Méthode de conception d'architectures tolérantes aux fautes transitoires en milieu spatial : application à une chaîne de compression d'images

Thierry VALENTIN[1], Olivier INGREMEAU[1], Catherine LAMBERT-NEBOUT[2], Eric MARTIN[1]

[1]LESTER, Université de Bretagne Sud : rue Saint Maudé, 56100 Lorient, France
[2]CNES : 18 avenue Edouard Belin, 31401 Toulouse, France

valentin@iuplo.univ-ubs.fr, olivier.ingremeau@univ-ubs.fr
catherine.lambert@cnes.fr, eric.martin@univ-ubs.fr

Résumé – . Ce papier présente une approche de l'adéquation algorithme-architecture pour les systèmes embarqués en milieu spatial. La tolérance aux fautes est traitée au niveau algorithmique et architectural, autour de composants commerciaux COTS[3] . Nous présentons le contexte de l'étude et une approche originale de la détection d'erreur par analyse de signature qui exploite les caractéristiques déterministes des algorithmes TDSI.

Abstract – This paper presents an approach of the algorithm –architecture adequacy for space embedded system. Fault-tolerance is processed at algorithmic and architectural level using COTS. We present the context of the study and an original approach of error-detection based on signature monitoring that exploits the determinism properties of signal and image processing algorithm.

1. Introduction

Les applications de traitement du signal et des images embarquées pour des missions spatiales sont de plus en plus complexes, ce qui implique des contraintes croissantes en puissance de traitement. Par ailleurs, les composants qualifiés spatial sont coûteux, peu disponibles et de moindre performance. Il y a donc un besoin de réduction des coûts et d'adaptation à la réduction de l'offre en composants performants haute fiabilité. Pour répondre à ces contraintes, il est nécessaire de faire appel à l'utilisation des composants commerciaux tels que la dernière génération de processeurs de traitement de signal (DSP). Malheureusement, ces composants sont sujets à des défaillances en présence des radiations rencontrées dans l'espace. Ces défaillances peuvent être matérielles ou fonctionnelles, et sont dues essentiellement au passage des ions lourds dans le composant. Il résulte alors deux types de fautes possibles:

- le déclenchement d'un effet thyristor par mise en conduction des transistors parasites localisés produisant un effet latch-up [1]. Cet effet peut être thermique et destructif.
- le changement accidentel d'état logique de bits d'information localisée sur les points de mémoire, aussi appelé SEU (Single Event Upset). Ce phénomène est particulièrement important pour les circuits à haute intégration.

Dans le but d'utiliser les composants commerciaux (COTS) dans les futurs projets spatiaux, nous sommes amenés à définir une méthode de conception d'architectures Tolérantes aux Fautes Transitoires (TFT[4]) pour les applications de traitement du signal et des images en environnement spatial. Cette démarche nécessite de prendre en compte les caractéristiques de l'algorithme de traitement de signal et relève de l'adéquation algorithmes et architectures [2].

L'application qui nous concerne est un algorithme de compression multirésoulution d'images satellites sous contraintes de la tolérance aux fautes transitoires. L'algorithme est basé sur la transformée en ondelettes au fil de l'eau [3].

A partir du modèle d'erreur retenu, notre approche consiste à étudier les interactions possibles entre les déterminismes de ce type d'algorithme et l'architecture du DSP. Les déterminismes tels que le séquencement, la vraisemblance des résultats de traitement, la valeur des codes d'instructions et la séquence d'adresses de données prédictibles, nous permettent d'établir des références pour la détection d'erreur. De l'architecture, nous extrayons les lieux de détection d'erreur pouvant donner un taux de couverture optimal. En associant ces deux aspects, référence et observabilité, nous définissons l'architecture TFT.

La deuxième partie présente le modèle d'erreur résultante de la faute du milieu spatial. La troisième traite la méthodologie de conception et propose un modèle d'architecture TFT. La quatrième partie présente des résultats de l'implémentation de l'algorithme (surcoût de mémoire et de temps).

2. Modèle d'erreur en milieu spatial et moyens de tolérance aux fautes

L'erreur résultante de la faute provoquée par les particules de radiations du milieu spatial peut être observable à différents niveaux d'abstraction : physique, logique, architecture et algorithme. Des moyens de tolérances adéquats peuvent être définis pour chacun de ces niveaux (cf tableau 1).

