MajecSTIC 2010
Bordeaux, France, du 13 au 15 octobre 2010

Sécurisation des communications dans une architecture multi-processeurs

Pascal Cotret[1], Jérémie Crenne[1] et Guy Gogniat[1]

1 : Laboratoire Lab-STICC, Université de Bretagne-Sud, Rue de Saint-Maudé - BP 92116, 56321 Lorient cedex - France.

Contact : pascal.cotret@univ-ubs.fr, jeremie.crenne@univ-ubs.fr, guy.gogniat@univ-ubs.fr

---

Résumé
La production de systèmes embarqués sécurisés ne cesse de progresser depuis plusieurs années. Ceci est dû au fait que des failles de sécurité potentielles sont présentes dès que l'on manipule des données sensibles ou des informations privées, ce qui rajoute de la complexité au développement d'un tel produit. Par conséquent, il est désormais indispensable de penser à la sécurité tout au long du flot de conception afin de garantir une protection globale des systèmes embarqués. La protection des communications est un problème fondamental étant donné que la majorité des données circulent à travers l'architecture de communication du système. Dans cet article, nous nous concentrons sur ce point et une solution est proposée pour échanger des données dans un environnement sécurisé et implanter un mécanisme de surveillance des communications dans une architecture multi-processeurs. Le travail présenté dans ce papier fait partie du projet SecReSoC financé par l'ANR. Afin de valider les résultats, un cas d'étude est proposé : il s'agit d'un token USB capable de chiffrer ou déchiffrer « à la volée » des fichiers stockés sur un ordinateur.

Abstract
The production of secured embedded systems is strongly increasing since several years. This trend is mainly due to factors such as manipulation of sensitive data and private information but also system complexity which lead to potential vulnerabilities. Thus it becomes mandatory to deal with security all along the design flow in order to guarantee a global protection of embedded systems. Protection of communication is a key issue as almost all sensitive data are traveling through the communication architecture. In this paper, we target this point and propose a solution to secure data exchanges and to monitor communication within a multiprocessor system. The work presented in this paper is part of the SecReSoC national project funded by the ANR. In order to validate our work a USB token able to cipher or decipher « on-the-fly » files stored on a computer is considered.

Mots-clés : MPSoC, architecture de communication, systèmes embarqués, protocoles de sécurité, contre-mesures.

Keywords: MPSoC, communication architecture, embedded systems, security protocols, countermeasures.

---

1. Introduction

Grâce aux évolutions des technologies d'intégration de composants électroniques, il est désormais possible d'inclure des dizaines d'éléments de calcul dans un circuit. Les applications relatives à la sécurité requièrent des traitements de plus en plus complexes afin de fournir à l'utilisateur des fonctionnalités avancées. Etant donné que les nouvelles technologies sont en constante progression, les mécanismes qui assurent la sécurité doivent posséder une certaine évolutivité afin d'être à jour avec les nouvelles versions des algorithmes et des protocoles mis en jeu.

Etant donné la complexité des systèmes embarqués, il est nécessaire d'analyser précisement l'organisation et les fonctionnalités des éléments de sécurité tout en conservant un bon compromis entre la surface et la latence globale : il s'agit d'un point critique du flot de développement car le produit final doit avoir un coût acceptable mais également être résistant face à des menaces extérieures.
L'objectif de ce travail, qui fait partie du projet ANR SecReSoC, est d'améliorer la sécurité des technologies reconfigurables telles que les FPGA [1]. Cet article se concentre sur les communications inter-processeurs dans une architecture du type MPSoC [2] qui contient (en plus des processeurs) des mémoires internes (c'est-à-dire des blocs intégrés dans le FPGA) et externes (par exemple, une barette de RAM), des modules d'entrée-sortie pour les interactions avec l'utilisateur (USB, RS232). Une architecture de communication interne connecte tous ces éléments et fait en sorte que les échanges de données soient conformes aux politiques de sécurité que les développeurs du système auront intégrées. Pour valider les travaux, un token de sécurité USB (tels que ceux produits par IronKey et Netheos) est utilisé comme étude de cas.

