

Premiers résultats sur l'utilisation d'ACL2 pour l'évaluation de la conséquence des erreurs logiques

Renaud CLAVEL, Laurence PIERRE, Régis LEVEUGLE
 TIMA (CNRS-GrenobleINP-UJF)
 46 Av. Félix Viallet - 38031 Grenoble cedex
 Email: Prenom.Nom@imag.fr

Abstract—Nous nous proposons de développer de nouvelles méthodologies, basées sur une combinaison de techniques d'injection de fautes et de méthodes formelles, pour l'analyse de la robustesse d'un circuit décrit au niveau RTL, vis à vis des erreurs créées par des fautes transitoires. Nous présentons ici nos premiers résultats quant à l'utilisation du démonstrateur de théorèmes ACL2, dans le contexte de systèmes avec dispositif de correction.

de certaines propriétés de fiabilité en présence de fautes transitoires.

Des travaux récents s'intéressent également à l'utilisation de méthodes formelles dans le contexte de l'injection de fautes. Les approches de [4] et de [5] font appel à des techniques de model-checking mais ont recours à la génération de mutants spécifiques ou énumèrent toutes les erreurs possibles, et n'exploitent donc pas la puissance de représentations symboliques. Le travail de [6] vise plus spécifiquement l'analyse du taux de couverture de la logique de détection ou de correction d'erreur. Enfin, [7] propose une représentation plus formelle de modèles d'injection de fautes (sous forme de règles d'inférence) mais concerne particulièrement la validation de mécanismes logiciels de tolérance aux fautes.

I. INTRODUCTION - CONTEXTE

La conception de circuits fiables nécessite en particulier de pouvoir évaluer, à chacune de ses étapes, le niveau de robustesse atteint vis à vis de divers types de fautes ou d'erreurs [1]. Dans les systèmes critiques (aéronautique et autres transports, centrales nucléaires, etc), les erreurs peuvent avoir des conséquences dramatiques, et sont généralement causées par des phénomènes naturels comme des impacts de particules ou des perturbations électromagnétiques. Les causes des erreurs, appelées fautes, sont généralement modélisées dans les systèmes digitaux par des inversions de bits ou des collages; elles peuvent être permanentes ou transitoires. Avec l'évolution des technologies, la sensibilité des circuits aux fautes transitoires s'accroît et devient une question cruciale. Le problème se pose également dans le contexte des circuits tels que les systèmes cryptographiques qui peuvent, eux, être victimes d'attaques volontaires destinées à récupérer des informations secrètes [2]. Dans tous les cas, il est primordial de pouvoir garantir un certain niveau de robustesse.

L'analyse de la conséquence des erreurs est classiquement basée sur des techniques dites d'injection de fautes, mises en oeuvre par simulation ou émulation [3]. Cependant, pour être utilisables en pratique, ces techniques ne peuvent réaliser que des analyses partielles, reposant sur l'injection d'un sous-ensemble des erreurs possibles. Cela peut être insuffisant pour garantir que certaines propriétés de fiabilité sont respectées pour toutes les erreurs amenées à se produire. C'est pourquoi nous nous proposons de développer de nouvelles méthodologies s'appuyant sur l'utilisation de méthodes formelles. Ces méthodologies visent des IPs synchrones décrites au niveau RTL, sujettes à des fautes transitoires donnant lieu à des inversions de bits simples ou multiples. Le but recherché est de *vérifier formellement l'existence*

II. OBJECTIF DU PROJET

L'objectif que nous poursuivons est de modéliser formellement le circuit aussi bien que l'injection de fautes, et de raisonner sur ces modèles au moyen de techniques de démonstration automatique, de résolution de contraintes, ou de model-checking.

