Proposition de thèse DGA

Architecture multi-cœur alliant tolérance aux fautes et prédictibilité

Mots-clés : architecture *many-core* hétérogène, tolérance aux fautes, prédictibilité, système à criticité mixte, consommation d'énergie.
Laboratoire : IRISA/INRIA - équipe CAIRN https://team.inria.fr/cairn
Directeur de thèse : Olivier Sentieys (olivier.sentieys@inria.fr)
Co-directeur de thèse : Angeliki Kritikakou (Angeliki.Kritikakou@inria.fr)
Tuteur DGA : Youri Helen (DGA MI/SDT/MAN/BSA, Rennes, youri.helen@intradef.gouv.fr)

Problématique générale

Les systèmes embarqués temps-réel critiques font face à une demande exponentielle en termes de **performance**, tout en ayant à respecter des **contraintes de temps réel strictes**. La diversité des applications s'exécutant sur ces systèmes aboutit à la notion de **système à criticité mixte** [1] où des fonctionnalités critiques s'exécutent en parallèle à d'autres moins ou non critiques. Pour les applications critiques, des garanties doivent être fournies en termes de **sûreté** et de **fiabilité**, ce qui implique d'une part une grande tolérance aux erreurs ou aux pannes et, d'autre part, leur exécution sous contrainte de temps stricte [2]. La notion d'erreurs et de pannes est liée à la problématique d'effets des radiations (d'origine natuelles ou artificielles) sur les composants électroniques, problématique qui non prise en compte peut mettre en péril le déroulement d'une mission (y compris au niveau du sol).

Les performances globales du système sont améliorées quand les ressources disponibles peuvent être utilisées par les applications à faible criticité. Les **applications embarquées critiques** sont présentes dans de nombreux domaines tels que les transports (avionique, automobile, etc.), la défense, le nucléaire ou encore le domaine fortement émergent du médical. En ce qui concerne la sûreté du système, comme les contraintes de temps réel strictes des applications critiques doivent être respectées, des outils d'analyse statique sont nécessaires pour obtenir des garanties sur leur **comportement temporel**. Il est important que ces outils estiment des bornes étroites afin d'éviter une sous utilisation des ressources et assurer le déterminisme du système. Cependant, ces bornes peuvent devenir très pessimistes à cause des difficultés à prédire le comportement du matériel et du logiciel.

Les **architectures multi-cœurs** *(multicore)*, ou foulti-cœurs *(manycore)*, offrent une puissance de calcul massive et une intégration des applications densifiée sur une même plateforme [3]. Remplacer les infrastructures existantes par des multi-cœurs permettra de réduire de façon très significative le coût global du système ainsi que sa consommation d'énergie et sa maintenance, tout en améliorant les fonctionnalités du système en lui offrant de plus grandes puissances de calcul. La fiabilité globale peut aussi être améliorée car moins de composants et de connecteurs sont utilisés pour le système.

Cependant, les architectures multi-cœurs augmentent de façon significative la complexité de l'architecture et donc la difficulté de prédire leur comportement, notamment à cause de certains mécanismes mis en œuvre pour augmenter leur performance. Cette difficulté à prédire le comportement d'un système est peu compatible de certaines des applications critiques visées (avionique), où le critère déterministe de l'architecture est obligatoire. Ceci est spécialement vrai pour les composants sur étagères (COTS - *Commercial-Off-The-Shelf*) disponibles actuellement. Cette difficulté de prédiction des performances vient notamment des dispositifs intra-cœurs, tels que les caches ou les prédicteurs de branchement, mais aussi plus récemment des aspects inter-cœurs, tels que les communications sur le réseau d'interconnexion sur puce, les mémoires cache partagées et les contrôleurs mémoire, puisque des accès concurrents aux ressources impliquent des variations des temps d'exécution très significatifs. Pour estimer ces bornes temporelles, soit des estimations WCET *(Worst Case Execution Time)* [4] pessimistes, soit des isolations temporelles et spatiales complètes entre les applications, peuvent être appliquées. L'isolation temporelle est obtenue si la durée de l'exécution d'une application dans une partition est indépendante des applications des autres partitions. L'isolation spatiale évite aux partitions d'accéder à la mémoire ou aux interfaces qui ne sont pas dans leur espace. Cependant, le résultat peut être une utilisation largement sous-optimale des ressources si le WCET est surestimé et si la réservation des ressources pour les applications critiques est sur-dimensionnée, ce qui laisse peu d'espace pour les applications moins critiques.

