil spécifique, ce qui signifie que les mises à jour ne peuvent se faire que par le remplacement de l'appareil. Lors de la mise en œuvre de la cryptographie, quatre options principales de plateformes sont disponibles : CPU, GPU, FPGA et ASIC. Chaque plateforme offre des avantages et des compromis uniques [2][3].

CPU (Central Processing Unit)

Les CPU sont conçus pour le traitement informatique général et sont optimisés pour exécuter des instructions de manière séquentielle. Les modèles récents de CPU sont capables de gérer plusieurs threads simultanément et utilisent des techniques telles que la prédiction de branchement et le parallélisme au niveau des instructions pour optimiser les performances. Les principaux avantages des CPU incluent leur grande flexibilité, leur facilité de programmation et leur disponibilité sur une large gamme d'appareils. Cependant, ils présentent des limites en matière de traitement parallèle, ce qui les rend moins efficaces pour des tâches spécifiques par rapport à du matériel spécialisé[4].

GPU (Graphics Processing Unit)

Les GPU sont conçus pour le traitement parallèle, avec de nombreux petits cœurs capables de gérer plusieurs opérations simultanément. Cette architecture les rend particulièrement adaptés à des tâches telles que le rendu graphique et l'analyse de données. Les avantages des GPU incluent une puissance de traitement parallèle élevée et une grande efficacité dans le traitement des données vectorielles. Cependant, ils sont moins adaptés aux tâches nécessitant un traitement séquentiel et peuvent être moins efficaces pour des tâches qui ne sont pas conçues pour une exécution parallèle[5].

FPGA (Field Programmable Gate Array)

Les FPGA sont composés de blocs logiques programmables reliés par des interconnexions programmables, permettant une configuration adaptée à des tâches spécifiques. Leurs principaux avantages sont leur grande flexibilité, leur capacité à être reconfigurés et leur bonne efficacité énergétique pour des applications spécifiques. Cependant, ils présentent des

inconvénients, notamment une complexité accrue en termes de programmation et de configuration, et une efficacité moindre par rapport aux ASIC pour des tâches dédiées[6].

ASIC (Application Specific Integrated Circuit)

Les ASIC sont spécialement conçus pour exécuter des tâches spécifiques, offrant des performances et une efficacité exceptionnelle pour ces tâches précises. Leurs avantages incluent une performance élevée, une faible consommation d'énergie et une grande efficacité pour des applications dédiées. Cependant, ils présentent des inconvénients tels que des coûts de développement élevés, des délais de conception prolongés et un manque de flexibilité une fois fabriqués. En résumé, le choix entre ces plateformes dépend des exigences spécifiques de l'application cryptographique, y compris les besoins en performance, la flexibilité, les considérations de coût et la nature des tâches à effectuer[7].

DSP

Les DSP (Digital signal processor) sont exploités afin de tirer profit des différentes techniques de traitement de signal dans la conception de systèmes embarqués efficaces et sécurisés. Dans [8] des exemples d'exploitation de la transformation de resynchronisation pour l'accélération des primitives de fonction de hachage et des exemples d'implémentation d'algorithme à clé publiques sur DSP sont donnés. Ils abordent également les scénarios pouvant être utilisés dans le cas d'attaques par canaux auxiliaires (side channel attacks) qui utilisent des informations telles que le temps d'exécution, la consommation d'énergie, etc.

La figure 1 présente une comparaison de ces différentes plateformes en termes de performances.

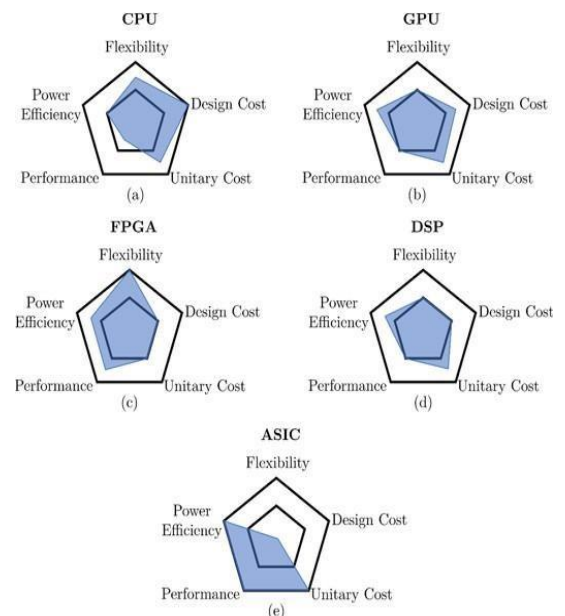


Figure 1 : Comparaison entre des plateformes Hardware.

