SETR-E3 HW/SW CoDesign

Document d'intégration contrôleur d'interruptions

Louison Gouy et Mathis Briard
January 18, 2023

École Polytech de Nantes
Electronique et Technlogie Numérique

Enseignant référent : Sébastien LE NOURS

Contents

1 Introduction

1.1 Contextualisation

La formation ETN (Électronique et technologies numériques) option SETR (Systèmes embarqués temps-réel) offerte par l'école polytechnique de l'Université de Nantes propose d'aborder diverses branches de l'électronique, des systèmes temps-réel aux systèmes à microprocesseur en passant par la conception de SoC (système sur puce). Cet ensemble de domaines techniques nécessite des compétences en matière de méthodologie de conception. Ce rapport détaille la planification de l'intégration du contrôleur d'interruptions conçu dans le module E2. Il est question de mettre l'accent sur les compétences transverses et sur notre capacité à prendre du recul sur les notions acquises cette année. Il ne s'agit de faire le travail d'intégration, mais bien de le planifier, d'évaluer la charge de travail et anticiper les difficultés rencontrées. Le scénario suivant est donné et guide ce rapport.

Vous êtes le chef d'un projet dont le résultat serait un SOC opérationnel mis en œuvre au sein d'un FPGA. Ce SOC repose sur un processeur Zynq-PS. La plate-forme devra intégrer à terme l'IP que vous avez conçue dans le cadre du module E2. Vous devez présenter à votre responsable de service un dossier succinct mais convaincant du travail à mener afin d'effectuer l'intégration de votre IP au sein de la plate-forme. Il s'agit de définir l'effort nécessaire pour aboutir à ce résultat, sachant que la solution fonctionnelle de l'IP existe en VHDL synthétisable car vous l'avez conçue, testée et validée.

Le travail présenté, se positionne donc comme la suite logique du module E2. Il peut être intéressant de rappeler sur le cycle en V l'avancement du projet.

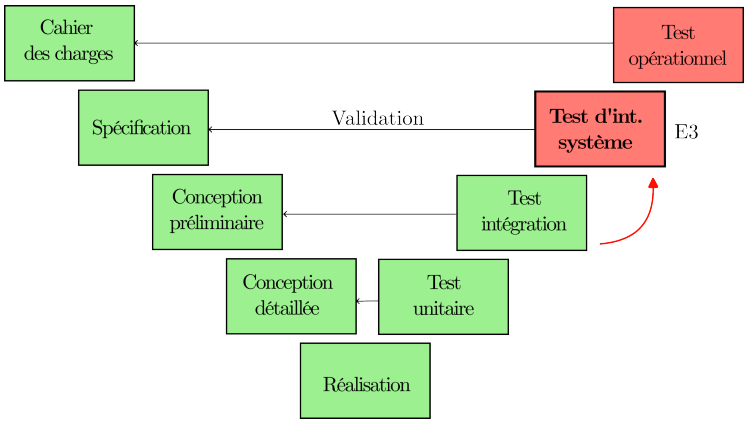

Figure 1: Positionnement de l'intégration sur le cycle en V

Sur ce cycle en V colorisé, représente en vert les étapes déjà réalisées et en rouge celle devant l'être pour finaliser le projet. Parmi les étapes manquantes, le scénario proposé n'évoque que l'intégration. Le test opérationnel sera donc laissé de côté. Le cycle en V donne une vue globale où on visualise bien l'étape précédente et le résultat souhaité.

2 Plateforme SoC

Cette partie présente le positionnement de l'IP sur la plateforme SoC. L'objectif est de mettre l'accent sur les ressources mises en jeu. Il n'est pour autant pas question d'aller jusqu'à lister le nombre de LUT ou bloc de BRAM. On choisit de conserver une vue globale. La carte FPGA retenue pour ce travail fait partie de la famille des Zynq7000 de Xilinx. Elle intègre deux cœurs de processeur ARM efficacement connectées à de la logique programmable tel qu'on la retrouve dans les FPGA. Cet assemblage permet de profiter à la fois de la flexibilité qu'apporte le logiciel tout en gardant la possibilité de paralléliser et customiser un circuit numérique. Ces deux propriétés sont particulièrement intéressantes et permettent d'implémenter des accélérateurs matériels pour le logiciel en profitant d'une communication efficace permise par l'interconnexion AMBA®.

