Les attaques matérielles visant la sécurité du bus

Ce chapitre présente les attaques matérielles développées dans le cadre de cette thèse visant la sécurité du bus AXI dans un SoC complexe hétérogène embarquant la technologie ARM TrustZone. Le chapitre commence par un rappel des aspects de la technologie ARM TrustZone liés au bus AXI. Ensuite, il présente la méthode à suivre pour utiliser la technologie ARM TrustZone dans la partie reconfigurable d'un SoC FPGA. A la fin du chapitre, des attaques qui visent à compromettre la sécurité du bus AXI sont présentées. Ces attaques sont mises en place durant la phase de conception du circuit à l'aide de différentes méthodes, notamment l'utilisation de script TCL malicieux, la modification malicieuse du code source d'une IP, ou l'utilisation d'un outil CAO de non-confiance.

Les attaques présentées dans ce manuscrit sont mises en place à l'aide du SoC Xilinx Zynq-7000 vu son emploi dans plusieurs projets académiques et industriels dans différents domaines. Mais ces attaques peuvent aussi ciblées tous les SoCs complexes hétérogènes embarquant la technologie ARM TrustZone.

1. Rappel

Dans un SoC complexe hétérogène embarquant la technologie ARM TrustZone :

- Les interfaces maîtres contrôlent l'état de sécurité des requêtes émises au bus.
- Le bus utilise les signaux de sécurité ARPROT[1] et AWPROT[1] (détaillés dans la section chap1-2.3.1) pour transmettre l'état de sécurité aux interfaces esclaves du système.
- La valeur des signaux de sécurité ARPROT[1] et AWPROT[1] (l'état de sécurité actuelle de l'interface maître émettrice de la requête) est comparée avec l'état de sécurité de l'interface esclave destinatrice. Selon l'architecture du système, cette comparaison est effectuée soit au niveau de l'AXI Interconnect, soit au niveau d'une IP TrustZone (détaillé dans la section chap1-2.2), soit directement au niveau de l'interface esclave de l'IP destinataire.
- Une requête sécurisée est acceptée par toutes les interfaces esclaves du système peu importe leurs états de sécurité. Si elle n'est pas sécurisée, une requête n'est acceptée que par les interfaces esclaves non-sécurisées du système.
- Si une requête non-sécurisée essaye de communiquer avec une interface esclave sécurisé, la requête est rejetée et l'erreur DECERR est envoyée à l'interface maître émettrice en utilisant les signaux de réponse du bus BRESP ou RRESP (détaillés dans la section chap1-1.1).

Les attaques introduites dans la suite de ce chapitre exploitent les vulnérabilités liées au bus AXI dans un SoC complexe hétérogène embarquant la technologie ARM TrustZone.

2. L'utilisation de la technologie TrustZone dans la partie reconfigurable

Comme indiqué dans la section chap1-2.3, l'utilisation de la technologie ARM TrustZone dans la partie reconfigurable est optionnelle dans tous les SoCs FPGA existants. Cette section présente en détail les étapes à suivre pour activer la propagation de l'état de sécurité d'une requête de communication émise vers la partie reconfigurable et le partitionnement des IP dans la partie reconfigurable du SoC FPGA Xilinx Zynq-7000. Le choix a été porté sur l'utilisation du SoC FPGA Xilinx Zynq-7000 vu son emploi dans plusieurs projets académiques et industriels dans

différents domaines. Cependant, les autres SoC FPGA du marché, comme ceux d'Intel, sont très proches.

L'utilisation de la technologie ARM TrustZone dans la partie reconfigurable est présentée dans ce chapitre pour son intérêt dans la compréhension de la suite de ce manuscrit.