En ce qui concerne la fiabilité du système, surtout quand l'environnement est sévère (environnement radiatif naturel ou artificiel, orage, haute température, etc.) et avec la réduction des dimensions des transistors, des **fautes transitoires ou permanentes** apparaissent de plus en plus souvent, y compris au niveau du sol. Ainsi, la fiabilité du système devient très problématique, notamment quand celui-ci contrôle des fonctionnalités critiques, comme par exemple un défibrilateur cardiaque implanté ou le contrôle de la stabilité ou de la direction en avionique. Actuellement, pour offrir une fiabilité suffisante, des processeurs mono-cœurs durcis aux radiations sont utilisés mais ils offrent des puissances de calcul limitées. Pour ce qui concerne les multi-cœurs, des architectures tripliquées *(triplex)* sont utilisées pour améliorer la fiabilité, au détriment d'une augmentation des ressources utilisées, du coût et de la consommation d'énergie. Ainsi, des recherches sur des nouvelles techniques efficaces de tolérances aux pannes sont nécessaires autour de la détection, du confinement ou de la correction de fautes, notamment par rapport aux méthodes de triplication coûteuses, ainsi que sur l'impact de la tolérance aux pannes sur la prédictibilité de ces architectures.

L'objectif de cette thèse concerne l'étude et le développement de nouvelles techniques (notamment de gestion de l'architecture, d'estimation de performance ou de génération de code) et des architectures associées pour améliorer la sûreté et la fiabilité dans les plateformes multi-cœurs, ceci en explorant différents compromis entre la prédictibilité, la sûreté de fonctionnement, le déterminisme, les performances et la consommation du système.

Etat de l'art

En ce qui concerne la prédictibilité du système, les composants dynamiques d'un cœur ou ceux partagés entre eux sont soit considérés au pire cas (ce qui aboutit à une surestimation du WCET), soit ignorés (ce qui réduit les performances) [5]. Le principal challenge pour fournir une architecture prédictive est de disposer de blocs matériels qui supportent une certaine composabilité temporelle avec peu de pertes de performance [6]. Plusieurs approches existent pour améliorer la prédictibilité des architectures multi-cœurs par modification des composants qui peuvent créer de la contention dans leur fonctionnement. La plupart des travaux visent les mémoires partagées, les contrôleurs mémoire ou les bus de communications [7, 8, 9]. D'autres travaux visent à définir des architectures conçues dès le départ avec cette contrainte. FlexPRET [10] est un processeur pour les applications à

criticité mixte dans lequel un compromis entre l'isolation des tâches et l'utilisation des ressources est recherché. Si aucun processus critique n'est ordonnancé pendant un cycle, celui ci est utilisé pour des tâches moins critiques selon un mécanisme de tourniquet *(round-robin)*. D'autres architectures telles que MERASA [11], ACCROSS [12] ou le projet Européen T-CREST [13] suivent le même genre d'idée en étendant le concept aux multi-cœurs, notamment en retravaillant la hiérarchie mémoire ou le réseau d'interconnexion pour proposer un système plus prédictif. D'autres études proposent des techniques logicielles pour la gestion de l'architecture ou la gestion des tâches et la supervision dans un environnement parallèle [14] [15] [16] [17]. Toutes ces approches restent cependant basées sur des estimations pessimistes des performances lors de la compilation et passent difficilement à l'échelle quand le nombre de cœurs augmente.
Une possibilité pour améliorer les performances consiste à utiliser une approche hybride où des bornes restent estimées à la compilation mais où le pessimisme est revu pendant l'exécution en observant et en adaptant le comportement du système à partir de données issues du *monitoring* du système. L'analyse statique du code [18] peut être un levier pour améliorer les estimations de temps d'exécution au pire cas dans un contexte multi-cœurs [19], pour fournir des indications dans le code au contrôleur qui prendra les décisions lors de l'exécution à partir des moniteurs matériels et pour générer automatiquement des moniteurs logiciels. Dans [20, 21], nous avons par exemple proposé des techniques de suspension/redémarrage de tâches basées sur le monitoring de leur temps d'exécution. La décision de suspendre les tâches moins critiques est faite à partir du WCET restant à exécuter sur les tâches critiques en isolation.