Au niveau **physique** la faute peut être évitée soit par technologie durcie (CMOS/SOS, SOI), soit par un blindage du composant, soit en intégrant dans le calculateur un détecteur de courant qui met à off le composant (évitement du latch-up). Au niveau **logique**, la faute physique peut se traduire par un SEU qui modifie l'état logique des bits. Cette erreur peut être minimisée par des moyens cités précédemment. Au niveau **fonctionnel**, l'erreur SEU du niveau logique perturbe le fonctionnement des unités fonctionnelles de l'architecture. Ce dysfonctionnement est dû aux modifications non conformes aux spécifications initiales des flux d'informations qui circulent dans l'architecture (données, programme, adresses, contrôle). L'évitement s'effectue par des moyens de tolérance aux fautes dans le domaine des systèmes informatiques. Au niveau **algorithme**, une faute physique peut être interprétée comme un « bruit » de calcul et provoque une erreur résiduelle dans les résultats de traitement.

Dans le cas qui nous concerne, l'erreur SEU due à une faute transitoire est considérée comme une erreur simple : un seul bit est erroné par mot mémoire.

TAB. 1 : Propagation et traitement d'erreur SEU

Niveau d'abstraction	Effets	Traitement d'erreur
Physique	Latch-up	Détecteur de courant, blindage, techno. durcie
Logique	SEU	Blindage, techno. durcie
Architecture	- Contrôle - Mémorisation	- test en ligne - Code correcteur, test en ligne
Algorithme	Traitement	- TFT intrinsèque de l'algorithme - Algorithme n versions

3. Méthode de conception

En se basant sur le principe d'adéquation algorithme architecture, la méthode de conception d'architecture TFT est déclinée en deux parties : la mise en œuvre méthodologique de la tolérance aux fautes logicielle par étude comportementale de l'algorithme en présence d'erreur SEU [3], et la définition de mécanismes de détection et de correction d'erreurs tant au niveau architectural (processeur) qu'au niveau circuit (mémoire).

3.1 Niveau algorithme

Pour permettre la mise en œuvre de mécanismes de traitement d'erreurs pouvant rendre l'unité de traitement de données TFT [4], nous analysons le comportement de l'algorithme en présence de fautes vis à vis du critère de qualité (RMSE pour une chaîne de compression d'image avec perte). L'algorithme est ainsi scindé en une partie TFT et une partie non-TFT. La partie TFT peut être considérée comme inhérente à la structure même de la transformée en ondelettes.

A la partie non-TFT est associée un mécanisme de traitement d'erreurs consistant en une programmation logicielle à 2 versions : duplication-comparaison. Le recouvrement d'erreur associé s'effectue par une reprise de traitement. En ne protégeant que les parties sensibles, le coût de la tolérance aux fautes logicielles est minimisé, notamment la perte de performance du DSP due à la duplication de traitement.

3.2 Niveau architecture

Au niveau architecture, la TFT est obtenue par la mise en œuvre des mécanismes de détection et de recouvrement d'erreurs pour toutes les unités fonctionnelles formant l'architecture. La détection repose sur deux points essentiels :

- L'analyse de traçabilité de la faute dans l'architecture. Celle ci à pour objectif de déterminer les points d'observation optimaux de l'apparition de l'erreur, par conséquent à donner au détecteur d'erreur une capacité maximale de détection. Cette étape constitue l'analyse comportementale de l'architecture en présence de SEU.
- Les propriétés invariantes dans l'algorithme de TDSI : séquencement, vraisemblance des résultats, séquence d'accès à la mémoire pré-déterministe, cohérence des valeurs des instructions à exécuter. Ces propriétés permettent d'établir des références pour la détection d'erreurs.

Au niveau processeur, l'étude de la traçabilité de la faute s'effectue, en décomposant le processeur en trois unités fonctionnelles : contrôle, mémorisation et traitement. Au niveau fonctionnel, l'erreur SEU se manifeste par les modifications non conformes de flux d'informations (programme, adresse, données) observables sur les bus externes du processeur.

Pour satisfaire les contraintes de temps réel, la détection d'erreur est basée sur les méthodes de test en ligne par analyse de signature.