La suite de l'article est organisée de la façon suivante : la section 2 propose un rapide état de l'art sur la sécurisation d'architectures MPSoC. La section 3 décrit la composition d'une architecture multi-processeurs généralisée. La section 3.2 identifie les points de faiblesses de l'architecture décrite dans la section 3. Ensuite, des solutions sont proposées dans la section 4 et appliquées au cas du token USB. Des premières contre-mesures sont explicitées dans la section 5. La section 6 propose une évaluation des performances de la solution développée. Finalement, la section 7 conclut l'article et invite à réfléchir sur les efforts qui restent à faire pour sécuriser une architecture multi-processeurs.

2. Etat de l'art

Avec l'avancement des nouvelles technologies de silicium et la miniaturisation des composants, il est désormais possible d'intégrer plusieurs processeurs dans une seule et même puce. Par conséquent, plus il y a d'unités, plus la sécurité d'un système embarqué devient problématique [6] [5]. Lors de la conception, il faut penser à l'architecture de communication entre les différentes unités de traitement et les processeurs. Il existe principalement deux technologies afin de mettre en oeuvre les communications :
– Le bus : il est généralement utilisé pour connecter quelques processeurs. Plusieurs standards co-existent : AMBA et CoreConnect, par exemple.
– Le *Network-on-Chip* (NoC ou réseau sur puce) : plus adapté à des systèmes contenant plusieurs dizaines d'unités de calcul et IP [3].
Toutes les architectures de communication présentent des failles de sécurité et plusieurs travaux ont été menés pour proposer des solutions qui résistent à un modèle d'attaque donné [5]. En ce qui concerne les bus, on peut citer les travaux de Charkradhar et al. [1] qui exposent des éléments de sécurité implantés aux interfaces de chaque IP du SoC : ce sont les *Security Enforcement Interface (SEI)*. Chaque SEI analyse les données qui transitent par l'IP qui lui est attachée et renvoie des informations à un gestionnaire principal, le *Security Enforcement Module (SEM)*.
De nombreux travaux se sont penchés vers une solution dédiée pour les NoCs. De par leur structure (composée de routeurs et d'interfaces réseaux), les NoCs sont sujets à des attaques différentes

1. Field Programmable Gate Array
2. Multi-Processor System-on-Chip
3. Intellectual Property, un bloc fonctionnel de l'architecture

qui exploitent les faiblesses du réseau formé par les routeurs [4].
Evain et al. [2] proposent une solution similaire à [1] : des contrôles sont effectués à chaque interface et un gestionnaire s'occupe de rassembler les informations et de mettre à jour chaque interface sécurisée. Il existe une autre approche qui consiste à utiliser les interfaces comme un filtre qui divise l'IP à laquelle il est connecté en plusieurs zones avec différentes politiques de sécurité (dans le cadre de [2], les auteurs se limitent à des accès mémoire) : ce filtre, appelé *Data Protection Unit (DPU)*, contrôle si l'adresse à laquelle le système veut accéder est autorisée ou non. Une extension à cette solution a été proposée en ajoutant différents capteurs directement dans les interfaces réseau [3] : ces capteurs sont capables de signaler un accès non autorisé ou une réduction de la bande passante du circuit.
Cet article propose une solution qui s'inspire des travaux existants pour une architecture de communication basée sur une technologie de type bus. De nouvelles contre-mesures sont également proposées afin d'étendre le périmètre de protection du système (analyse du format des données, politique de sécurité étendue). Ainsi, la contribution proposée apporte une couche de sécurité supplémentaire par rapport aux solutions existantes pour protéger les communications au sein d'une architecture MPSoC.