¹ Une attaque par force brute consiste à déchiffrer des mots de passe ou des clés de cryptage en essayant toutes les combinaisons possibles.

III. IMPLEMENTATION DE LA FONCTION SHA3

Lors de la création d'une implémentation, plusieurs options sont disponibles. Il est possible de choisir des solutions basées entièrement sur le matériel, entièrement sur le logiciel, ou opter pour des approches hybrides. Les solutions matérielles utilisent des composants dédiés, tandis que les solutions logicielles s'appuient uniquement sur des ressources logicielles. Les approches hybrides, comme les Environnements de Développement Intégré (ISE) ou les Processeurs d'Instructions Spécifiques à l'Application (ASIP), combinent des processeurs standard avec du matériel spécialisé pour améliorer l'efficacité et la performance.

La figure 2 représente les différentes stratégies d'optimisation pour implémenter l'algorithme SHA-3 en utilisant des solutions logicielles et matérielles. Du côté logiciel, l'optimisation peut se faire au niveau de l'algorithme (parallélisation, vectorisation, etc.) et du compilateur (optimisation des drapeaux, prélecture, etc.). Pour les solutions matérielles, l'optimisation des performances inclut le dépliage et le pipeline, tandis que l'exploration de conception s'intéresse à la manipulation de l'état de Keccak et au partage des ressources. Une solution hybride combine ces méthodes pour une efficacité maximale [9].

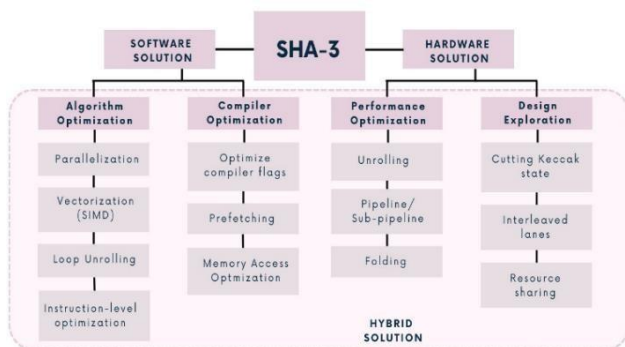


Figure 2 : Optimisation de l'implémentation de SHA-3 : Solutions logicielles et matérielles [5].

III.1 Solutions logicielles et hybrides

L'optimisation logicielle d'un algorithme est essentielle pour maximiser l'utilisation des ressources disponibles et améliorer les performances globales du système. En perfectionnant le code, en tirant parti des fonctionnalités spécifiques des processeurs et en appliquant des techniques avancées comme la parallélisation et la vectorisation, il est possible d'accélérer significativement l'exécution des algorithmes, même sans accélérateurs matériels dédiés. Cette stratégie est particulièrement adaptée aux architectures comme RISC-V, ARM ou FPGA, où l'optimisation logicielle devient un atout clé pour améliorer l'efficacité sans recourir à du matériel supplémentaire. Parmi les techniques utilisées le Parallélisme, la vectorisation, Optimisation au niveau des instructions et au niveau de la compilation [10].

Les solutions hybrides, qui allient matériel et logiciel, représentent une approche efficace pour améliorer les performances des systèmes complexes. Elles exploitent la

puissance brute du matériel tout en préservant la flexibilité et l'adaptabilité du logiciel. Dans ce modèle, les tâches de calcul intensif sont confiées à des accélérateurs matériels spécialisés, tandis que le logiciel prend en charge les aspects nécessitant des mises à jour fréquentes ou une logique adaptable. Par exemple, l'architecture RISQ-V, décrite par Fritzmann et al. [11], combine des accélérateurs sur mesure dans une conception hybride, démontrant comment la synergie entre matériel et logiciel peut répondre à des besoins complexes tout en offrant une grande flexibilité.