3.2.1 TFT de l'unité de contrôle

La détection d'erreur pour l'unité de contrôle consiste en la vérification du bon déroulement du programme. Pour un faible coût et un taux de couverture acceptable, dans le cas de fautes transitoires, la détection d'erreur du flot de contrôle (séquencement et valeur de code) est basée sur le test en ligne continu par analyse de signature [5][6][7][8]. Elle est ensuite associée à la reprise pour constituer la TFT de l'unité de contrôle.

Détection d'erreur

La mise œuvre du test en ligne nécessite trois composantes : - le programme assembleur de l'application découpé en bloc linéaire d'instructions (BLI), un compacteur (logiciel/matériel) pour la génération de signature de référence et en ligne, et un moniteur pour la surveillance. La détection d'erreur du flot de contrôle s'appuie sur la connaissance des BLI dans le programme. Les BLI sont constitués des instructions dont le point d'entrée du bloc est une instruction de type « branch-in », et le point de sortie est uns instruction de type « branch-out ». La propriété fondamentale d'un BLI est qu'aucun branchement n'est autorisé à l'intérieur du bloc. Si une première instruction est exécutée alors toutes les autres instructions du bloc sont exécutées dans l'ordre indiqué par le programme assembleur. La signature de référence est obtenue par une

compaction logicielle des instructions contenues dans un BLI depuis la source assembleur de l'application. Au lieu d'introduire la signature de référence dans le programme, celle ci est placée dans une mémoire locale de l'analyseur de signature (AdS). Un pointeur d'adresse de signature est placé au début de chaque BLI pour accéder à la signature de référence. Deux bits indicateurs sont ajoutés aux mots mémoire pour distinguer le début, la fin du BLI et le pointeur de signature. En cours de fonctionnement, l'AdS génère une signature en ligne. Celle-ci est ensuite comparée à une signature de référence lorsque la fin du BLI est rencontrée. Une différence entre les deux signatures indique une présence d'erreur dans le système. La figure 2 montre l'organisation de la mémoire de programme.

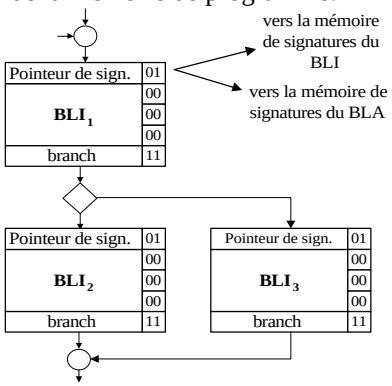

FIG 1 : Programme assembleur avec ajout de pointeurs d'adresses de signatures

La fonction de compaction choisie est la division polynomiale : $G(x) = 1 + X^1 + X^2 + X^{22} + X^{32}$, réalisée sur circuit par un MISR (Multiple Input Shift Register). L'architecture de l'AdS du flot de contrôle est illustrée à la figure 4.

Recouvrement d'erreurs : la reprise

Dans le cas de notre application, la reprise à « gros grain » est retenue. Elle consiste à reprendre l'exécution du programme depuis le début, lorsqu'une erreur est détectée. Ce mode de reprise nécessite un seul point de reprise, et elle est adaptée à une faible occurrence de SEU. Le tableau 2 montre deux taux d'occurrence de SEU pour deux DSP.

TAB. 2 : Taux d'occurrence de SEU

	TMS320C40	**ADSP21062**
Taux d'occurrence unitaire (SEU/ jour / bascule D)	1E-5	1E-5
Taux d'occurrence (SEU/ itération / processeur)	**2,29E-8**	**1E- 6**

3.2.2 TFT de l'unité de mémorisation

La détection d'erreurs d'accès à la mémoire de données utilise le même principe que celui du flot de contrôle : test en ligne par analyse de signature. Pour cela il est important d'établir un graphe flot d'adresses, appelé GFA. Sa synthèse s'effectue en se basant sur la séquence des adresses prédictible lors de la phase de compilation, par exemple la zone d'adressage des coefficients des filtres, et des données.

Le GFA est composé de blocs linéaires d'adresses, appelés BLA. Les BLA sont obtenues par extrapolation du graphe GFC à la séquence des adresses. Ainsi, un BLI peut être associé à un ou plusieurs BLA. Différents cas sont à analyser.

Cas 1 : un BLI produit un seul BLA.

Dans ce cas il y a une correspondance exacte entre le BLI et le BLA associé. La signature du BLA est générée de la même manière que celle obtenue pour le BLI (cf figure 2).

