Une approche pour la fiabilité des plateformes Multiprocesseurs

NICOLAS HEBERT[1, 2], LIONEL TORRES[1], GILLES SASSATELLI[1], PASCAL BENOIT[1]

[1] Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
Université Montpellier 2, LIRMM UMR CNRS 5506, 161 rue Ada, 34392 Montpellier Cedex 5, France
[2] STMicroelectronics
850 rue Jean Monnet, 38920 Crolles, France

[1,2]Nicolas.Hebert@st.com, [1]Lionel.Torres@lirmm.fr, [1]Gilles.Sassatelli@lirmm.fr,[1]Pascal.Benoit@lirmm.fr

Résumé - Détecter et réagir à l'occurrence de pannes de divers types sur des systèmes multiprocesseurs sur silicium (MP-SoC) devient un challenge pour les années à venir. Nous apportons deux solutions architecturales de fiabilisation : un « chien de garde » sur une architecture à flots de données et une technique de diagnostic et de reconfiguration à base de tâches logicielles sur une architecture multiprocesseur. Ces deux méthodes introduisent un surcoût architectural et logiciel, dimensionné notamment pour le traitement du signal et des images.

Abstract – Detecting and reacting to the various failure occurrences on multiprocessor system on chips (MP-SoC) will be challenging in the next few years. We propose two reliable architectural solutions: a watchdog for streaming architecture and a diagnostic and reconfiguration technique based on software tasks. These two methods incur a cost overhead, that can be tuned for signal and image processing algorithms.

1 Introduction

L'étude simultanée des aspects algorithmiques et architecturaux pousse à une implantation optimisée tout en réduisant le temps de développement. L'Adéquation Algorithme Architecture (A3) a permis les développements opérationnels et matériels de solutions auto-adaptables. Plusieurs plateformes matérielles reconfigurables sont proposées aujourd'hui. Parmi celle-ci l'architecture multiprocesseur HS-Scale [**1**] est auto-adaptable, la programmation de l'ordonnancement des tâches est statique mais la plateforme alloue dynamiquement des ressources aux tâches. Ceci permet de supporter des spécifications récursives avec une implémentation qui reste efficace.

L'utilisation de technologies de gravure extrêmement fines permet d'intégrer sur silicium des systèmes de plus en plus complexes. Cependant les phénomènes de variabilité technologique dus aux limitations lithographiques et physiques ont sensiblement augmentées ces dernières années. Avec une gravure en 16 nanomètres il ne reste que dix atomes dopant dans le canal. Un atome en moins engendre ainsi une variation du dopant de 10%. La couche d'oxyde est devenue si mince (moins de 1,5 nm sur les technologies actuelles, le rayon atomique du silicium étant de 110 pm) qu'il est possible de faire passer des électrons par effet tunnel quantique du substrat de silicium à l'électrode de la grille. Les phénomènes de vieillissements et d'usures s'accentuent, notamment l'usure de l'oxyde de grille, des interconnexions, les phénomènes d'injection de porteurs chaud deviennent marqués[**2**]. Les phénomènes fautes transitoires sont principalement dues à la radioactivité naturelle des boitiers qui émettent des particules alpha chargées et qui viennent perturber la distribution des électrons au niveau du substrat et pour une autre partie du rayonnement cosmique composé à 95 % de neutrons énergétiques. Ces neutrons entrent en contact avec les électrons du substrat ce qui augmente la charge électrique de la zone d'impact. Les fautes transitoires sont en nette augmentation; en 16 nm elles sont estimées à cent fois plus fréquente qu'en 180 nm.

Cet article s'attache à définir une méthodologie globale afin d'assurer aux plateformes un niveau de sûreté et de sécurité en fonction du domaine d'application ciblé. Les méthodes de protection portent simultanément sur le matériel et la couche d'interface (Système d'exploitation ou OS - Operating System). L'approche à haut niveau d'abstraction a été privilégiée, ses caractéristiques la rendant intéressante du point de vue de la généricité, configurabilité et adaptabilité des interventions.