III.2 Solutions matérielles

Lors de la mise en œuvre de Keccak en matériel, il est important de choisir les bons paramètres de conception et stratégies en fonction des objectifs et des contraintes spécifiques de l'application. Ces objectifs concernent principalement la vitesse, l'efficacité énergétique et l'espace utilisé. Dans cette section, nous examinerons les différents éléments à considérer pour l'implémentation matérielle de Keccak, en nous concentrant sur ces paramètres clés.

Déroulement (Unrolling) : Le déroulement consiste à répliquer et exécuter en parallèle plusieurs itérations de la boucle de calcul, améliorant ainsi le débit, notamment pour le hachage de messages uniques. Par exemple, si une boucle dans le calcul de Keccak prend normalement n cycles pour être complétée, le déroulement avec un facteur de 2 permettrait d'exécuter deux itérations de la boucle en parallèle, réduisant ainsi le temps total à $n/2$ cycles, mais en utilisant plus de composants matériels.

Pipeline : Le pipeline améliore le débit dans le hachage multi-messages en traitant plusieurs messages simultanément. Deux types de pipeline sont couramment utilisés : **Pipeline classique** permet de traiter plusieurs messages simultanément, chaque message étant à une étape différente du pipeline et le **Sous-pipeline** peut être appliqué à l'intérieur d'un même tour. Par exemple, si un tour de Keccak se compose de plusieurs sous-étapes (comme p , q , r), un sous-pipeline pourrait être utilisé pour commencer la sous-étape suivante avant que la précédente ne soit complètement terminée sur un autre message.

Repliement (Folding) : Le repliement réduit l'utilisation de l'espace en calculant chaque tour de Keccak sur plusieurs cycles d'horloge, selon un facteur de repliement choisi. Par exemple, si une boucle dans le calcul de Keccak prend normalement n cycles pour être complétée sans repliement, en appliquant un facteur de repliement de 2, on diviserait le calcul en deux étapes séquentielles. Chaque étape de la boucle serait exécutée sur deux cycles d'horloge distincts, ce qui porterait le temps total de la boucle à $2n$ cycles. Cependant, cela permettrait d'utiliser beaucoup moins de composants matériels, car les mêmes circuits sont réutilisés pour effectuer chaque étape du calcul.

Découpe de l'état de Keccak : La gestion de l'état de Keccak peut être optimisée par des techniques de découpe, telles que le traitement par tranches, par plans, ou l'entrelacement des bits, pour mieux utiliser l'espace et les ressources. Par

exemple, si l'état complet de Keccak (qui est un tableau de 1600 bits) doit normalement être traité en un seul cycle d'horloge, la découpe de l'état avec un facteur de 2 permettrait de diviser cet état en deux parties égales de 800 bits chacune. Au lieu de traiter tout l'état en une seule fois, chaque partie de 800 bits serait traitée sur deux cycles d'horloge distincts. Cela double le nombre de cycles nécessaires pour traiter l'état complet, mais réduit la complexité matérielle et l'espace requis, car les mêmes unités de traitement peuvent être réutilisées pour chaque moitié de l'état.

Lanes entrelacées (Interleaved Lanes) : L'entrelacement des bits permet de diviser les lanes de 64 bits en morceaux plus petits, optimisant ainsi l'utilisation des ressources matérielles. Par exemple, si un lane de Keccak (qui est une série de 64 bits) doit normalement être traité en un seul cycle d'horloge, l'entrelacement des lanes avec un facteur de 2 permet de diviser chaque lane en deux parties de 32 bits chacune. Plutôt que de traiter les 64 bits d'un lane en une seule fois, les deux parties de 32 bits sont traitées sur deux cycles d'horloge distincts, en entrelaçant les bits des différentes lanes. Cela allonge le temps de traitement à $2n$ cycles au lieu de n cycles, mais permet de réduire la largeur des unités de traitement nécessaires, économisant ainsi des ressources matérielles tout en répartissant les bits sur plusieurs cycles pour une gestion plus flexible des données.