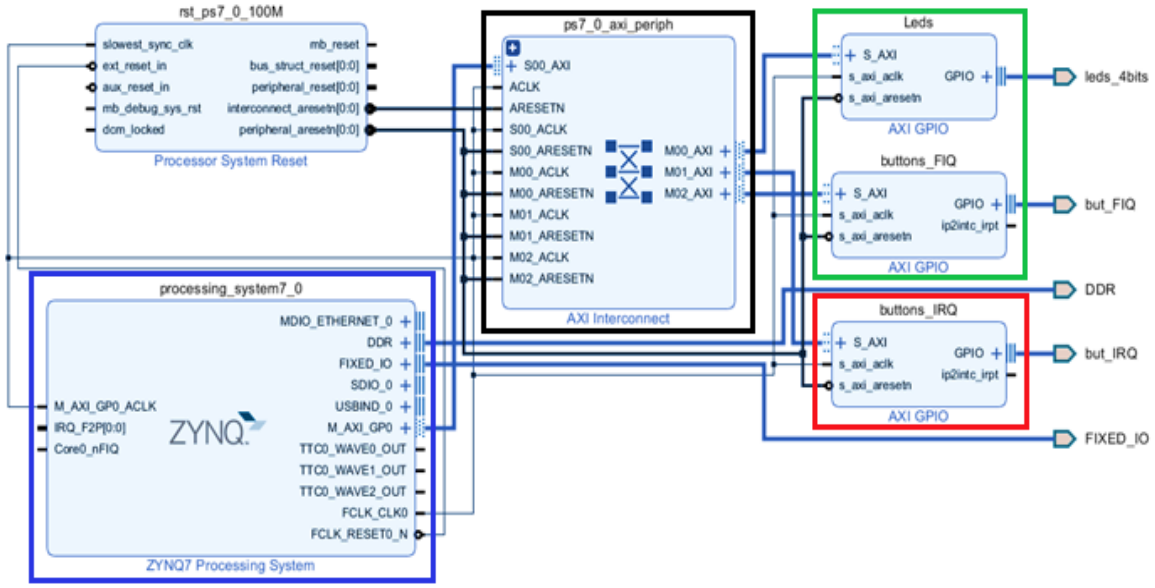

Figure 12: Design créé avec l'outil Vivado de Xilinx

La figure 13, présente le design créé à l'aide de l'outil CAO Vivado de Xilinx. Pour créer ce design nous avons suivi un processus bien précis que nous avons détaillé dans le tutoriel [63] que nous avons rendu disponible à toute la communauté. En suivant scrupuleusement ce tutoriel il est possible d'implémenter un système utilisant la technologie ARM TrustZone dans un SoC FPGA Xilinx Zynq.

Dans la figure 13, le rectangle bleu encercle l'IP ZYNQ7 Processing System qui représente la partie processeur du SoC Xilinx Zynq-7000. Cette IP permet d'effectuer des configurations système, comme la configuration des horloges, l'activation de la propagation de l'état de sécurité des cœurs ARM (l'état de sécurité de la partie processeur) vers la partie reconfigurable, etc. Le reste des blocs constituent les éléments qui sont implémentés dans la partie reconfigurable du SoC. Le rectangle noir encercle l'AXI Interconnect qui permet de connecter les IP entre elles. Le rectangle vert entour les IP déclarées comme sécurisées à l'aide de la fonctionnalité de sécurité de l'AXI Interconnect (fonctionnalité détaillée dans la section chap1-2.3.2), le rectangle rouge entour l'IP déclarée non-sécurisée.

Pour utiliser la technologie TrustZone dans la partie reconfigurable du SoC Xilinx Zynq-7000, il faut passer par les deux étapes suivantes :

- **Etape 1 : L'activation de la propagation de l'état de sécurité du cœur ARM vers la partie reconfigurable du SoC Zynq-7000**

La propagation des signaux de sécurité dans le SoC FPGA Xilinx Zynq-7000 doit être activée pendant la phase de conception. Pour cela, le concepteur doit changer la valeur du paramètre ''***AXI Non Secure Enablement***'' de l'IP ZYNQ7 Processing System (rectangle bleu figure 13) à 1. Sous l'outil Vivado, Le concepteur peut configurer ce paramètre en ligne de commande. Pour cela, le concepteur doit utiliser la commande TCL de l'outil Vivado '***set_property [-dict args] [-quiet] [-verbose] name value objects***'. Cette commande permet de configurer les paramètres d'un objet (l'IP processing_system7_0 dans notre exemple, figure 13) en ligne de commande. La figure 14 présente la commande à exécuter pour changer la valeur du paramètre '***CONFIG.PCW_USE_AXI_NONSECURE***' qui permet d'activer la propagation de l'état de sécurité des requêtes émise vers la partie reconfigurable.

```
set_property -dict [list CONFIG.PCW_USE_AXI_NONSECURE {1}] [get_bd_cells processing_system7_0]
```

Figure 13: Commande TCL Activation de la propagation des signaux de sécurité à la partie reconfigurable

- **Etape 2 : Affecter un état de sécurité aux IP de la partie reconfigurable**

Comme indiqué dans la section chap1-2.2, La technologie TrustZone se base sur l'utilisation de contrôleurs matériels pour partitionner la totalité des ressources du système en sécurisés et non-sécurisés. Pour partitionner les IP en IP sécurisées et IP non-sécurisées dans notre exemple, l'AXI Interconnect intègre une fonctionnalité optionnelle (détaillée dans la section chap1-2.3.2) qui permet de déclarer une IP comme sécurisée. Si cette fonctionnalité n'est pas activée la totalité des IP de la partie reconfigurable sont considérer comme non-sécurisées.

Une fois que la propagation de l'état de sécurité de la partie processeur vers la partie reconfigurable du SoC Xilinx Zynq-7000 est activée (Etape 1 réalisée). Il reste au concepteur à déclarer l'interface maître qui connecte cette IP à l'AXI Interconnect comme sécurisée. Pour réaliser cela en ligne de commande, le concepteur doit utiliser la commande TCL 'set_property' utilisée précédemment pour activer la propagation des signaux de sécurité. Cette commande sera utilisée pour déclarer l'interface M00_AXI et M01_AXI sécurisées et l'interface M02_AXI non-sécurisée. Par exemple pour sécuriser l'interface maître M00_AXI, il faut changer la valeur du

paramètre ***CONFIG.M00_Secure*** de l'AXI Interconnect (***ps7_0_axi_periph***, rectangle noir figure 13) en ***true*** en ligne de commande. La figure 15 présente les 3 lignes de commandes à utiliser pour la réalisation du design figure 13.

```
set_property CONFIG.M00_Secure true [get_bd_cells ps7_0_axi_periph]
set_property CONFIG.M01_Secure true [get_bd_cells ps7_0_axi_periph]
set_property CONFIG.M02_Secure false [get_bd_cells ps7_0_axi_periph]
```

Figure 14: Commande TCL malveillante pour changer l'état de sécurité de l'interface AXI_M00 à non-sécurisée

3. La mise en œuvre des attaques matérielles visant la sécurité du bus AXI

Cette section présente les attaques visant la sécurité du bus AXI dans un SoC complexe hétérogène embarquant la technologie ARM TrustZone. Les attaques, que nous présentons [64], [65], visent les signaux de sécurité AWPROT[1]/ARPROT[1], les signaux de réponse BRESP/RRESP, et l'AXI Interconnect.