Nous présentons rapidement les généralités sur les architectures des MP-SoC. Nous établissons une énumération des types de fautes et de la modélisation de celles-ci. Nous abordons ensuite un état de l'art des protections existantes. Au terme de cette étude nous proposons deux protections que nous implémentons sur l'architecture cible.

2 Généralités sur les Architectures des MP-SoC

De nombreux critères peuvent être utilisés afin de caractériser une architecture multiprocesseur, parmi lesquels :

- La nature (programmable, reconfigurable, dédiée) et la granularité des unités de calcul, ainsi que leur structure globale qui peut être homogène ou hétérogène.
- L'architecture mémoire (partagée ou distribuée) et l'accès à la mémoire (symétrique ou asymétrique)

- Le système de communication utilisé pour connecter les unités de calcul
- Le niveau de sûreté de fonctionnement

Un exemple d'architecture est donné en section 6 de cet article.

Les architectures MP-SoC régulières sont potentiellement moins complexes à concevoir, elles facilitent le passage à l'échelle « scalability », et disposent d'un temps de mise sur le marché accéléré. De plus, elles sont surtout potentiellement plus fiables grâce aux symétries qui créent une redondance naturelle.

3 Fautes et Modèles

Afin de définir le processus qui mène du défaut à la défaillance et de définir ce qu'est une faute, nous avons choisi d'utiliser la terminologie formalisée par Laprie[**3**]. Un défaut peut être modélisé sur la base de sa conséquence au niveau de la structure du circuit. Cette modélisation correspond à une faute dans le système. En réalisant une opération sur l'operateur défectueux la faute peut être activée. Dans ce cas elle fait dévier le système de l'état de fonctionnement normal, nous avons généré une erreur. Cette erreur peut ensuite contaminer les éléments sains du système en injectant des valeurs erronées. Si la contamination fait dévier fonctionnellement l'application, nous avons une défaillance.

Comme pour tous les phénomènes physiques, il est nécessaire de définir un modèle de fautes afin de simuler fidèlement leur impact sur le système. Nous nous basons sur la formalisation de Anghel et de son équipe[**4**].

Nous avons choisi de prendre en compte les fautes dues à un défaut physique (Stuck-at 0/1) ainsi que les fautes transitoires de type inversion de bit (SEU, Single Event Upset), de signal (SET, Single Event Transient) et les effets vectorisés (MBU, Multiple Bit Upset). Nous modélisons ces fautes selon un modèle à deux niveaux : au niveau des fautes transitoires de type « multiple changements de bits » (TMBU, Transiant Multiple Bit Upset) avec le cas particulier d'un changement de bit simple, puis un second niveau de type changement de bit irrévocable (IMBU, Irrevocable Multiple Bit Upset), qui nous permettra de modéliser le vieillissement ou les fautes matérielles de conception en fonction de leurs occurrences en début de simulation ou pendant le fonctionnement.

4 Protection (Matériel, OS, Application)

Une application est considérée comme critique quand au moins l'un de ces six attributs est requis : la fiabilité, la sécurité, la confidentialité, la disponibilité, l'intégrité ou la « réparabilité ». Dans chacun de ces domaines il est possible d'appliquer des protections à différent niveau de la conception. Nous interviendrons au niveau de l'architecture système afin d'opérer une protection du MP-SoC au niveau de son intégrité, sa « réparabilité » et sa disponibilité en présence de fautes.

Les differentes techniques de protection peuvent être : la redondance d'information, le durcissement de la technologie, les points de reprise sans perte d'information, …

Historiquement c'est avant tout la technologie qui doit fournir des portes logiques fiables. Quand la technologie ne peut pas être durcie suffisamment sans impacter le rendement de celle-ci (utilisation de cellule standard). La principale technique de protection devient alors la redondance.