Partage des ressources (Resource Sharing) : Le partage des ressources est une technique d'optimisation pour maximiser l'utilisation efficace des ressources disponibles, particulièrement important dans les FPGA et ASIC, pour réduire l'empreinte matérielle et améliorer l'efficacité énergétique. Par exemple, si plusieurs opérations de Keccak nécessitent chacune une unité de calcul distincte, le partage des ressources permettrait d'utiliser une seule unité de calcul pour exécuter ces différentes opérations de manière séquentielle. Par exemple, si trois opérations distinctes nécessitent normalement 3 unités matérielles pour être exécutées en parallèle en un cycle d'horloge, le partage des ressources permettrait d'utiliser une seule unité matérielle pour exécuter ces trois opérations sur trois cycles d'horloge. Cela triple le nombre de cycles nécessaires, portant le temps total à $3n$ cycles au lieu de n cycles, mais réduit considérablement l'espace matériel requis, en maximisant l'utilisation des ressources disponibles tout en minimisant la complexité matérielle.

III.3 Exemples de travaux

Plusieurs solutions d'implémentation de la fonction de hachage SHA3 ont été proposées sur FPGA. Certains travaux font des propositions de différentes structures (simple en pipeline, en repliement (folding) dans le but d'améliorer l'efficacité, à travers une conception plus petite et à une réduction du chemin critique (optimisation des délais de propagation des signaux à travers le circuit) [12] [13]. Aussi, différents éléments de son architecture peuvent être utilisés : Dans [13] l'utilité LUT-6 qui est une table de correspondance (Look-up Table) de l'FPGA à 6 entrées est exploitée, donc elle peut générer $2^6 = 64$ sorties.

Dans [14], les auteurs présentent deux conceptions d'algorithme de hachage (SHA-3) à faible consommation sur un réseau de portes programmable sur site (FPGA) utilisant l'unité intégrée de traitement du signal numérique (DSP48E), l'une pour les environnements à ressources restreintes et l'autre pour les applications à grande vitesse. Les sept équations de SHA-3 sont logiquement optimisées pour des organisations pipeline à trois et quatre étages pour nos conceptions compactes et à grande vitesse, respectivement. La méthode proposée dans ce travail nous a permis de déplacer les fonctions logiques intensives telles que XOR, AND & NOT de l'algorithme SHA-3 de la primitive LUT vers ces tranches DSP inutilisées et d'offrir la possibilité d'économiser davantage de ressources logiques pour la mise en œuvre d'autres parties du système sur la même puce.

IV. PRÉSENTATION DES DSP

Les DSP, ou processeurs de signal numérique, sont des composants électroniques spécialisés qui traitent les signaux numériques. Leur configuration architecturale, basée généralement 4 sur le modèle de Harvard modifié, est conçue pour améliorer l'efficacité des calculs. Elle intègre des méthodes avancées comme la segmentation de la mémoire et l'utilisation de pipelines, permettant d'optimiser les performances de traitement. Comparés aux méthodes analogiques, les DSP offrent une flexibilité et une capacité de traitement bien supérieures, les rendant essentiels dans de nombreuses applications modernes [15].

IV.1 Description générale du DSK6713 DEVELOPMENT BOARD

Le TMS320C6713 est un processeur de signal numérique de haute performance créé par Texas Instruments et appartenant à la série C6000. Il est basé sur l'architecture VLIW (Very Long Instruction Word) qui permet l'exécution de jusqu'à 8 instructions par cycle d'horloge, ce qui optimise l'efficacité et accélère le traitement des données. Il dispose d'une fréquence d'horloge pouvant atteindre 225 MHz. Il est doté d'une unité de calcul en virgule flottante double précision. Il intègre 256 KB de SRAM sur puce, garantissant un accès rapide aux données critiques. Aussi, il inclut des ports d'entrée/sortie polyvalents pour les interfaces de communication. Le C6713 DSK (DSP Starter Kit) est une plateforme de développement peu coûteuse et autonome, conçue pour évaluer et développer des applications pour le processeur de signal numérique TMS320C6713 de Texas Instruments. Le kit de développement C6713 DSK comprend généralement plusieurs composants matériels, y compris le DSPC6713, ainsi que des interfaces d'entrée et de sortie telles que des convertisseurs analogique-numérique (CAN) et numérique-analogique (CNA), des ports de communication, des boutons-poussoirs, et des LEDs pour l'interaction avec l'utilisateur. Il dispose également d'un port de débogage pour le développement et le débogage du code. Ce kit est souvent fourni avec un environnement de développement intégré (IDE) tel que code composer studio (CCS) et des bibliothèques logicielles pour faciliter la programmation du DSPC6713. Les utilisateurs peuvent écrire leur code en langage C ou en assembleur, et utiliser les outils fournis pour compiler, déboguer, et télécharger le code sur le DSP pour une exécution en temps réel [16].