Implémenter un contrôleur d'interruption sur la plateforme est toutefois un peu particulier. Cela permet effectivement d'obtenir un traitement des priorités avec une latence réduite et inférieure au cycle horloge. Mais, la plateforme Zynq-7000 intègre une fonction appelée Generic Interrupt Controller (GIC) qui couvre largement les fonctionnalités de notre IP et même bien d'avantage. Notre souhait d'intégrer un second contrôleur d'interruption, est a priori difficile à justifier dans un cadre industriel. Nous ferons l'hypothèse par la suite, pour des raisons pédagogiques, que notre IP a bien ça place et apporte un réel avantage. Sans se perdre dans les détails techniques, nous essayeront de confirmer que ces deux blocs peuvent cohabiter dans le même SoC sans entrer en conflit. La figure 2 illustre le positionnement du contrôleur d'interruption (notre IP) sur la plateforme SoC.

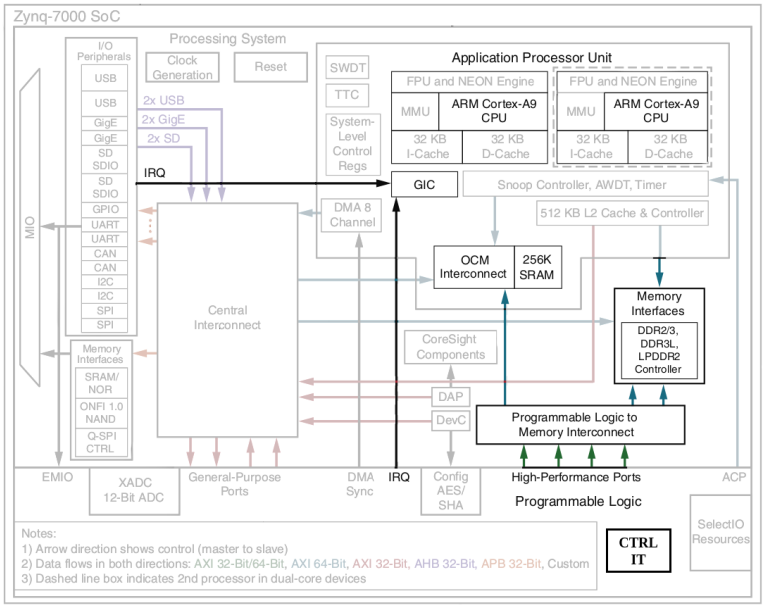

Figure 2: Positionnement de l'IP sur la plateforme SoC.

Elle se place naturellement en bas à droite dans la partie "programmable logic" (PL). Son implémentation repose bien sur un ensemble de ressources logiques composé de LUT, bascules, RAM, etc. La figure 2 tirée de la documentation Xilinx [3] a été modifiée afin de mettre en avant les éléments en lien avec notre IP. Elle n'a pas vocation à être parfaitement exhaustive. On retrouve parmi les fonctions clés le "Programmable Logic To Memory Interconnect". Il est composé de ports AXI hautes performances fournissant un accès depuis la PL à la DDR et à la mémoire de la puce[1]. Ses quatre ports AXI allant de la PL aux

[1]Dans la documentation Xilinx on retrouve le terme on-chip memory (OCM)

CPUs (Processing System PS) sont configurables en tant qu'interfaces 32 bits ou 64 bits. C'est par cette interface que les échanges entre notre IP et le processeur auront lieu.

La particularité du contrôleur d'interruption est d'avoir une seconde interface de communication avec les autres IPs. Cette dernière est formée de l'ensemble des signaux `nIT_xxx` indicateurs d'une requête d'interruption. Le terme "les autres IP" fait référence aux IPs "sur mesure" implémentées dans la PL et celle déjà présente dans l'"I/O peripherals" comme le SPI, l'UART, etc. Il y a donc deux sources bien distinctes. Nous pensons, en nous basant sur la documentation [3], qu'il est possible de rediriger toutes les sources d'interruptions vers notre IP. La figure 3 illustre le chemin des signaux `nIT_xxx` et `nIT_CPU` en se basant sur l'architecture du GIC.