Grâce à une méthode de simulation symbolique adaptée de [8], nous obtenons, à partir d'une description VHDL au niveau RTL, une représentation symbolique des fonctions de transition et de sortie, λ et δ . Plus précisément, étant donnés I , O et S les ensembles des entrées, sorties, et états, dans le cas général (machine de Mealy), $\lambda : I \times S \rightarrow S$ et $\delta : I \times S \rightarrow O$. Le but est de symboliser également l'injection de fautes sous forme de fonctions ou de contraintes, de façon à pouvoir vérifier formellement des propriétés sur les fonctions λ et δ en présence de fautes.

III. PREMIERS RÉSULTATS

Dans un premier temps, nous nous sommes intéressés à l'évaluation des possibilités fournies par un outil de démonstration automatique tel qu'ACL2 [9], [10]. Ce démonstrateur de théorèmes est basé sur une logique du premier ordre sans quantificateur (les variables sont implicitement universellement quantifiées), avec égalité. Il est construit au-dessus de Common Lisp et manipule donc des représentations fonctionnelles (fonctions totales). Ses mécanismes reposent essentiellement sur deux puissants principes : le principe de définition récursive et le principe d'induction. Grâce au principe de définition récursive, une fonction récursive ne peut

être admise par l'outil que si sa terminaison est garantie, c'est à dire si, pour chaque appel récursif, il existe un argument (ou une combinaison des arguments) dont la mesure décroît selon une relation bien fondée (classiquement la relation d'ordre $<$). Grâce au principe d'induction, les mesures utilisées dans la phase d'acceptation des fonctions récursives sont reprises pour permettre l'automatisation complète des preuves par induction : la variable et le schéma d'induction sont déterminés automatiquement. L'un des avantages majeurs de cet outil est donc son haut niveau d'automatisation. Il a été intensivement utilisé pour la preuve formelle d'architectures matérielles complexes [11], [12], et diverses autres structures [13].

Nous l'utilisons ici dans le contexte de la vérification de propriétés respectées en présence de fautes. Plus précisément, nous nous intéressons à la vérification de propriétés d'auto-correction dans un circuit comprenant un dispositif TMR (Triple Modular Redundancy). Notre expérimentation se fait sur un système très simple de compteur, illustré par la Fig. 1 : la sortie d'un incrémenteur (avec incrémentation conditionnée par le signal *inc*) est connectée au dispositif TMR, le résultat est produit sur la sortie *count_out*.

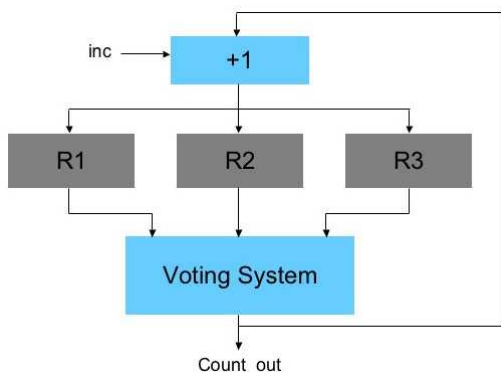


Fig. 1. Exemple : compteur avec dispositif TMR

Nous vérifions que ce système est capable d'auto-correction, dans le cas où une faute est injectée dans l'un des registres R_1 , R_2 ou R_3 . Le système est représenté par des fonctions λ et δ , tel que décrit dans la section II.

Pour la modélisation du processus d'injection de fautes, nous adaptions le principe suivant : il est associé à une fonction f que nous ne caractérisons pas par une définition, mais par une conjonction de contraintes. Ainsi, nous ne définissons pas exhaustivement toutes les fautes possibles, mais nous les caractérisons logiquement, dans leur ensemble. La fonction d'injection de fautes f est donc spécifiée ici par la conjonction des propriétés suivantes :

- elle prend en paramètre un état s , et renvoie un état $f(s)$
- $f(s)$ est différent de s (l'injection de faute est effective)
- $f(s)$ ne diffère de s que par rapport à un seul élément de mémorisation (une seule faute est injectée).

Bien qu'ACL2 soit limité à la logique du premier ordre, cette caractérisation d'ordre supérieur peut être codée facilement au moyen du *principe d'encapsulation* de cet outil.