En ce qui concerne la fiabilité du système, il est tout d'abord nécessaire d'identifier le type de fautes à considérer, leur effet sur le fonctionnement du système et les mécanismes de détection de fautes, de correction de fautes et de reprise *(recovery)* à une état connu de l'exécution d'une application. Pour la détection de fautes, de nombreuses approches ont été proposées [22] [23] [24] mais il reste de nombreuses questions ouvertes telles que le surcoût sur le contrôle, le nombre de points mémoire à surveiller ou le pourcentage de fautes à détecter dans les systèmes critiques, qui dépend de la criticité de la tâche exécutée, de la propagation des erreurs, de la partition mémoire entre les applications ou de la latence induite par une faute. Pour la correction des fautes, les mécanismes de reprise peuvent être soit du *retry*, soit de la correction ou du masquage de fautes [25]. Dans le premier cas, il s'agit de ré-exécuter une partie de l'application à partir d'un point de sauvegarde connu (ce qui implique une perte de performance et la nécessité de trouver un compromis entre le nombre de points de reprise et l'augmentation de la taille du code et de la mémoire), tandis que dans le second cas, l'exécution n'est peu ou pas perturbée par l'apparition de fautes transitoires (mais en général ceci vient avec un surcoût important en termes de surface et donc de coût de l'architecture). Il s'agit donc de trouver un bon compromis entre le surcoût induit par le masquage de fautes et la perte de performance due à la ré-exécution. Dans ce cas, l'impact sur la prédictibilité des performances, dépendant de la fréquence des fautes, doit être étudié.

Programme de la thèse

Plusieurs directions existent pour la conception des architectures multi-cœurs fiables et sûres. Dans cette thèse, nous nous baserons sur les conclusions de l'analyse faite dans [5] sur les contentions des ressources qui indiquent que les multi-cœurs COTS ne sont pas adaptés à l'exécution de tâches parallèles nécessitant des accès intensifs et simultanés aux mémoires. Nous nous focaliserons donc sur la **proposition** de **mécanismes de gestion** sur **une architecture multi-processeurs hétérogène** pour garantir une **meilleure sûreté et fiabilité du système**, tout en offrant de **meilleures performances** que les techniques classiques et architectures existantes.

Nous explorerons en particulier les extensions de ces systèmes pour inclure de la **tolérance aux fautes transitoires et aux fautes permanentes** et étudierons l'effet de l'introduction de ces techniques sur le comportement temporel (performance, prédictibilité) dans le contexte d'un **système à contraintes strictes ou à criticité mixte**. En particulier, des mécanismes de contrôle lors de l'exécution ***(run-time)*** seront développés pour garantir un niveau donné de sûreté et de fiabilité de la plateforme, notamment en explorant le compromis entre le gain en qualité de service, l'*overhead* des techniques de gestion des fautes et de garantie de la prédictibilité, une meilleure utilisation des ressources et le pourcentage de fautes détectées et corrigées. Ce compromis peut aboutir à la création de stratégies complexes mais très efficaces d'adaptation *run-time* pour décider quel mécanisme de détection et/ou de corrections des fautes sera utilisé lors de l'exécution. Pour cela, un mélange entre des techniques d'analyse statique du code *(off-line)*, des moniteurs matériels ou logicielles pour analyser les performances ou les fautes et différents algorithmes implémentés dans un contrôleur ayant pour rôle de prendre des décisions pendant l'exécution *(on-line)*, devra être privilégié. La sûreté de fonctionnement du contrôleur lui même devra aussi être étudiée.