3. Architecture multi-processeurs

3.1. Présentation de l'architecture

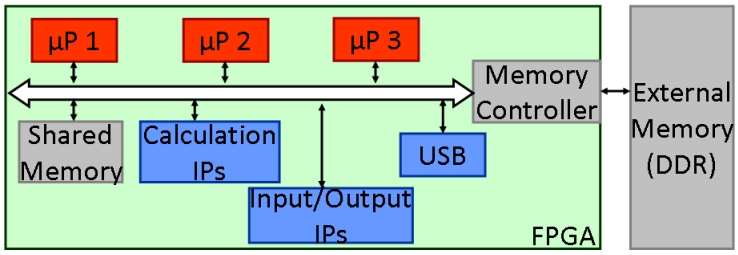
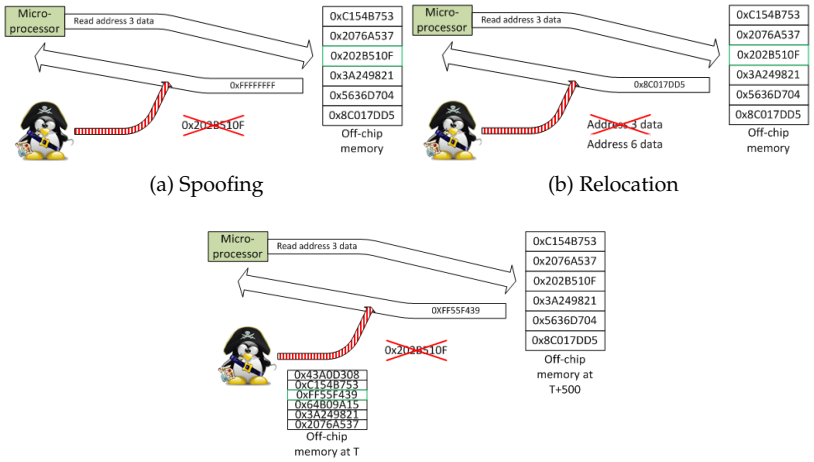


FIGURE 1 – Architecture généralisée d'un système multi-processeurs

Cette catégorie d'architectures ne contient pas uniquement des processeurs, elle contient également d'autres éléments tels que des mémoires (embarquées dans le FPGA ou en externe sous forme de barrettes), des périphériques d'entrée-sortie pour interagir avec l'utilisateur (écran LCD, boutons, liaison série RS 232, port USB,...) et des blocs de traitement (génériques ou spécifiques aux domaines d'études). Les applications relatives à la sécurité ne requièrent généralement pas un grand nombre de coeurs de calcul : jusqu'à présent, une architecture fondée sur une technologie de bus est souvent suffisante pour atteindre les performances requises.

3.2. Modèle de menaces

Le système embarqué présenté précédemment présente des failles qu'un attaquant pourrait exploiter pour récupérer des données sensibles telles que des clés cryptographiques. On peut globalement distinguer deux catégories d'attaques :
– Les attaques sur les mémoires.
– Les attaques relatives aux processeurs.
Dans un premier temps, les attaques par canaux cachés (observation du temps ou de la consommation) ne sont pas prises en compte dans le modèle de menaces.
Il y a des menaces qui pèsent sur les mémoires (internes et externes) : le *spoofing*, le *replay* et la *relocation*. Ces attaques peuvent altérer le fonctionnement du système ou même exécuter un code malveillant au lieu du code implanté à l'origine.