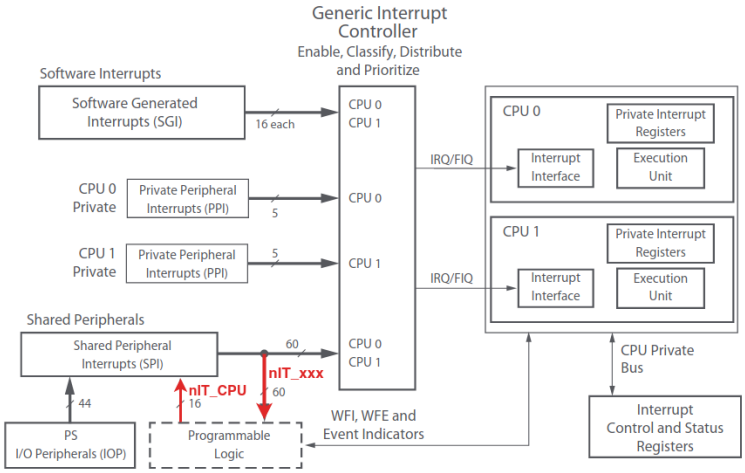

Figure 3: Chemin des sources d'interruptions d'après l'architecture du GIC.

La solution que nous proposant est de récupérer les requêtes des "I/O peripherals" via le bus marqué `nIT_xxx`. Les sources émises par les IP "sur mesure" peuvent être connectée directement à notre IP sans passer par le GIC. Le signal `nIT_CPU` peut lui être transmis via parmi les 16 sources provenant de la PL. Il faudra au préalable masquer dans le GIC les sources provenant des "I/O peripherals" sans quoi elles seraient traitées deux fois.

Le dernier élément mis en évidence par la figure 2 est naturellement l'unité processeur applicatif. L'aspect multicœur est complexe. Il est difficile de se projeter sur la gestion des interruptions par ces deux processeurs ARM A9. Cela dépend aussi de la stratégie de l'application. Nous ferrons l'hypothèse que les interruptions sont traitées de la même manière que sur une architecture monocœur. À la réception du signal `nIT_CPU`, l'adresse de branchement est lue et la routine est exécutée. Le cœur utilisé est indifférent de ce point de vue.

3 Outils de développement

3.1 Analyseur Logique Intégré

L'environnement de développement Xilinx met à disposition une IP appelée IPA pour analyseur logique intégré (Integrated Logic Analyzer). Cet outil peut être instancié dans le FPGA et permet de capturer les chronogrammes des signaux souhaités. Il est parfaitement adapté pour le debug de l'interface AXI mais peut être connecté à tout autre point d'intérêt. Une source d'horloge synchrone au design observé doit idéalement être connectée. [2] Les données enregistrées pourront directement être visibles dans l'interface "Harware manager" de Vivado. La figure 4 illustre un exemple d'utilisation de l'ILA en observation du bus AXI.

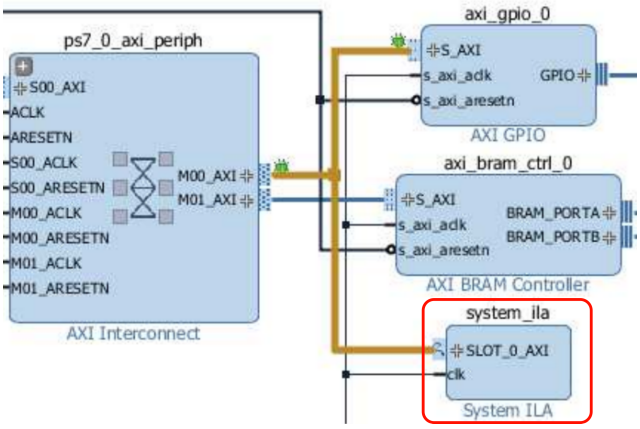

Figure 4: Positionnement de l'IP ILA sur un bus AXI

Les blocs logiques habituels sont utilisés pour implémenter la sonde d'échantillonnage et les triggers. Les valeurs des signaux échantillonnées sont ensuite sauvegardé dans la RAM. L'utilisation de l'ILA n'est possible qu'à condition d'avoir les ressources logiques nécessaires. Il est probablement possible de faire varier sa surface d'occupation en déduisant le nombre de sondes et la complexité des réglages lors de l'instanciation. N'ayant pas d'informations supplémentaires sur la taille du circuit à tester, nous feront l'hypothèse la plus optimiste pour laquelle aucun problème d'espace n'est rencontré.

3.2 Débugueur logiciel

Central Processing Unit (CPU)

Acronyms

CPU Central Processing Unit 6

[1]

References

[1] J.P. Calvez. *Spécification et conception des systèmes - une méthodologie*. 1991.

[2] Xilinx. Embedded processor hardware design. techreport UG585 (v1.13), April 2016.

[3] Xilinx. Zynq-7000 soc technical reference manual. techreport UG940 v2016.4, November 2021.