Pour implementer les strategies proposées, nous étendrons des architectures multi-cœurs existantes. Notre approche se basera par exemple sur l'utilisation du processeur RISC-V, qui est une architecture open-source disponible sous forme de bloc IP (simulation, synthèse, compilateur) offrant la possibilité d'ajouter des accelerateurs matériels, des instructions specifiques et des mécanismes d'interconnexions permettant de concevoir rapidement une architecture multi-cœurs [26]. Une autre piste est l'utilisation des architectures Zynq de chez Xilinx intégrant un système dual-core (processeurs ARM® Cortex™-A9) à de la logique programmable dans laquelle des accélérateurs matériels spécifiques peuvent être synthétisés et facilement connectés à l'architecture multi-cœurs. Leur évolution vers les plateformes UltraScale+ intègre jusque 6 cœurs de processeurs [27]. Quelle que soit l'architecture cible choisie, elle permettra des développements rapide de prototypes en vue d'évaluer les performances des techniques proposées. Les IP disponibles permettent d'assurer qu'il n'y a de pas de risque de conception matérielle identifié.

Au delà de la proposition de nouvelles techniques de gestion des effets singuliers (fautes temporaires ou permanentes), l'objectif de ces travaux est de fournir rapidement un modèle de simulation et un modèle synthétisable en vue d'une implémentationFPGA. Ces modèles intégreront les strategies de gestion de fautes développées, sur lesquels nous pourrons exécuter des W n /Cs1 currons

International Conference on Embedded Computer Systems : Architectures, Modeling and Simulation (IC-SAMOS), (Samos, Greece), July 2012.

[4] M. Gatti, "Development and certification of avionics platforms on multi-core processors," in *Tutorial Mixed-Criticality Systems : Design and Certification Challenges, ESWeek*, (Montreal, Canada), 2013.

[5] S. Vasudevan, A. Easwaran, and R. Klyne, "Profiling COTS multi-cores to quantify shared resource contention," in *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2015.

[6] S. Hahn, J. Reineke, and R. Wilhelm, "Towards compositionality in execution time analysis – definition and challenges," in *Proc. 6th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS)*, December 2013.

[7] S. Wasly and R. Pellizzoni, "A dynamic scratchpad memory unit for predictable real-time embedded systems," in *25th Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 183–192, July 2013.

[8] H. Kim, D. de Niz, B. Andersson, M. Klein, O. Mutlu, and R. Rajkumar, "Bounding memory interference delay in cots-based multi-core systems," in *IEEE 20th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 145–154, April 2014.

[9] M. Paolieri, E. Quiñones, F. J. Cazorla, G. Bernat, and M. Valero, "Hardware support for wcet analysis of hard real-time multicore systems," in *Proc. 36th Annual International Symposium on Computer Architecture (ISCA)*, pp. 57–68, ACM, 2009.

[10] M. Zimmer, D. Broman, C. Shaver, and E. A. Lee, "Flexpret : A processor platform for mixed-criticality systems," Tech. Rep. UCB/EECS-2013-172, EECS Department, University of California, Berkeley, Oct 2013.

[11] M. Paolieri, J. Mische, S. Metzlaff, M. Gerdes, E. Quiñones, S. Uhrig, T. Ungerer, and F. J. Cazorla, "A hard real-time capable multi-core smt processor," *ACM Trans. Embed. Comput. Syst.*, vol. 12, pp. 79 :1–79 :26, Apr. 2013.