Dans la suite de ce chapitre, les attaques sont implémentées à l'aide du SoC Xilinx Zynq-7000, mais celles-ci peuvent toucher tous les SoC embarquant la technologie ARM TrustZone. La figure 16 est une simplification du design de la figure 13. La figure présente deux interfaces, une maître et une esclave, connectées par le bus AXI qui inclus l'AXI Interconnect (détaillé dans la section chap1-2.3.2). L'interface maître contrôle l'état de sécurité des requêtes de communication destinées à l'interface esclave. Le bus AXI transmet la requête à l'AXI Interconnect qui compare l'état de la requête avec l'état de sécurité statique de l'interface esclave. Si la comparaison est correcte, l'AXI Interconnect transmet la requête à l'interface esclave avec un état de sécurité neutre et envoie un OK comme réponse à l'interface maître. Dans le cas contraire, l'AXI Interconnect rejette la requête et envoie une erreur à l'interface maître. L'état de sécurité de l'interface esclave est statiquement fixé durant la phase de conception en utilisant la fonctionnalité de l'AXI Interconnect.

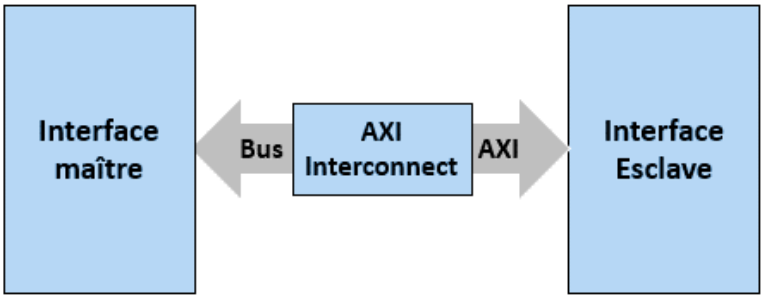

Figure 15 : Communication entre interface maître et esclave

Durant une transaction normale du bus AXI, si l'état logique des signaux AWPROT[1]/ARPROT[1] est à un niveau logique bas à l'entrée de l'AXI Interconnect, l'interface maître peut accéder à toutes les IP connectées à cet AXI Interconnect. Dans le cas contraire, l'accès par l'interface maître aux IP non-sécurisées est restreint. En cas de violation d'accès, l'AXI Interconnect envoie une erreur à l'aide des signaux de réponse à l'interface maître ou au contrôleur du système. Le récepteur de l'erreur effectue le nécessaire (déclencher une interruption par exemple) pour informer le système de l'erreur.

3.1. Compromettre les signaux de sécurité du bus AXI

Pour contrôler les signaux de sécurité, AWPROT[1]/ARPROT[1], un cheval de Troie est placé avant l'AXI Interconnect (le contrôleur TrustZone qui compare les états de sécurité) comme indiqué dans la figure 17. Le cheval de Troie modifie les signaux avant leur comparaison avec l'état de sécurité statique de l'IP ciblée (Etat fixé durant la conception par le designer). Si le cheval de Troie fixe la valeur des deux signaux à un niveau logique bas, l'attaque permettra une escalade de privilège, elle autorisera l'interface maître non-sécurisée à utiliser les IP sécurisées connectées à l'AXI Interconnect. Dans le cas contraire, l'attaque crée un déni de service. Les IP déclarées sécurisées sont condamnées pour la vie du système, tout accès les ciblant (même par une interface maître sécurisée) génère un message d'erreur et la requête de communication est rejeter par le contrôleur.

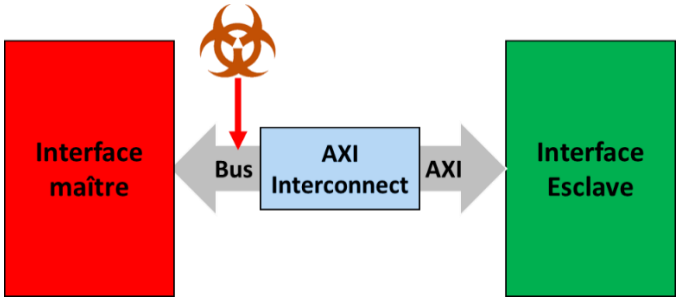

Figure 16 : accéder aux ressources sécurisées depuis le monde non-sécurisé

3.1.1. Exemple de modification matérielle malveillante visant les signaux de sécurité AWPROT[1]/ARPROT[1] du bus AXI

Dans la figure 18, l'interface maître contrôle la sécurité des requêtes émises vers l'interface esclave. L'interface esclave est l'interface esclave du contrôleur TrustZone (l'AXI Interconnect dans la figure 17) qui compare la valeur des signaux de sécurité avec l'état de sécurité de l'IP destinatrice. La figure 18 présente quatre modifications matérielles malveillantes pour réaliser l'escalade de privilège et le déni de service. Les modifications malveillantes visent à modifier le signal de sécurité AWPROT[1] du canal d'adresse d'écriture (les 5 canaux du bus AXI sont détaillés dans la section chap1-1.1) avant sa comparaison avec l'état de sécurité de l'IP destinatrice. Les mêmes modifications peuvent être utilisées pour compromettre le signal ARPROT[1] du canal d'adresse de lecture du bus AXI. Certaines des modifications matérielles malveillantes utilisent un trigger qui permet de déclencher la fonction malveillante uniquement au moment où on en a besoin, et revenir au fonctionnement normal. Le trigger permet de cacher la fonction malveillante durant le test structurel du circuit et éliminer les soupçons.