(a) Spoofing (b) Relocation

(c) Replay

FIGURE 2 – Attaques sur les mémoires

Une attaque de type *spoofing* (figure 2a) arrive lorsqu'un attaquant injecte une donnée aléatoire sur le bus : la donnée qui est lue n'a plus aucune signification pour le micro-processeur responsable de la requête et provoque un dysfonctionnement du système. Une attaque de *relocation* (figure 2b) provient lorsqu'une instruction est copiée d'une adresse mémoire vers une autre : au lieu d'avoir la donnée située à l'adresse 3, le micro-processeur recevra la donnée située à l'adresse 6. Le saut d'adresse peut permettre d'exécuter un code malveillant situé dans un espace mémoire normalement non utilisé. L'attaque *replay* (figure 2c) est similaire à la *relocation* sauf que la donnée renvoyée est celle qui existait au même emplacement à un instant précédent.
Si on regarde le circuit reconfigurable FPGA sur lequel est implantée l'architecture multi-processeurs, le bus se révèle être le lien entre les processeurs : ce sont là que toutes les données circulent. Dans un mode de fonctionnement normal (figure 3), le processeur A peut envoyer des données au processeur B qui effectue un traitement et renvoie le résultat à l'utilisateur du système (par exemple, via la liaison RS232). Si un attaquant est capable de se substituer à l'un des processeurs ou de se greffer sur l'architecture de communication entre A et B, alors l'utilisateur final aura un résultat erroné.
Il y a clairement un besoin d'authentifier (de vérifier l'origine) de chaque entité de l'architecture MPSoC et de contrôler rigoureusement les séquences d'exécution qui régissent son fonctionnement normal. En outre, les mécanismes de protection doivent s'adapter aux contextes d'exécution et aux menaces instantanées en proposant la mise en place d'une politique de sécurité dynamique. Si l'attaquant est capable d'identifier la chaîne de configuration de la sécurité, il peut neutraliser l'ensemble du système.

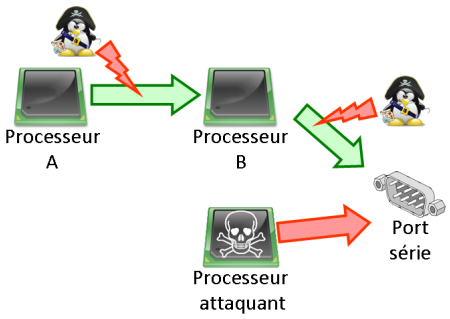


FIGURE 3 – Attaques sur les processeurs

4. Sécurisation d'une architecture multi-processeurs

4.1. Etude de cas : le token USB

Dans le cadre de ce travail, on considère le modèle de menaces défini précédemment au cas d'un token USB. Ce système, qui se présente sous la forme d'une clé USB contient la logique nécessaire pour chiffrer ou déchiffrer « à la volée » des fichiers stockés sur un ordinateur sans avoir besoin d'installer un logiciel ou un driver. La figure 4 (qui est un cas particulier de la figure 1) présente l'architecture d'un tel système avec les mécanismes de sécurité qui sont décrits ultérieurement.

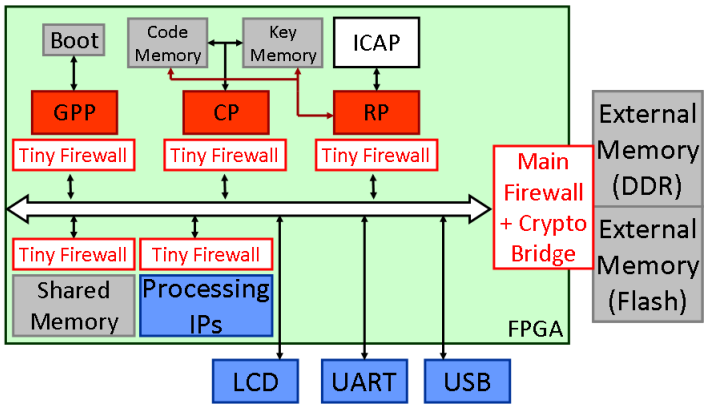


FIGURE 4 – Architecture sécurisée d'un token USB

Le coeur du système est composé de trois processeurs :