[12] C. Salloum, M. Elshuber, O. Hoftberger, H. Isakovic, and A. Wasicek, "The across mpsoc - a new generation of multi-core processors designed for safety-critical embedded systems.," in *Euromicro Conference on Digital System Design (DSD)*, pp. 105–113, 2012.

[13] P. Puschner, D. Prokesch, B. Huber, J. Knoop, S. Hepp, and G. Gebhard, "The T-CREST approach of compiler and WCET-analysis integration," in *IEEE 16th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, pp. 1–8, June 2013.

[14] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha, "Memguard : Memory bandwidth reservation system for efficient performance isolation in multi-core platforms," in *Proc. IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 55–64, 2013.

[15] M. Masmano, I. Ripoll, A. Crespo, and J. Metge, "Xtratum : a hypervisor for safety critical embedded systems," in *Eleventh Real-Time Linux Workshop*, (Dresden, Germany), September 2009.

[16] J. Nowotsch, M. Paulitsch, D. Bühler, H. Theiling, S. Wegener, and M. Schmidt, "Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement," Tech. Rep. 2013-10, University of Augsburg, Germany, 2013.

[17] J. Nowotsch and M. Paulitsch, "Quality of service capabilities for hard real-time applications on multi-core processors," in *Proc. 21st International Conference on Real-Time Networks and Systems (RTNS)*, pp. 151–160, ACM, 2013.

[18] K. N. Parashar, D. Menard, and O. Sentieys, "Accelerated Performance Evaluation of Fixed-Point Systems With Un-Smooth Operations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, pp. 599–612, Apr. 2014.

[19] T. Stripf, O. Oey, T. Bruckschloegla, J. Becker, G. Rauwerda, K. Sunesen, G. Goulas, P. Alefragisc, N. Voros, S., S. Derrien, O. Sentieys, N. Kavvadias, G. Dimitroulakos, K. Masselos, D. Kritharidis, N. Mitas, and T. Perschke, "Compiling scilab to high performance embedded multicore systems," *Microprocessors and Microsystems*, vol. 37, pp. 1033–1049, Nov. 2013.

[20] A. Kritikakou, C. Pagetti, O. Baldellon, M. Roy, and C. Rochange, "Run-time control to increase task parallelism in mixed-critical systems," in *Proc. 26th Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 119–128, 2014.

[21] A. Kritikakou, C. Pagetti, C. Rochange, M. Roy, M. Faugère, S. Girbal, and D. G. Pérez, "Distributed Run-time WCET Controller for Concurrent Critical Tasks in Mixed-critical Systems," in *Proc. 22nd International Conference on Real-Time Networks and Systems (RTNS)*, pp. 139–148, ACM, 2014.

[22] C. Angeli and A. Chatzinikolaou, "On-line fault detection techniques for technical systems : A survey," *International Journal of Computer Science & Applications*, vol. 1, no. 1, pp. 12–30, 2004.

[23] S. M. A. H. Jafri, J. Piestrak, Stanislaw, O. Sentieys, and S. Pillement, “Design of the coarse-grained reconfigurable architecture DART with on-line error detection,” *Microprocessors and Microsystems*, vol. 38, pp. 124–136, Mar. 2014.
[24] S. M. A. H. Jafri, S. J. Piestrak, O. Sentieys, and S. Pillement, “Design of a fault-tolerant coarse-grained reconfigurable architecture : A case study,” in *Proc. of the 11th IEEE International Symposium on Quality Electronic Design (ISQED 2010)*, (San Diego, CA, USA), p. 6 pages, IEEE, Mar. 2010.
[25] M. Pignol, “DMT and DT2 : two fault-tolerant architectures developed by CNES for COTS-based spacecraft supercomputers,” in *12th IEEE International On-Line Testing Symposium (IOLTS)*, pp. 1–10, 2006.
[26] The RISC-V Instruction Set Architecture, http://riscv.org, 2016.
[27] Xilinx All Programmable SoC, http://www.xilinx.com/products/silicon-devices/soc.html, 2016.