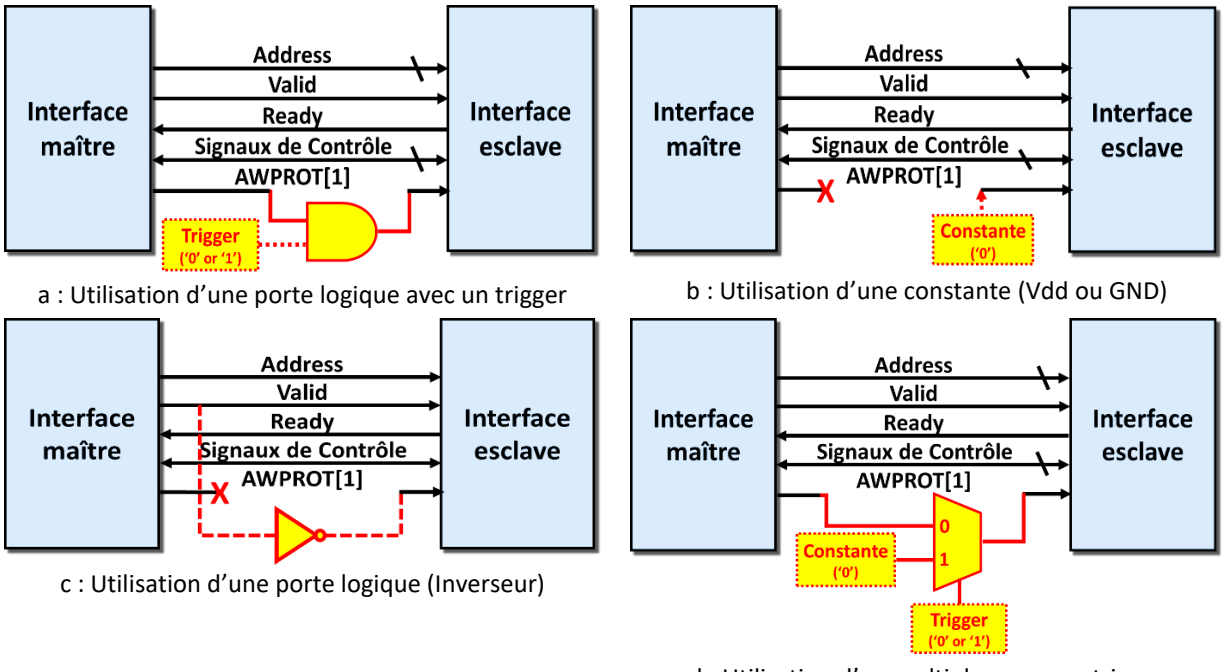

Figure 17: Quatre exemples de modifications matérielles malicieuses ciblant le signal de sécurité AWPROT[1]

Dans la modification matérielle malicieuse de la figure 18-a, le signal AWPROT[1] est contrôlé par une porte logique ''et'' entre l'interface maître et l'interface esclave du contrôleur TrustZone. La porte logique ''et'' a comme entrée, un signal trigger utilisé pour activer la fonction

malveillante et le signal AWPROT[1]. Si le trigger est à l'état logique haut, le système fonctionne normalement. Dans le cas contraire, toutes les transactions émises par l'interface maître seront vues comme sécurisées par le contrôleur TrustZone (escalade de privilège). Cette modification malicieuse permet au monde non-sécurisé d'utiliser toutes les ressources sécurisées connectées au contrôleur TrustZone visé par l'attaque.

Dans La modification matérielle malicieuse de la figure 18-b, le signal AWPROT[1] est corrompu entre l'interface maître et l'interface esclave. Et l'entrée du signal AWPROT[1] sur l'interface esclave est connectée à une source constante, soit GND pour le fixer au niveau logique bas, soit Vdd pour le fixer au niveau logique haut. Si la constante est fixée à GND, la modification malicieuse permet de réaliser une escalade de privilège depuis le monde non-sécurisé. Dans le cas contraire, la modification malicieuse permet de réaliser un déni de service, condamnation des IP sécurisées.

Dans la modification matérielle malicieuse de la figure 18-c, le signal AWPROT[1] est corrompu entre les deux interfaces comme dans la modification de la figure 18-b, et l'entrée du signal AWPROT[1] est connectée à la sortie d'un inverseur du signal Valid du canal d'adresse d'écriture. Cette modification se base sur le fait que durant le handshake (signal Valid active), l'interface esclave fait une capture de l'état des signaux de contrôle (enregistre la valeur des signaux dans des registres dédiés), signale AWPROT[1] inclus. Du coup, si l'interface esclave récupère l'inverse du signal Valid pour le signal de sécurité AWPROT[1], toutes les transactions provenant de l'interface maître seront sécurisées pour l'AXI Interconnect. Cette modification permet de réaliser seulement une escalade de privilège.

Dans la modification matérielle malicieuse de la figure 18-d, le signal AWPROT[1] est contrôlé par un multiplexeur avec deux entrées, le signal AWPROT[1] et une constante, et un trigger comme signal de contrôle du multiplexer. Le trigger permet de choisir entre le fonctionnement normal ou malveillant. Le choix de la constante (Vdd ou GND) dépend de l'attaque à implémenter.