IV.2 Architecture du DSKC6713

Le DSK C6713 de Texas Instruments est conçu comme une plateforme de développement complète et fiable pour des applications de traitement du signal numérique. Voici un aperçu de son architecture [14] [16] [17] qui est illustré dans la figure 3 :

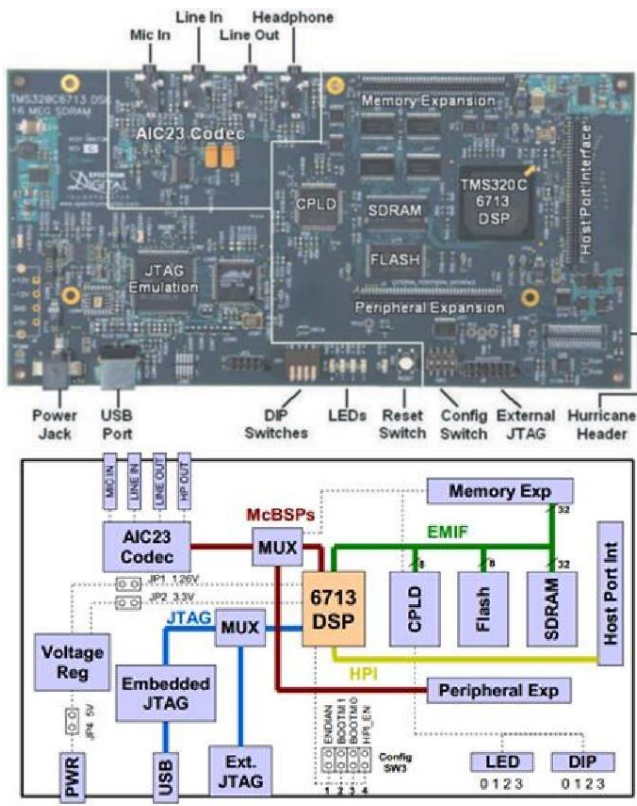


Figure 3 : La carte DSP TMS320C6713 et son architecture.

- Interfaces d'entrée/sortie (I/O) : Les convertisseurs Analogique-Numérique sont utilisés pour convertir des signaux analogiques en numériques, permettant ainsi au processeur de traiter des données analogiques provenant de capteurs ou d'autres sources. Les convertisseurs Numérique-Analogique servent à transformer des signaux numériques en signaux analogiques, utiles pour les sorties audio ou autres dispositifs analogiques. Les ports de communication incluent des ports série pour la communication avec d'autres dispositifs, ainsi qu'une interface USB qui est utilisée pour le téléchargement et le débogage du code. De plus, l'interface mémoire externe (EMIF) permet de connecter des mémoires SDRAM et flash externes, augmentant ainsi la capacité de stockage et les options de mémoire pour les applications développées.

- Périphériques de Contrôle Utilisateur : Les boutons-poussoirs permettent aux utilisateurs de fournir des entrées simples et immédiates pour le contrôle et le test des applications en cours de développement. Les LEDs, quant à elles, servent d'indicateurs visuels pour les sorties et les états du système.

- Extensions et Connecteurs : Les connecteurs d'extension permettent l'ajout de cartes filles, ce qui étend les fonctionnalités du kit en permettant la connexion de périphériques supplémentaires ou de modules spécifiques pour des applications avancées. Ces extensions peuvent inclure des modules de communication, des capteurs supplémentaires, ou des interfaces spécialisées. Par ailleurs, le port de débogage JTAG intégré est crucial pour le développement et le débogage en temps réel et le diagnostic des applications complexes.

IV.3 Langage de programmation et outils de développements

Le langage de programmation le plus répandu pour travailler avec des DSPs est le C. Sa popularité repose sur son équilibre entre performance et flexibilité, ainsi que sur le large éventail de compilateurs et de bibliothèques optimisées disponibles pour les DSP. Les autres langages utilisés pour programmer les DSP incluent l'assembleur, MATLAB/Simulink pour le prototypage rapide et Python pour la facilité d'utilisation et l'accès à des bibliothèques scientifiques étendues.