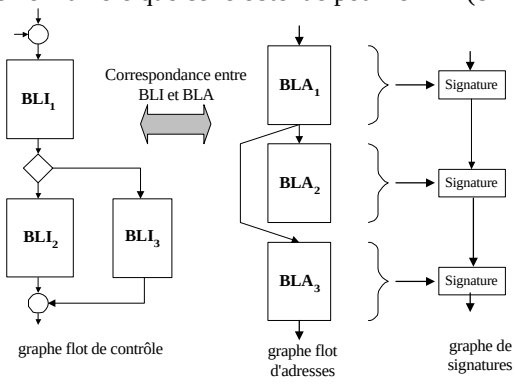

FIG 2 : un BLI pour un BLA

Cas 2 : un BLI produit plusieurs BLA

Ce cas est représentatif des algorithmes TDSI. Par exemple le cas d'une boucle où les codes d'instructions forment un seul BLI mais chaque itération i de la boucle produit un BLA différent (cf figure 3). Cependant, le pointeur placé au début du BLI ne peut pointer que sur une seule adresse de signature, il est donc nécessaire de rendre invariante l'ensemble des signatures des BLA par rapport aux itérations de la boucle, et d'obtenir une unique signature qui permet de couvrir l'ensemble des BLA générés par la boucle. Pour rendre invariantes les différentes signatures des BLA, nous inversons les matrices régissant le fonctionnement du MISR afin de déterminer un polynôme diviseur permettant d'avoir S1 = S2 = . . .=Sn. Selon le contenu du BLA, la soulution n'est pas unique. La résolution doit alors prendre en compte des degrés de liberté.

Cas 3 : séquence d'adresses non prédictible

Si la séquence d'adresses n'est pas prédictible lors de la phase de compilation (par exemple un adressage hors de la zone autorisée), il n'est pas possible de déterminer les BLA, donc de calculer la signature.

Le graphe de signature est implémenté dans une mémoire locale de l'analyseur. Les points de vérification sont synchronisés avec l'AdS du flot de contrôle.

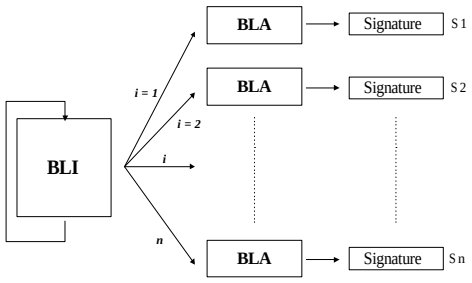

FIG 3 : Exemple d'une boucle

3.3 Modèle d'architecture TFT

Le modèle d'architecture TFT est présenté à la figure 4. Le masquage de pointeur de signature s'effectue en chargeant le DSP avec l'instruction NOP câblée dans le multiplexeur. Si une erreur est détectée le comparateur génère un signal d'interruption NMI qui annule le traitement en cours. L'algorithme est alors repris depuis le début. La machine d'état de l'AdS du flot d'adresses est synchronisée avec celle de l'AdS du flot de contrôle. Les deux mémoires de signatures sont des mémoires TFT et seront placées éventuellement à l'extérieur de l'AdS. L'implémentation peut être réalisée sur des FPGA de programmation anti-fusible, comme les FPGA d'Actel.

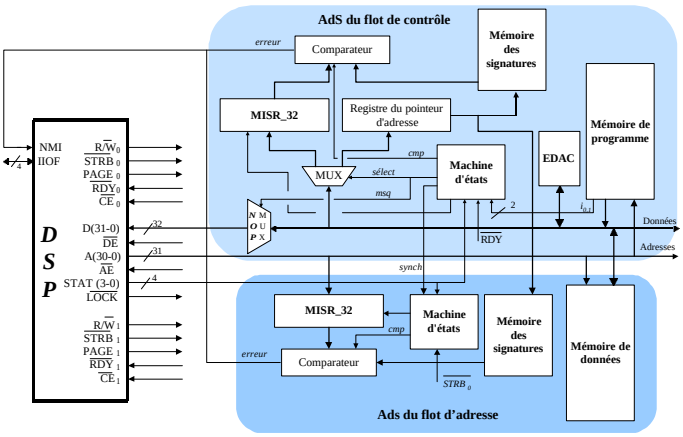

FIG. 4 : Modèle d'architecture TFT

4. Résultats

La chaîne de compression est décrite à la figure 5. Seule la partie T. O (transformée en ondelettes) a été implémentée et les résultats sont montrés dans le tableau 3.