Elle peut s'effectuer sur le matériel, citons par exemple : codes correcteurs d'erreur (ECC, Error-correcting code), parité, duplication, triplication ou réplication M-N modulaire [**5**]. La redondance s'applique aussi au niveau logiciel, notamment : ECC, programme en n-version [**6**]. La redondance est un outil puissant mais coûteux. Elle permet d'intervenir sur tous les domaines de sûreté de fonctionnement et à différents niveaux d'intégration. Des techniques moins coûteuses permettent d'assurer la stabilité et la disponibilité, comme par exemple : processeur avec chien de garde [**7**], point de reprise (C&R) [**8**]. Il est aussi important de disposer d'outils de reconfiguration, tel que : isolation configurable, migration de tâche, agent, compensation de processus.

5 Système MP-SoC HS-Scale

Nous avons fait le choix d'utiliser la plateforme HS-Scale développée par le laboratoire du LIRMM. Cette architecture est une architecture multiprocesseur homogène, symétrique, de grain moyen, à mémoire distribuée. HS-Scale est reconfigurable et adopte une architecture et un modèle de programmation de type tâches communicantes. Comme exposé Figure 1, elle est prévue pour intégrer un SoC comme une brique de traitement reconfigurable.

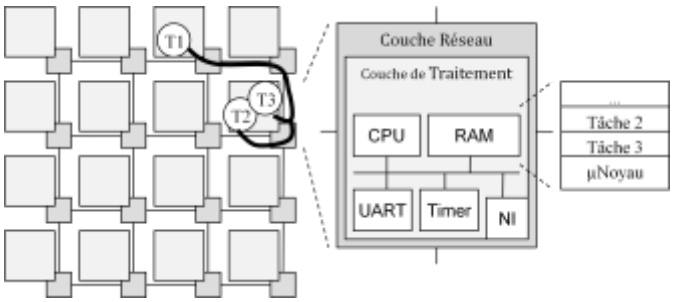

Figure 1: HS-Scale, brique de traitement auto-adaptable

La couche réseau des unités de traitement (NPU, Network Processing Unit) est interconnectée par le réseau sur puce (NoC, Network on Chip) HERMES[**9**] qui possède une topologie d'interconnexion en grille. Ce NoC Asynchrone à commutation de paquets permet d'isoler fonctionnellement les NPU. Chaque NPU se compose d'un routeur HERMES et d'une Unité de traitement (PE, Processing Element). L'interface entre le PE et la NPU s'effectue au travers de deux FIFO (NI, Network Interface). Le PE se compose d'une mémoire RAM interne configurable, d'un processeur RISC-32 bits (processeur Plasma [**10**]) qui a un jeu d'instruction équivalent au MIPS I™, d'une interface de communication de type UART, d'un Timer, d'une

interface vers l'extérieur (I/O), d'un contrôleur d'interruption, d'un contrôleur mémoire et d'un bloc contrôleur de fréquence. HS-Scale est actuellement implanté sur un réseau de 16 FPGA Spartan 3 et forme ainsi un réseau de 16 NPU.

Chaque NPU possède un micro système d'exploitation (OS, Operating System). Les applications sont programmées sous forme de réseau de processus de Kahn. Les tâches peuvent être hébergées sur différents NPU ou bien partager les ressources de traitement d'un unique NPU (exécution en temps partagé).

6 Protections proposées et implémentation

Notre intervention se limite aux couches des niveaux microarchitecture et architecture globale. Les flux de données à traiter ne sont pas critiques mais la plateforme doit cependant être fiable, disponible et intègre même en présence de détériorations ou d'interférences physiques.

Nous apportons deux solutions : un chien de garde sur une architecture à flot de données et une technique de diagnostic et reconfiguration à base de tâches logicielles.

6.1 Chien de garde sur architecture à flots de données.

Cette solution doit permettre de faire remonter vers les éléments sains des problèmes de disponibilité et d'intégrité dus à des fautes catalectiques, de déviation ou transitoires. Le principe consiste à envoyer un paquet de type chien de garde (WD, Watch Dog) régulièrement entre deux tâches d'une même application dans la direction opposée au flux de données. Ce paquet est envoyé régulièrement. Il contient un minimum d'informations (tâche source, tâche destination, identifiant du dernier bloc de données traitées), dans une première phase d'étude ce paquet contient uniquement l'information : « je suis en vie ».