Pour développer des applications sur des processeurs de signal numérique (DSP) tels que ceux de Texas Instruments, un outil couramment utilisé est Code Composer Studio (CCS). CCS est un environnement de développement intégré (IDE) qui offre un large éventail de fonctionnalités pour le développement, le débogage et le déploiement d'applications DSP. Il fournit un éditeur de code avancé, des outils de compilation et de gestion de projet, ainsi que des fonctionnalités de débogage telles que le suivi en temps réel, la visualisation des variables et le profilage du code.

V. IMPLEMENTATION DE LA FONCTION SHA3 SUR DSP

Les étapes de l'implémentation de la fonction SHA3 sur le DSP C6713 sont comme suit :

Branchement de la carte TMS6713 au PC à travers le câble USB (Figure 4).



Figure 4 : Branchement de la carte DSP.

Lancement du Code composer Studio (CCS version 7.3). La figure 5 représente son interface, les principales fenêtres représentées sur la figure sont :

- 1) Explorateur de projet

- 2) Fichier main contenant le code
- 3) Console de débogage pour indiquer les étapes du débogage
- 4) La fenêtre problème indique les erreurs et les Warning (avertissement) liés au programme.

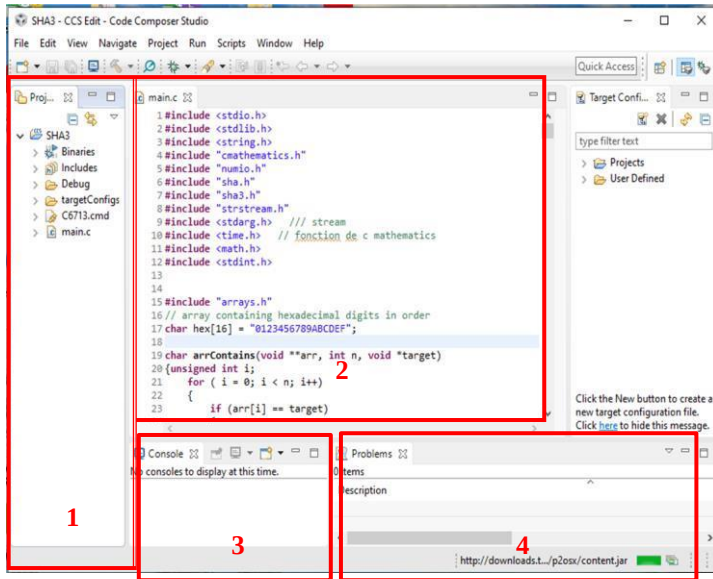


Figure 5 : Interface du Code Composer Studio.

L'algorithme de la fonction SHA3 englobe l'initialisation des paramètres, le padding du message, le mapping, la permutation de l'état avec la fonction f de Keccak, la troncature de la sortie, et enfin la génération du haché final. La mise en œuvre de la fonction de hachage SHA-3 dans un programme C repose sur l'intégration de cinq processus fondamentaux : Theta (θ), Rho (ρ), Pi (π), Chi (χ) et Iota (ι) qui sont détaillées dans le premier chapitre. Voici ces processus en langage C :

```
void sha3_keccak_f(unsigned long long A[5][5])
{ for (int i = 0; i < SHA3_NR; i++)
{ // THETA =====
    unsigned long long C[5]; unsigned long long D[5];
    // calculate C
    for (int x = 0; x < 5; x++)
    { C[x] = 0L;
      for (int y = 0; y < 5; y++)
      { C[x] ^= A[y][x];
      } }
    // calculate D
    for (int x1 = 0; x1 < 5; x1++)
    { D[x1] = C[(x1 + 4) % 5] ^ leftRotateLL(C[(x1 + 1) % 5], 1); }
    // Mettre le resultat dans la matrice A
    for (int x2 = 0; x2 < 5; x2++)
    { for (int y = 0; y < 5; y++)
```

```
      { A[y][x2] ^= D[x2];
      } }
    // RHO, PI =====
    unsigned long long B[5][5];
    memset(B, 0, 5 * 5 * sizeof(unsigned long long));
    for (int x3 = 0; x3 < 5; x3++)
    { for (int y = 0; y < 5; y++)
      { B(2 * x3 + 3 * y) % 5][y] = leftRotateLL(A[y][x3],
      sha3_rotConst[y][x3]);
      } }
    // CHI =====
    for (int x4 = 0; x4 < 5; x4++)
    { for (int y = 0; y < 5; y++)
      { A[y][x4] = B[y][x4] ^ ((~B[y][(x4 + 1) % 5]) & B[y][(x4 + 2) % 5]);
      } }
    // IOTA =====
    A[0][0] ^= sha3_roundConsts[i]; } }
```