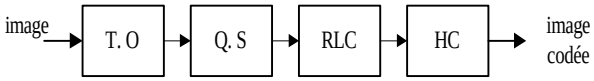

FIG 5: Chaîne de compression

Q. S : quantification scalaire ; RLC : Run Length Coding ; HC : Huffman Coding.

Mem. prog		***Mem. données***		***Mem. sign***
T.O (image 8x512)	*6 Ko*	*Image (8x512)*	*16 Ko*	*2 x 1,02 Ko*
Nb BLI dans T.O	*263*	*EDAC*	*3,5 Ko*	
Nb bits ajoutés (pointeurs + bit indic.)	*1,4 Ko*	*Reprise*	*19,5 Ko*	
Total	*7,4 Ko*	*Total*	*39 Ko*	
Surcoût	*24%*	*Surcoût*	*143 %*	
Surcoût total	*120 %*			

TAB. 3 : Surcoût de mémoire

Le nombre de bits ajoutés correspond au nombre de signature et de bits indicateurs qui sont ajoutés dans le programme. La taille de l'image est une bande de 8x512 pixels, et l'EDAC (Error Detection And Correction) ajoute à chaque mot de mémoire 7 bits de contrôle [9]. De plus 19.5 Ko de mémoire sont nécessaires pour assurer la reprise.

Le surcoût temporel introduit par l'insertion de signature dans l'algorithme sans mécanisme de tolérance aux fautes logicielle est d'environ 4%. Ce surcoût est évalué par rapport au cas où le DSP est sous la configuration : RAM interne off, et Cache d'instruction off.

5. Conclusion

Nous avons présenté dans cet article une méthode de conception d'architecture TFT utilisant les composants de commerciaux. Par cette méthode, l'algorithme est découpé en une partie qui est intrinsèquement tolérante aux fautes, et en une partie non-TFT. Seule la partie non-TFT est protégée contre les erreurs SEU par une programmation à 2 versions.

Au niveau matériel, l'unité de contrôle et de mémorisation interne du DSP sont rendues TFT par deux analyseurs de signature. Les résultats obtenus sont encourageants et devraient permettre la conception à faible coût. d'une architecture simplex tolérante aux fautes pour les applications TDSI.

Remerciements

Nous tenons à remercier C. Lecordier doctorante au LESTER, M. Pignol du département traitement de bords du CNES, ainsi que C. Lesthievent d'ALCATEL Espace.

Références

[1] M. Pircher, M. Labrunée, O. Musseau. *L'environnement spatial : impact des radiations cosmiques sur les satellites.* CNES-CEA/DAM, pp. 29-49.

[2] E. Martin, J-L. Philippe. *Ingénierie des systèmes à microprocesseurs : application au traitement du signal et de l'image.* Collection CENT-ENST, Edition MASSON, 1996.

[3] C. Lambert-Nebout, G. Moury. *A survey of on-board image compression for CNES space missions : past, present and futur.* DSp'98, 6th Inter. Workshop on Dignal Processing Techniques for Space Application, sept. 1998.

[4] T. Valentin, C. Lecordier, O. Ingremeau, E. Martin, C. Lambert-Nebout, C. Lesthievent. *Architecture tolérante aux fautes transitoires d'une chaîne de compression d'images embarquée pour mission spatiale.* Colloque λμ-11, 1998, pp. 575-585.

[5] M. Namjoo. *Technique for concurrent testing of VLSI processor operation.* ITC-82, 1982, pp. 461-468.

[6] M. Schuette, J. P. Shen. *Processor control flow monitoring using signature instruction streams.* IEEE Trans. Comp. vol. C-36. no. 3, march 1987. pp. 264-276.

[7] J. P. Shen, M. A. Schuette. *On-line self-monitoring using signatured instruction stream.* ITC-83, 1983, pp. 275-282.

[8] J. Ohlson, M. Rimen, *Implicit signature checking.* FTCS-25, 1995, pp. 218-227.

[9] National Semiconductor.*EDAC's.* Guide Users-1995.

[3] *COTS : Components Off The Shelf désigne les composants commerciaux, appelé également composants sur étagère*

[4] *TFT ; ce terme désignera dans la suite de l'article « tolérance » ou « tolérante » aux fautes transitoires*