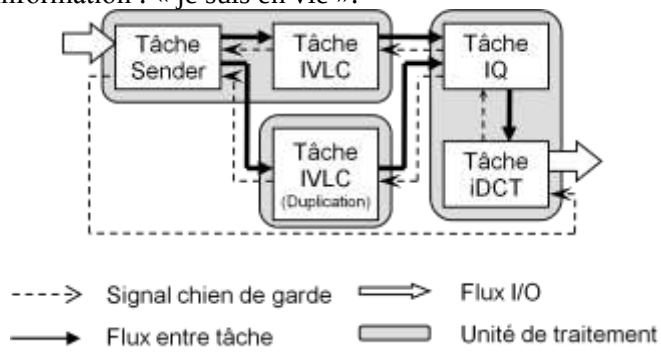

Figure 2 : Chien de garde sur HS-Scale, application MJPEG

Le WD, n'est pas un outil pour diagnostiquer une faute mais il permet de faire remonter une information d'occurrence d'erreur vers le processeur amont. Le diagnostic des fautes doit être effectué par d'autres processus qui sortent du périmètre de notre étude.

Quand une tâche ne reçoit pas son paquet de WD en un temps imparti, elle considère alors que la tâche en aval n'est plus vivante. Elle stoppe alors son flux de données vers cette tâche et demande la création de cette tâche aval à un processeur voisin ainsi que la création de tâches de diagnostic. La tâche peut alors reprendre son envoi de données au niveau d'un point de reprise aval (Rollforward), vers la nouvelle tâche créée sur un NPU voisin.

La Figure 2, montre un exemple d'implantation sur l'application MJPEG composée de 3 tâches principales. Les signaux de WD sont générés dans le sens opposé au flux de données. En cas de panne, au moins une tâche doit survivre pour permettre la recréation de l'application complète. Il faut donc répartir les applications sur au moins deux unités de traitement physique différentes.

6.2 Technique de diagnostic et de reconfiguration à base de tâches logicielles.

Après la détection de défaillances, il faut effectuer un diagnostic afin de déterminer l'origine de la faute et de l'isoler. Pour cela trois tâches de diagnostic (DT, Diagnostic Task) semblables vont être nécessaires. Nous illustrons sur la Figure 3 les différentes phases du processus qui amènent au diagnostic à trois taches logicielles : l'isolation, la reconfiguration et le diagnostic.

Phase d'isolation : suite à un défaut la tâche T2' n'émet plus son signal de chien de garde vers T1. T1 doit alors isoler cette tâche de l'application (T1, T2, T2', T3, T4). T1 stoppe tous ses flux vers la tâche T2'. T3 est avertie par T1 que T2' est défectueuse et qu'une procédure d'isolation est en cours. Cela permet à T3 de se préparer à se resynchroniser avec ses différentes entrées afin de préparer un point de reprise aval. T2 est maintenant isolé du point de vue de l'application.

Phase de reconfiguration : T1 ordonne à un OS présent sur un NPU voisin de créer une nouvelle tâche T2'. Le nouvel OS rapatrie le code de la tâche depuis l'extérieur de HS-Scale en passant par l'antémémoire des tâches courantes en direction de sa mémoire de tâche. Il lui faut ensuite ouvrir les ports de communication avec T1 et avec T3. T2' peut alors passer en phase de réarmement afin de repartir sur un point de reprise commun à T1 et T3. L'application est de nouveau fonctionnelle.

Phase de diagnostic : Il faut procéder à un diagnostic afin de déterminer l'origine de la panne. Nous procédons en trois étapes : test de la communication avec le NPU, test du processeur et des éléments du NPU puis mise à jour des tables qui régissent l'état de la plateforme.