3.1.2. L'utilisation d'un script malicieux

Cette section présente l'utilisation d'un script malicieux pour réaliser automatiquement les modifications présentées dans la figure 18. Le script malicieux peut être généré par un outil d'automatisation de flot de conception tiers malveillants (Outil de CAO), ou développer par un concepteur compromis. Les scripts TCL sont utilisés partout durant la phase de conception et cela

pour plusieurs raisons, notamment pour l'automatisation du processus d'implémentation, la vérification de l'exactitude d'une conception (DRC), la livraison d'un design à un autre utilisateur, etc. Un script malicieux constitue un chemin d'attaque efficace et une grande menace durant la conception d'un SoC. Mais la plus grande menace est l'exécution des scripts provenant de tiers non fiable sans vérification, soit par un manque de temps, soit par un manque d'expérience.

Par exemple, il n'est pas difficile de réaliser la modification de la figure 18-b en utilisant les commandes TCL. Un attaquant peut utiliser les deux commandes TCL présentées dans la figure 19, la première commande déconnecte le signal AWPROT[1] de l'entrée de l'AXI Interconnect, et la deuxième commande connecte l'entrée débranchée au GND afin de faire apparaitre comme des transactions sécurisées toutes les transactions ciblant les IP connectées à l'AXI Interconnect. Pour ne pas laisser de trace dans le journal des commandes de l'outil CAO, certaines commandes ont des options notamment ***–notrace*** et ***–quiet*** qui permettent de ne pas apparaitre aux yeux du concepteur.

Disconnect_net **–quiet** –net [get_nets *design/signal_AXI_AWPROT[1]*] –objets [get_pins *design/AXI_Interconnect/S_AXI_AWPROT[1]*]

Connect_net –hier **–quiet** –net *design/<constant0>* –objets [get_pins *design/AXI_Interconnect /S_AXI_AWPROT[1]*]

Figure 18: Deux commandes TCL à ajouter au script pour modifier automatiquement et discrètement le code RTL

3.2. Compromettre les signaux de réponse BRESP/RRESP

Après la vérification des signaux de sécurité par l'AXI Interconnect, une réponse est envoyée à l'interface maître en utilisant les signaux de réponse à 2 bits BRESP et RRESP (détaillés dans la section chap1-1.1). Ces signaux sont essentiels pour la gestion de la sécurité dans un SoC complexe hétérogène embarquant la TrustZone. En conséquence, ces signaux sont des cibles d'attaque appropriées.

3.2.1. Exemple de modification matérielle visant les signaux BRESP/RRESP

La figure 20 montre une modification matérielle malveillante possible du signal BRESP du canal de réponse du bus AXI (cette modification peut être utilisée aussi pour attaquer le signal RRESP du canal lecture de donnée). Certaines de ces modifications matérielles malveillantes présentées dans la figure 18 sont également valables dans ce contexte. Les deux bits du signal renvoyés à l'interface maître comme réponse sont contrôlés par un multiplexeur 4x1 avec un trigger comme signal de contrôle. Le trigger permet de choisir de forcer ou non la réponse de

l'esclave. En fonction du scénario d'attaque, ce multiplexeur pourrait être plus petit et capable de ne forcer qu'une ou deux des réponses possibles.

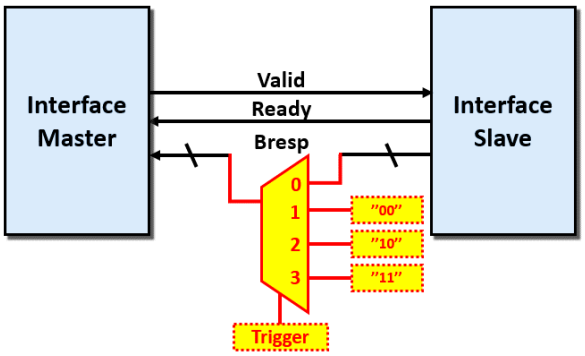

Figure 19: Exemple de modification matérielle malveillante du signal BRESP

Lorsque le signal BRESP est forcé sur la valeur binaire "00", la réponse de l'esclave est forcée sur OK même en cas de violation de l'état de sécurité. Dans cette attaque, la requête de communication est rejetée par l'AXI Interconnect et l'interface maître n'est pas informé pour prendre les mesures nécessaires contre la violation de sécurité.

Lorsque le signal BRESP est forcé sur les valeurs binaires "10" ou "11", la réponse de l'esclave est forcée sur SLVERR ou DECERR, même dans le cas d'une transaction correcte. Cette attaque permet de créer un déni de service, l'interface maître reçoit un message d'erreur pour chaque requête qu'elle émet au bus.

3.2.2. Modification de la configuration de la LUT

Pour compromettre le signal BRESP, une autre forme d'attaque possible consiste à modifier malicieusement la configuration des LUT (LookUp Table) après synthèse du design. En effet, le vecteur d'initialisation stocké dans une LUT est responsable de sa configuration et donc de la fonction logique entre les entrées et les sorties de la LUT. Un développeur interne malveillant qui a accès au design après synthèse peut modifier malicieusement les vecteurs d'initialisation des LUT critiques, ce qui modifie leur fonction logique ainsi que le comportement normal du design. La modification peut être effectuée directement depuis l'onglet de configuration de la LUT dans l'outil CAO, ou en utilisant la commande TCL présentée dans la figure 21.

```
Set_property INIT "32'h00000000" [get_cells cell_name]
```

Figure 20: Commande TCL pour modifier le vecteur d'initialisation d'une LUT.