- **General Purpose Processor (GPP) :** Ce processeur exécute l'application principale du système sous un environnement multi-tâches. Par exemple, il est en charge de récupérer les fichiers à chiffrer (ou déchiffrer) par le port USB et de les transmettre au CP.
- **Cryptographic Processor (CP) :** Le CP utilise un algorithme de chiffrement (par exemple, l'AES dans ses différents modes) qui utilise des clés cryptographiques stockées dans une mémoire interne dédiée. Son rôle principal est de chiffrer (ou déchiffrer) des trames de données.
- **Reconfiguration Processor (RP) :** Le RP gère la reconfiguration globale (ou partielle) du système depuis un serveur de confiance où sont stockées les différentes configurations sous forme de bitstreams (fichier de configuration du FPGA) et de codes (par exemple, code associé au GPP ou au CP). Ces données sont chiffrées et sont traitées par le CP avant d'être exécutées. Ce processeur est également en charge de configurer les mécanismes de sécurité intégrés décrits dans la section suivante.

En dehors des processeurs, l'architecture contient des éléments mémoires :

- Une mémoire de code (*Code Memory*) et une mémoire clé (*Key Memory*) pour le cryptoprocesseur (CP).
- Une mémoire externe qui contient le code des GPP et RP.
- Une mémoire de boot pour l'initialisation du processeur généraliste GPP.
- Une mémoire interne partagée (*Shared Memory*) pour le stockage des données temporaires.

Enfin, il y a des éléments qui servent à interagir avec l'utilisateur tels que l'écran LCD ou le port série. Cette architecture peut être typiquement utilisée pour créer un token USB. Dans ce cas, le GPP reçoit (ou transmet) les trames de données et les communique au CP pour les chiffrer ou les déchiffrer. Le processeur de reconfiguration est également en charge de la configuration du système et du contrôle de sa mise en route (configuration initiale).
Compte tenu de l'architecture, le modèle de menaces détaillé précédemment peut être appliqué. Les mécanismes de sécurité qui protégent le système sont décrits dans les sous-sections 4.2 et 4.3.

4.2. Crypto-bridge

Toutes les données stockées dans la mémoire externe sont chiffrées, c'est-à-dire que si un attaquant arrive à les récupérer, il les obtiendra sous une forme incompréhensible.
Par conséquent, des mécanismes de cryptographie sont nécessaires pour chiffrer (respectivement déchiffrer) les données qui sont écrites (respectivement lues) dans la mémoire externe. Ce *Crypto-Bridge* (CB) a pour caractéristique d'être transparent vis-à-vis de l'architecture principale : il opère en totale indépendance sans influer sur les processeurs et les unités de calcul. Le Crypto-Bridge est un module qui fonctionne de manière « half-duplex » :

– Une donnée en provenance d'un module de l'architecture principale doit être écrite en mémoire externe : celle-ci sera automatiquement chiffrée.
– Un module a besoin d'une donnée située dans la mémoire externe : cette donnée sera déchiffrée avant d'être transmise au module à l'origine de la requête.

4.3. Firewalls

4.3.1. Main Firewall

Le Main Firewall est également implanté au niveau du contrôleur mémoire. Ce module effectue essentiellement l'analyse des séquences d'opérations telles que celles expliquées dans la section 3.2. Il est donc capable de détecter si une opération anormale a lieu au cours du fonctionnement du système et de réagir le cas échéant.

4.3.2. Tiny Firewall

Le Tiny Firewall (TF) est un firewall unique intégré dans chaque IP au niveau de l'interface avec le bus principal du système. Chaque TF vérifie l'intégrité et l'authenticité des données qui sont transmises ou reçues par l'IP et effectue des opérations de monitoring qui sont gérées sous forme de machine à états finis : ces opérations peuvent être dédiées à son IP mais peuvent également être génériques.
Par exemple, le Tiny Firewall du GPP analyse les trames de données à chiffrer ou déchiffrer et vérifie l'intégrité (« Est-ce que le format de la donnée est valide ? ») et l'authenticité (« Est-ce que la donnée provient d'une source sûre ? »).