Les étapes de l'implémentation :

Implémentation de l'algorithme

L'algorithme de la fonction SHA3 a été traduit en code en utilisant le langage C.

Test et débogage

Nous avons effectué plusieurs tests pour évaluer notre implémentation. Les exemples ci-dessous sont données à titre indicatif. Ce sont des vecteurs de test disponible sur le site web de NIST.

Tableau 1: Exemples de l'application de la fonction de hachage SHA3 sur quelques messages donnés.

Message texte	The quick brown fox jumps over the lazy dog
Hachage mode 256	3c6ee413b3b8e4b2b1e1b2c3b4a5e6f7d8e9a0b1 c2d3e4f5a6b7c8d9e0f1a2b
Hachage Mode 512	4c1c1e2e3f4f5e6f7d8e9a0b1c2d3e4f5a6b7c8d9 e0f1a2b3c4d5e6f7g8h9i0
Message texte	(vide)
Hachage mode 256	60e893e6d54d8526e55a81f98bfac5da236bb203e8 4ed5967a8f527d5bf3d4a4
Hachage Mode 512	e307daea2f0168daa1318e2faa2d67791e9d8e036 92a6f7d1eb974e664fe721e81a47b4cf3d0eb19ae5d 57afa19a095941cad5a5c050774ad56a8e5e21105757

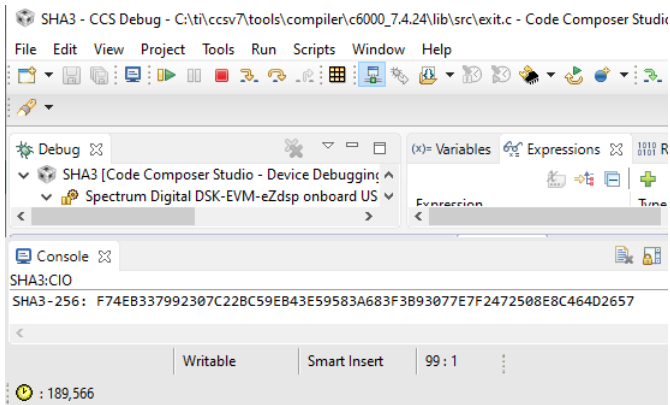


Figure 6 : Résultat donné en utilisant CCS.

<https://bob.nem.ninja/test-vectors.html>

Optimisation

Le Code Composer Studio nous offre la possibilité d'améliorer les performances du programme sur DSP. Il existe deux optimisations, l'optimisation par rapport au code et aux performances et par rapport à la vitesse d'exécution. En appliquant des performances, on passe de 330,52 cycles à 189,56 cycles d'horloge.

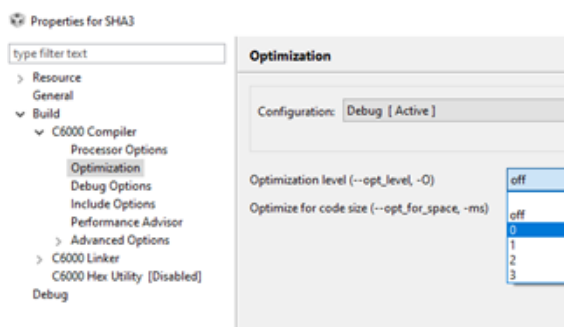


Figure 7 : Optimisation par rapport aux performances.

VI. CONCLUSION

En conclusion, ce travail a permis d'explorer les différentes méthodes d'implémentation de la cryptographie, tant sur des plateformes matérielles que logicielles, en mettant en lumière les avantages et les inconvénients de chaque approche.

Il est donc important de choisir la solution la plus adaptée en fonction des besoins de l'application et en considérant les différences entre les plateformes en termes de performance, consommation d'énergie, etc.