Afin de tester les composants de communication du NPU nous utilisons une approche de test mutuel logiciel à trois NPU, nous reprenons des travaux de Piotr Zając et Jacques Henri Collet basés sur le « Self-diagnosis via mutual testing » [**11**]. Deux NPU sont considérés comme sain et le troisième est le NPU isolé. Les trois NPU testent leurs I/O en effectuant un test mutuel logiciel orienté communication. Un NPU sain teste, simultanément, d'une part le deuxième NPU sain et d'autre part le NPU isolé. La comparaison des résultats est effectuée et en cas de différence un diagnostic est établi et renseigne les tables (redondantes) du système.

Si l'erreur ne provient pas de la communication entre NPU nous effectuons un autotest logiciel (SBST, Software-Based Self-Testing) sur le NPU fautif. Pour cela nous nous basons sur les travaux effectués par Nektarios Kranitis et son équipe [**12**] qui concernent des SBST structurels à faible coût. Ils ont appliqué leurs travaux d'expérimentations sur le processeur Plasma avec un taux de couverture supérieur à 95 % (faute de type « Stuck-at ») en seulement 5707 cycles horloge (dont 74 % des cycles utilisés pour tester le multiplieur et le diviseur). Le résultat du test est envoyé à un NPU sain, afin qu'il puisse renseigner les tables (redondantes) du système sur l'état du NPU isolé.

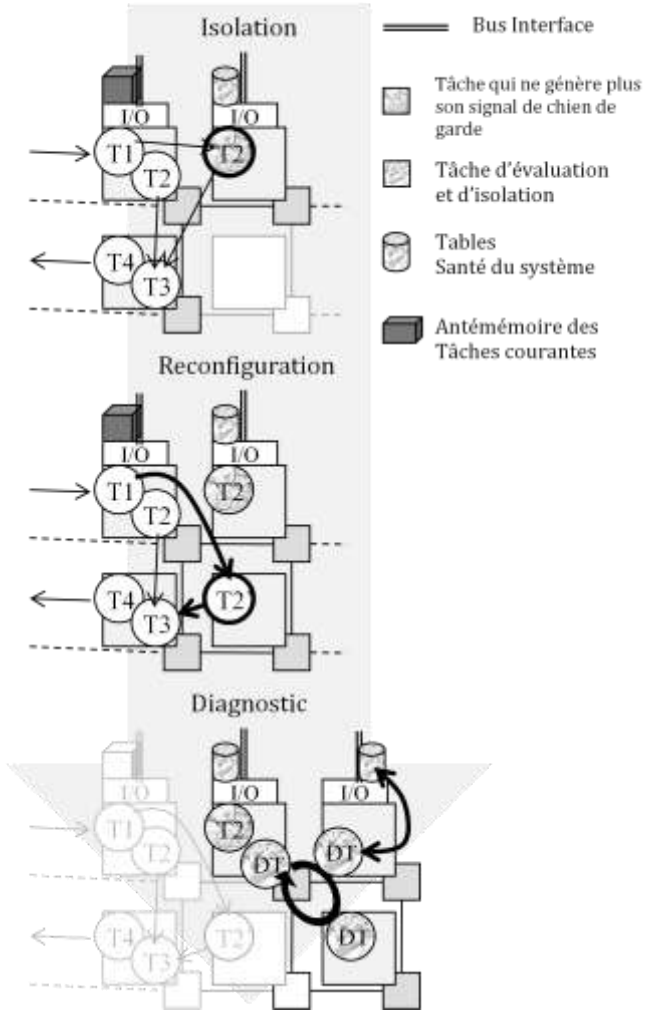

Figure 3 : Tâches d'Inspection, d'Isolation et de Reconfiguration (DT)

Il est important d'isoler les éléments fautifs avec un grain adapté [**13**]. Si par exemple le multiplieur matériel d'un PE est considéré hors de fonctionnement, nous pouvons faire fonctionner la plupart des tâches qui n'utilisent pas cet élément. En effet à travers de cet exemple (multiplieur) nous montrons l'avantage de la technique d'isolation partielle. Isoler partiellement un NPU a de nombreux avantages. Quand une tâche est créée, migrée ou dupliquée, l'OS source inspecte les tables du système afin de trouver la destination la plus adaptée en fonction des propriétés de la tâche et de l'état du système. Cette table est mise à jour par les tests réalisés sur un équipement de test automatique externe après fabrication, puis par les trois tâches de diagnostic.