Cette attaque demande une étape de rétro-ingénierie pour choisir les LUT critiques à cibler. Pour cela, différentes configurations de l'AXI Interconnect sont synthétisé pour choisir les LUT

critiques qui sont stables et qui ne changent ni de vecteur d'initialisation (la même fonction logique) ni de nom. Par la suite, les LUT qui affectent la valeur du signal BRESP sont sélectionnées. Enfin, deux LUT avec les noms ***s_axi_bresp[0]_INST_0*** et ***s_axi_bresp[1]_INST_0*** sont ciblées. Ces deux LUT sont initialisés par une fonction logique qui contrôle le signal BRESP avant de l'envoyer à l'interface maître. Pour ne pas signaler d'erreur à l'interface maître, la fonction logique des deux LUT est forcée à générer des sorties nulles (vecteur d'initialisation "***32'h00000000***") peu importe l'état de ses entrées.

Si l'attaquant ne connaît pas les noms des IP utilisées ou des LUT générées après synthèse, il peut utiliser les commandes TCL proposée par l'outil CAO Vivado [66] pour profiler le design et trouver les éléments à cibler. Ces commandes TCL peuvent être utilisées pour généraliser l'attaque et cibler un grand nombre de design.

3.3. Compromettre l'AXI Interconnect

Dans le design de la figure 13, l'AXI Interconnect joue un rôle important dans la sécurité du système. Comme contrôleur TrustZone, il compare l'état de sécurité des signaux de sécurité du bus AXI et l'état de sécurité de l'IP destinatrice, il sauvegarde les états statiques de sécurité de toutes les IP connectées et il envoie des messages d'erreur pour informer le système de toute violation de sécurité. Cette section présente des attaques qui peuvent cibler l'AXI Interconnect et donc le système complet.

3.3.1. Modification malicieuse du code du Crossbar

Le Crossbar (détaillé dans la section chap1-2.3.2) est le bloc central de l'AXI Interconnect. Il est chargé de vérifier l'état de sécurité de chaque transactions en comparant les signaux de sécurité AWPROT[1]/ARPROT[1] avec l'état de sécurité statique de l'IP matérielle ciblée. En cas de violation, il envoie une erreur à l'interface maître par le bus AXI. Pour cela, une attaque est présentée par modification malveillante du code Verilog du Crossbar dans cette section.

Avant de réaliser l'attaque, il faut effecteur une étape de rétro-ingénieur pour comprendre et analyser le code Verilog du Crossbar. Cette étape a révélé l'importance des deux lignes de code présentées dans la figure 22-a. Les deux lignes sont responsables de la vérification de sécurité, la comparaison entre le signal de sécurité AWPROT[1] (***S_APROT[P_NONSECURE_BIT]***, dans le code) et l'état de sécurité statique de l'IP ciblée (***target_secure***), et la génération d'erreur de sécurité (***any_error_i[1]***). Dans le cas de violation de sécurité la valeur du signal ***any_error_i[1]*** est égale à '1', sinon '0'. Le signal ***target_secure*** est le résultat du ''et'' logique bit à bit entre le

signal ***target_mi_hot*** qui est un vecteur contenant l'ID de l'IP ciblée, et le vecteur ***P_M_SECURE_MASK*** qui contient l'état de sécurité des interfaces maîtres connectées à l'AXI Interconnect.

```
assign target_secure = |(target_mi_hot & P_M_SECURE_MASK);
assign any_error_i[1] = target_secure && S_APROT[P_NONSECURE_BIT];
```

a. Code Verilog d'origine de l'opération de vérification

```
assign target_secure = |(target_mi_hot & P_M_SECURE_MASK);
assign any_error_i[1] = (target_mi_hot == 2'b01) ? 1'b0 : (target_secure && S_APROT[P_NONSECURE_BIT]);
```

b. Code Verilog modifié

Figure 21: Modification malveillante du code Verilog du Crossbar de l'AXI Interconnect

Plusieurs attaques sont possibles en modifiant malicieusement les deux lignes du code Verilog de la figure 22-a. Tous les bits du vecteur ***P_M_SECURE_MASK*** peuvent être validés (tous les interfaces maîtres sont non-sécurisées) pour créer une escalade de privilège, le monde non-sécurisé accède aux IP sécurisées, et le monde sécurisé a son fonctionnement normal due au fait que le monde sécurisé a accès à la totalité des ressources. Les bits du vecteur ***P_M_SECURE_MASK*** peuvent être mis à zéro (tous les interfaces maîtres sont sécurisées) pour créer un déni de service au monde non-sécurisé, tout accès depuis le monde non-sécurisé génère une erreur. La valeur du signal **any_error_i[1]** peut être fixée pour la vie du système, si la valeur fixée est à '0', le Crossbar ne remonte jamais d'erreur. Dans le cas contraire, le Crossbar envoie toujours un message d'erreur au système à l'interface maître.

La figure 22-b montre une modification qui vise le fonctionnement d'une IP sécurisée spécifique. Lorsque la transaction cible cette IP depuis une interface maître non-sécurisée, le Crossbar autorise la transaction au lieu de la rejeter et il envoie un message d'erreur. Lorsque les transactions ciblent le reste des IP, le système se comporte normalement. Le code est modifié par l'ajout d'une branche conditionnelle afin d'obtenir un accès privilège lorsque le bit du vecteur **target_mi_hot** lié à l'IP cible est valide et a un comportement normal pour les autres IP.