L'étude de la fonction de hachage SHA3 a également été essentielle, car elle représente une avancée majeure dans le domaine de la cryptographie moderne. Enfin, l'implémentation de la fonction de hachage sur un DSP C6713 a démontré la faisabilité et l'efficacité de l'utilisation de processeurs numériques pour des tâches cryptographiques. Cette réalisation pratique a non seulement mis en évidence les capacités de traitement du DSP, mais a également ouvert la voie à de futures recherches sur l'optimisation des algorithmes de hachage pour assurer l'authentification des données dans des systèmes de communication embarquée.

Bibliographie

- [1] L. BESSON, "Conception, analyse et implémentation d'algorithmes de chiffrement symétrique sur FPGA", Thèse de doctorat, Université de Limoges, 2013.
Hackaday.io, "CPU, GPU, FPGA, and ASIC - Each has their place". Disponible sur : <https://hackaday.io/project/168291-cpu-gpu-fpga-and-asic-each-has-their-place>.

[fpga-and-asic-each-has-their-place.](#)

- [2] D-Central, "CPU vs GPU vs FPGA vs ASIC for Crypto Mining," D-Central Blog. Disponible sur : <https://d-central.tech/cpu-vs-gpu-vs-fpga-vs-asic-for-crypto-mining>.
- [3] Knežević, Miroslav & Batina, Lejla & Mulder, Elke & Fan, Junfeng & Gierlichs, Benedikt & Lee, Yong & Maes, Roel & Verbauwhede, Ingrid. (2010). Signal Processing for Cryptography and Security Applications. 10.1007/978-1-4419-6345-1_7.
- [4] M. Durbhakula, "Branch Prediction Related Optimizations for Multithreaded Processors", IEEE, 2019.
- [5] "GPU Computing Advantages And Disadvantages," Restack.io, 2021. Disponible : <https://www.restack.io>.
- [6] F. Fallahlalehzari, "FPGA vs GPU for Machine Learning Applications: Which one is better?" Aldec, 2022.
- [7] A. Kumar, "Challenges and Benefits of Using ASICs in High-Volume Production," IEEE Journal of Semiconductor Circuits, Mar. 2021.
- [8] Dolmeta A, Martina M, Masera G. Comparative Study of Keccak SHA-3 Implementations. *Cryptography*. 2023; 7(4):60. <https://doi.org/10.3390/cryptography7040060>
- [9] J. Choi et al., "Optimizing SHA-3 Implementations on General-Purpose Processors with Software Techniques," *IEEE Transactions on Computers*, Oct. 2020.
- [10] T. Fritzmann, T. Pöppelmann, and G. Sigl, "RISQ-V: Tightly Coupled RISC-V Accelerators for Post-Quantum Cryptography," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020
- [11] M.M.Sravani, C.H.Pallavi, DESIGN OF COMPACT IMPLEMENTATION OF SHA-3(512) ON FPGA, International Research Journal of Engineering and Technology (IRJET), Volume: 02 Issue: 02, 2015
- [12] Rao, Muzaffar & Neue, Thomas & Grout, Ian. (2020). Secure Hash Algorithm-3(SHA-3) implementation on Xilinx FPGAs, Suitable for IoT Applications. International Journal on Smart Sensing and Intelligent Systems. 7. 1-6. 10.21307/ijssis-2019-018.
- [13] Dur-e-Shahwar Kundi, Arshad Aziz, A low-power SHA-3 designs using embedded digital signal processing slice on FPGA, Computers & Electrical Engineering, Volume 55, 2016, Pages 138-152.
- [14] Texas Instruments. "C6713 DSK Technical Reference". Disponible en ligne. "TMS320C6713 DSP Overview." Texas Instruments, 2023. Disponible sur : <https://www.ti.com/product/TMS320C6713>.
- [15] "C6713 DSP Kit (DSK) for TI TMS320C6713." Spectrum Digital, 2023. Disponible <https://community.element14.com/products/devtools/product-pages/w/documents/25964/spectrum-digital-c6713-dsp-kit-dsk-for-ti-tms320c6713>.
- [16] "TMS320C6713 DSK User's Guide." Texas Instruments, 2023. Disponible sur : <https://www.ti.com/lit/ug/spru301a/spru301a.pdf>.