Les tâches de diagnostic peuvent aussi être exécutées sur les NPU qui ont un taux d'utilisation important donc un état d'usure plus avancé afin de prévenir les fautes de vieillissement.

7 Conclusion

Les récents efforts menés dans le domaine des architectures multiprocesseurs montrent l'importance d'amener un environnement robuste vis-à-vis des fautes matérielles. La prise en compte au plus tôt d'un environnement matériel/logiciel est donc nécessaire. A ce jour nous avons proposé une première méthodologie qui doit être validée sur la plateforme HS Scale. Nous ciblons spécifiquement les algorithmes de type flots de données, algorithmes très présents dans le domaine du Traitement du Signal et des Images. Cette étape est donc nécessaire pour améliorer les techniques A3 actuelles.

8 Références

[1] N. Saint-Jean, G. Sassatelli, P. Benoit, L. Torres, and M. Robert, "HS-Scale: a Hardware-Software Scalable MP-SOC Architecture for embedded Systems," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2007, pp. 21-28.

[2] V. Narayanan and Y. Xie, "Reliability Concerns in Embedded System Designs," pp. 118-120, Jan. 2006.

[3] J. C. Laprie, "Sûrté de Fonctionnement et Tolérance aux Fautes : Concepts de Base," LAAS Rapport 88.287, 1988.

[4] L. Anghel, R. Leveugle, and P. Vanhauwaert, "Evaluation of SET and SEU Effects at Multiple Abstraction Levels," in *On-Line Testing Symposium (IOLTS)*, Saint Raphael, France, 2005, pp. 309-312.

[5] M. Portolan, "Conception d'un Système Embarqué Sûr et Sécurisé," TIMA Thèse, 2006.

[6] A. A. Avizienis and M. R. Lyu, *Software Fault Tolerance, Chap 2 : The Methodology of N-Version Programming*, Lyu, Ed. New York, USA: John Wiley & Sons Ltd, 1995.

[7] F. Vargas, L. Piccoli, J. Benfica, A. A. de Alecrim, and M. Moraes, "Time-Sensitive Control-Flow Checking for Multitask Operating System-Based SoCs," in *On-Line Testing Symposium. 13th IEEE International (IOLTS 07)*, Crete, 2007, pp. 93-100.

[8] Y. Tamir and M. Tremblay, "High-performance fault-tolerant VLSI systems using micro rollback," *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 548-554, 1990.

[9] F. Gehm Moraes, N. Laert Vilar Calazans, A. Vieira de Mello, L. Heleno Möller, and L. Copello Ost, "HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip," FACULDADE DE INFORMÁTICA PUCRS, Brazil, Technical Report Series 034, 2003.

[10] S. Rhoads. (2001, Sep.) Plasma. [Online]. http://plasmacpu.no-ip.org:8080/

[11] P. Zajac and J. H. Collet, "Production Yield and Self-Configuration in the Future Massively Defective Nanochips," in *Defect and Fault-Tolerance in VLSI Systems (DFT)*, Rome, Sep. 2007, pp. 197-205.

[12] N. Kranitis and D. Gizopoulos, "Software-Based Self-Testing of Embedded Processors," *IEEE Transaction on Computers*, vol. 54, no. 4, pp. 461-475, Apr. 2005.

[13] N. Aggarwal, P. Ranganathan, N. P. Jouppi, and J. E. Smith, "Configurable isolation: building high availability systems with commodity multi-core processors," *ACM SIGARCH Computer Architecture News*, vol. 2, no. 35, pp. 470-481, May 2007.