3.3.2. La modification malicieuse des configurations de sécurité de l'AXI Interconnect

Cette section présente une attaque qui utilise les commandes TCL pour attaquer l'AXI Interconnect. Cette attaque cible à modifier l'état de sécurité statique des IP connectées à l'AXI

Interconnect durant la phase de conception. Les commandes TCL peuvent être dissimulées dans un script d'automatisation de l'étape de synthèse ou dans l'étape de création de design.

```
Set_property CONFIG.M00_Secure false [get_bd_cells AXI_Interconnect]
```

Figure 22: Commande TCL malveillante pour changer l'état de sécurité de l'interface ''M00''

L'AXI Interconnect utilisé dans le design d'exemple de la figure 13 offre 16 emplacements d'interfaces maîtres à connecter aux IP (interface esclave). La figure 23 montre la modification de l'état de sécurité de l'interface maître numéro 0, l'interface maître ''M00''. Cette commande change l'état de l'IP connectée à l'interface maître ''M00'' en non-sécurisée. La variable ***P_M_SECURE_MASK*** présentée dans la section précédente est affectée car elle sauvegarde les états de sécurité statique des 16 interfaces dans le code Verilog du Crossbar. La modification de la figure 23 conduit à une escalade de privilège, elle permet à une interface maître non-sécurisée d'utiliser l'IP connectée à l'interface ''M00'' de l'AXI Interconnect.

3.3.3. L'insertion d'une FIFO malicieuse

En 2016, Fern et al. [67] ont présenté pour la première fois l'insertion d'une FIFO malicieuse dans l'AXI Interconnect pour fuiter des données sensibles. Comme dans [67], une FIFO malicieuse est utilisée pour espionner les transactions sécurisée entre l'interface maître et une IP sécurisée (interface esclave verte, figure 24). Supposons que la FIFO malicieuse est insérée dans le code RTL de l'AXI Interconnect durant la phase de conception. La taille de la FIFO malicieuse dépend de la quantité de données à divulguer, mais elle doit être aussi petite que possible pour assurer le succès de cette attaque et le non détection de la modification.

La figure 24 présente les modifications malicieuses apportées à l'AXI Interconnect pour l'insertion de la FIFO malicieuse. En générale, une FIFO a besoin de deux conditions d'activation, une pour la lecture de donnée et une autre pour l'écriture de donnée. Dans ce scénario d'attaque, le handshake du canal d'écriture connecté à l'interface esclave sécurisée est utilisé comme condition d'écriture et le handshake du canal d'écriture connecté à l'interface esclave non-sécurisée est utilisé comme condition de lecture. L'attaquant peut ajouter des signaux de contrôle pour indiquer le début et la fin de l'enregistrement ou pour indiquer le remplissage de la FIFO malicieuse.

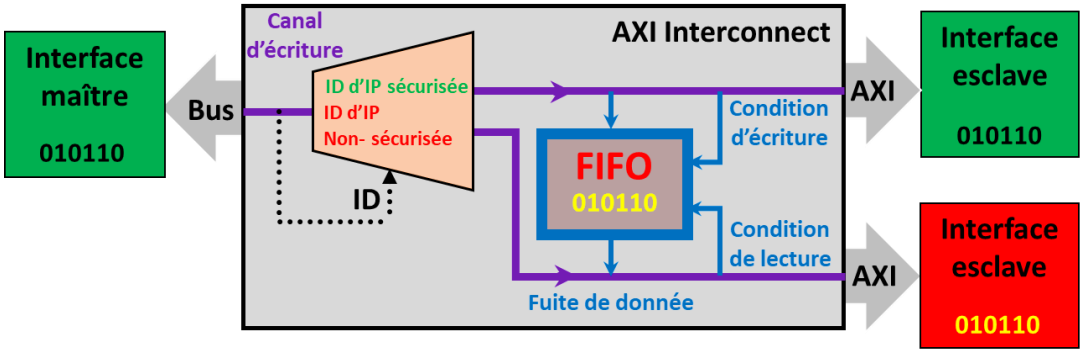

Figure 23: L'insertion d'une FIFO malveillante au niveau de l'AXI Interconnect pour espionner les données envoyées vers une IP sécurisée

Durant une transaction entre l'interface maître et l'interface esclave sécurisée, la FIFO malicieuse enregistre les données échangées. Par la suite, l'interface esclave non-sécurisée récupère les données enregistrées dans la FIFO durant une requête de lecture envoyée par l'interface maître (trame de donnée ''010110'' en jaune, figure 24).

4. Scénario complet d'attaque visant l'outil CAO Vivado de Xilinx

Les sections précédentes de ce deuxième chapitre ont déjà discuté la problématique des attaques basées sur l'utilisation des outils CAO qui ne sont pas de confiance sans présenter de scénario complet d'attaque. Cette section présente un scénario d'attaque complet visant l'outil CAO Vivado de Xilinx. Cette section démontre aussi l'importance de la sécurité des outils CAO utilisés pour la création des SoC hétérogènes de confiance.

4.1. Fichiers de descriptions matérielles dans un projet Vivado

La figure 25 présente un exemple de design créé par l'outil Vivado. Le design contient l'IP Zynq7 Processing System qui permet de configurer la partie processeur du Zynq-7000, l'AXI Interconnect, le Processeur System Reset et un AXI GPIO qui contrôle en sortie quatre Leds.