4.3.3. Topologie des firewalls

Le MF se comporte comme un gestionnaire des TF : il reçoit les informations des TF (issues des différentes machines à états finis) et calcule une nouvelle configuration qu'il pourrait déployer en cas d'attaques (pointillés sur la figure 5). Les TF sont capables de se communiquer leurs états respectifs selon différents niveaux de sécurité (sûr, sécurité moyenne, TF infecté) : par exemple, le TF de l'IP à laquelle on tente d'accéder connaîtra l'état actuel du TF auteur de la requête pour savoir s'il est conforme aux règles de sécurité (figure 5). Par conséquent, cette approche réduit le temps de latence du traitement des données à travers le firewall.
Le Main Firewall est reconfigurable matériellement en injectant une nouvelle configuration du FPGA (grâce à un bitstream) alors que les Tiny Firewalls sont configurés directement par le Main Firewall : on aura ainsi une reconfiguration « temps réel » des contrôles aux interfaces des IPs.

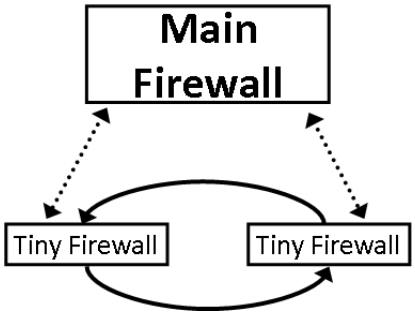


FIGURE 5 – Topologie des firewalls

5. Contre-mesures

5.1. Isolation d'une IP suspecte

Si un TF détecte que l'IP qui lui est associée est attaquée, il crée une zone d'isolation autour de l'IP afin de la déconnecter du reste du système. Par conséquent, l'IP infectée n'aura aucune influence sur le reste du système et toutes les requêtes en provenance ou à destination de cette IP seront ignorées.

5.2. Reconfiguration dynamique

Si un firewall (MF ou TF) détecte une erreur, une reconfiguration du sous-réseau de firewalls peut être nécessaire pour avoir une nouvelle politique de sécurité qui concorde avec les nouveaux besoins du système. Le processeur de reconfiguration doit être capable de reconfigurer le Main Firewall (qui lui-même s'occupe des Tiny Firewalls) en téléchargeant un bitstream partiel. De même, il doit aussi être capable de mettre à jours les différentes mémoires de codes des processeurs GPP [4] et CP [5].

6. Analyse des performances

Afin d'obtenir une première estimation des performances des mécanismes de sécurité proposés, plusieurs éléments de l'architecture de sécurité ont été synthétisés sur une carte ML605 de Xilinx (FPGA Virtex-6 XC6VLX240T). Des estimations en surface (surcoût par rapport à l'architecture de référence) et en latence (si la donnée est disponible) sont ainsi données dans la suite de cette section par l'intermédiaire du tableau 1.

		Estimation des ressources	
		Surface	Latence-Vitesse
Architecture de base (non protégée)		16 407 slices	
Main Firewall	Module de chiffrement	+2,45%	13 cycles | 2,56 Go/s
	Module de monitoring	+4,88%	-
	Module de protection adresses et données	+3,54%	3 cycles
	Total	+10,87%	-
Tiny Firewall	Module de protection adresses et données	+3,54%	3 cycles
	Module de monitoring	+4,88%	-
	Fonction de hachage	+7,45%	65 cycles | 800 Mo/s
	Total	+15,87%	-

TABLE 1 – Estimation en surface et en latence des différents éléments de l'architecture de communication

Architecture de référence (non protégée) : Afin de pouvoir évaluer l'impact des solutions de sécurité, il est important de connaître le coût relatif de la solution de sécurité par rapport à l'architecture fonctionnelle. L'architecture développée contient deux processeur Microblaze (qui symbolisent le processeur généraliste GPP et le processeur de reconfiguration RP), un Crypto-processeur (CP), une mémoire partagée et un contrôleur pour la mémoire externe, et des périphériques d'entrée-sortie (LCD,USB, liaison série RS232).