La figure 26 présente la description hiérarchique du design de la figure 25, cette description est générée dans l'onglet source de l'outil CAO Vivado en même temps que la création du design. La description hiérarchique présente la hiérarchie du design avec les noms (le nom est indiqué entre parenthèse devant le nom du bloc intégré dans le design) des fichiers contenant la description matérielle des IP intégrées dans le design. Ces fichiers sont copiés depuis la bibliothèque standard fournie par Xilinx dans le dossier du projet.

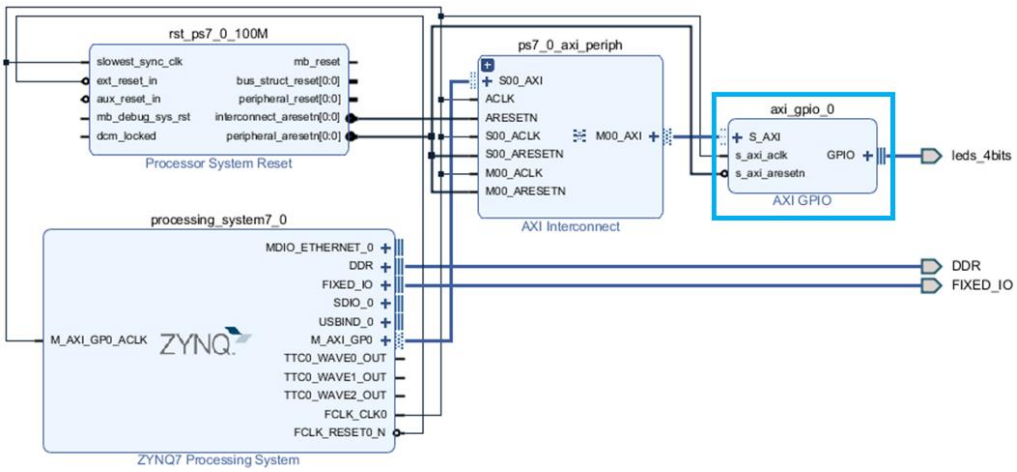

Figure 24: Design créé à l'aide de l'outil Vivado

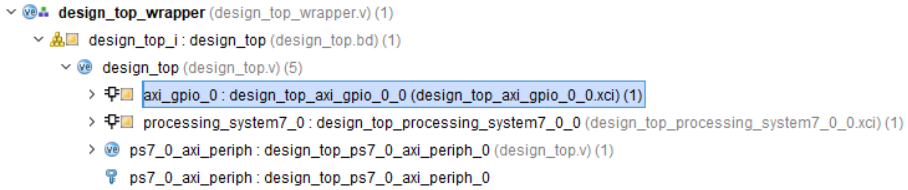

Figure 25: La description hiérarchique du design de la figure 25

Les fichiers contenant la description matérielle d'une IP Xilinx sont séparés en deux types. Des fichiers contenant la description matérielle en Vhdl ou Verilog de l'IP et qui sont accessible en lecture seule au concepteur dans l'interface de l'outil Vivado, comme le montre le rectangle bleu en haut à droite de la figure 27. Le deuxième type est des fichiers sécurisés par l'outil Vivado pour protéger la propriété intellectuelle. Ce deuxième type de fichiers contient seulement les entités des sous blocs constituant l'IP.

Figure 26: Fichier accessible en lecture seulement

4.2. Scénario d'attaque complet

Le scénario d'attaque présenté dans la figure 28 vise à modifier malicieusement le code source de l'IP GPIO intégrée dans le design de la figure 25. Comme indiqué dans la section précédente les fichiers de l'IP GPIO ciblé par l'attaque ne peuvent pas être modifiés par le

concepteur dans l'outil Vivado car ce dernier les ouvre seulement en lecture dans son interface. Par contre, ils sont modifiables en dehors de l'outil Vivado à l'aide d'un simple éditeur de texte. Dans ce scénario d'attaque un programme malicieux installé dans le système est utilisé pour modifier les fichiers de l'IP GPIO.

Figure 27: L'emplacement du fichier cible

L'attaque demande une étape de rétro-ingénierie pour comprendre la description matérielle de l'IP GPIO et trouver les lignes de code critique à modifier. L'attaque commence par rechercher l'emplacement du fichier à cibler, pour cela le programme malicieux utilise le fichier ***nom_du_projet.xpr*** qui se trouve dans la racine du dossier du projet Vivado. Ce fichier est un fichier XML qui contient toutes les informations concernant le projet, comme la configuration de l'environnement de travail, les différents fichiers utilisés dans le projet, etc. La figure 28 présente le bout du code XML qui présente la liste des différents IP utilisés dans ce projet et l'emplacement de leurs fichiers. Une fois le fichier cible trouvé, le programme malicieux peut éditer les lignes du code à modifier.

Pour le succès de ce scénario d'attaque, le programme malicieux doit exécuter sa routine avant l'étape de synthèse et à chaque modification du design car l'outil Vivado recopie à nouveau les fichiers des IP utilisées dans le design depuis la bibliothèque standard. La recopie écrase la version du fichier modifié par le programme malicieux.

5. Conclusion :

Ce chapitre présente les méthodes à suivre pour propager l'utilisation de la technologie ARM TrustZone dans la partie reconfigurable d'un SoC FPGA. Ce chapitre présente également des

attaques qui visent la sécurité du bus AXI dans un SoC complexe hétérogène embarquant la technologie ARM TrustZone. Les attaques démontrent l'importance de la sécurité des signaux du bus AXI et de ses composantes (le Crossbar par exemple). Le chapitre montre que les outils CAO et les IP tiers provenant des fournisseurs qui ne sont pas de confiance sont une menace de sécurité pour les systèmes.