Main Firewall :

– Chiffrement : correspond au Crypto-Bridge décrit dans la section 4.2. Il exécute un algorithme AES 128 bits classique qui chiffre (ou déchiffre) un bloc de données de 128 bits en 13 cycles

4. General Purpose Processor
5. Cryptographic Processor

d'horloge. Cet algorithme provient d'un coeur opensource de `opencores.org`[6].
– Monitoring : contient uniquement la logique qui permet « d'écouter » ce qui se passe sur le bus système, autrement dit de récupérer toutes les données qui circulent sur le bus.
– Protection adresses et données : version simplifiée de l'*APU* et de la *DPU* décrites dans [1] qui filtrent les accès selon les adresses et les valeurs des données écrites dans la mémoire externe.

Tiny Firewall :

– Protection adresses et données : voir la description du Main Firewall.
– Monitoring : contient la logique d'écoute des données qui circulent sur l'axe IP-bus.
– Fonction de hachage : fonction SHA-256 (*Secured Hash Algorithm*) qui réalise l'authentification d'un bloc de donnée en créant une « image » de taille fixe de ce bloc. Cet algorithme provient d'un coeur opensource de `opencores.org`[7].

Cette première étude permet d'évaluer l'impact des éléments de sécurité vis-à-vis du système. Les élements visant à sécuriser les communications entre les processeurs ont un impact non négligeable au regard de l'architecture complète, que ce soit en termes de surface (environ 10% pour un firewall) ou de latence (chaque firewall introduit des délais supplémentaires inévitables). De plus, ils sont aptes à travailler individuellement mais ils ne communiquent pas entre eux. Ils offrent un premier niveau de protection qui doit être approfondi afin de conserver un compromis raisonnable en terme de surface-latence. Il est clair que la mise en place d'une solution de sécurité a un coût, aussi il est fondamental de proposer des solutions dynamiques qui permettent d'ajuster les mécanismes de sécurité en fonction des menaces potentielles.

7. Conclusion et perspectives

Nous proposons une solution afin de sécuriser les communications au sein d'une architecture multi-processeurs. Ce travail offre une solution de sécurité avancée qui intègre des mécanismes de sécurité mais également des contre-mesures. L'association de firewalls et du Crypto-Bridge améliore le niveau de protection du MPSoC implémenté sur une cible reconfigurable afin d'accroître le niveau de réactivité au modèle de menaces. L'étude concernant le sous-réseau formé par les firewalls et le Crypto-Bridge va être approfondie afin d'optimiser le couple latence/surface, le choix de la technologie d'interconnection des firewalls (pas abordée dans ce travail) sera une des problématiques à résoudre. Le développement de ces mécanismes influera également sur la conception des contre-mesures.

Bibliographie

1. S. Chakradhar, J. Coburn, S. Ravi, et A. Raghunathan. *SECA : Security-Enhanced Communication Architecture*. ACM Press, New York, New York, USA, 2005.
2. S. Evain, J. P. Diguet, R. Vaslin, G. Gogniat, et E. Juin. NoC-centric security of reconfigurable SoC. page 223. IEEE, 2007.
3. L. Fiorin, G. Palermo, et C. Silvano. *A security monitoring service for NoCs*. ACM Press, New York, New York, USA, 2008.
4. L. Fiorin, C. Silvano, et M. Sami. Security aspects in networks-on-chips : Overview and proposals for secure implementations. In *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*, volume 0, page 539, Los Alamitos, CA, USA, 2007. IEEE Computer Society.
5. P. Kocher, R. Lee, G. McGraw, et A. Raghunathan. Security as a new dimension in embedded system design. In *Proceedings of the 41st annual conference on Design automation*, page 753. ACM New York, NY, USA, 2004.
6. S. Ravi, A. Raghunathan, P. Kocher, et S. Hattangady. Security in embedded systems : Design challenges. *ACM Transactions on Embedded Computing Systems (TECS)*, 3(3) :461, 2004.

6. `http://opencores.org/project,aes_core`
7. `http://opencores.org/project,sha_core`