Nous réalisons alors les vérifications suivantes :

- 1) l'état initial S_0 des trois registres étant $(0, 0, 0)$, si une

faute est injectée après n tops d'horloge, l'erreur sera corrigée au top d'horloge suivant, i.e. l'état obtenu sera équivalent à l'état obtenu sans injection de faute :

$$S_0 = (0, 0, 0) \Rightarrow (\lambda(f(\lambda^n(S_0, I)), i) \Leftrightarrow \lambda(\lambda^n(S_0, I), i))$$

où I est une séquence d'entrées et i est l'entrée courante.

- 2) même propriété, mais plus générale : on part d'un état initial S_0 valant (X, X, X) , X quelconque :

$$S_0 = (X, X, X) \Rightarrow (\lambda(f(\lambda^n(S_0, I)), i) \Leftrightarrow \lambda(\lambda^n(S_0, I), i))$$
- 3) si l'injection de faute est réalisée dans n'importe quel état valide S_n (ses trois registres ont des valeurs égales), l'erreur sera corrigée au top d'horloge suivant :

$$S_n = (X, Y, Z) \wedge X = Y \wedge X = Z$$

$$\Rightarrow (\lambda(f(S_n), i) \Leftrightarrow \lambda(S_n, i))$$

Les temps CPU (sur Intel Xeon à 1.6 GHz) pour les preuves de ces trois théorèmes dans ACL2 sont :

- Théorème 1 : 5.47 secondes
- Théorème 2 : 6.35 secondes
- Théorème 3 : 0.01 secondes. Ce temps bien meilleur s'explique par le fait que le théorème ne fait pas intervenir de puissance de λ et sa preuve ne nécessite donc pas d'induction.

IV. CONCLUSION

Nous avons présenté nos premiers résultats sur l'utilisation d'un outil de démonstration automatique pour la vérification de propriétés en présence de fautes. Nous avons pu prouver des propriétés génériques (remarquons notamment que n est un entier positif quelconque), avec une représentation logique du processus d'injection de fautes. Ces travaux se poursuivent dans cette direction, et également dans la perspective de mettre en jeu des outils de résolution de contraintes ou de model-checking pour d'autres cas de figures.

REFERENCES

- [1] L. Anghel, R. Leveugle, and P. Vanhauwaert, "Evaluation of SET and SEU effects at multiple abstraction levels," in *Proc. 11th IEEE International On-Line Testing Symposium*, July 2005.
- [2] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," *Proceedings of the IEEE*, vol. 94, no. 2, Feb. 2006.
- [3] R. Leveugle and K. Hadjiat, "Multi-level fault injections in VHDL descriptions: alternative approaches and experiments," *Journal of Electronic Testing: Theory and Applications*, vol. 19, no. 5, Oct. 2003.
- [4] R. Leveugle, "A new approach for early dependability evaluation based on formal property checking and controlled mutation," in *Proc. 11th IEEE International On-Line Testing Symposium*, July 2005.
- [5] S. Seshia, W. Li, and S. Mitra, "Verification-guided soft error resilience," in *Proc. DATE'07*, April 2007.
- [6] U. Krautz, M. Pflanz, C. Jacobi, H. Tast, K. Weber, and H. Vierhaus, "Evaluating Coverage of Error Detection Logic for Soft Errors using Formal Methods," in *Proc. DATE'06*, March 2006.
- [7] D. Larsson and R. Hähnle, "Symbolic Fault Injection," in *Proc. 4th International Verification Workshop*, July 2007.
- [8] G. A. Sammane, "Symbolic simulation of circuits described at the algorithmic level," Ph.D. dissertation, Joseph Fourier University, Grenoble, France, July 2005.
- [9] M. Kaufmann, P. Manolios, and J. Moore, *Computer Aided Reasoning: an Approach*. Kluwer Academic Pub., 2002.
- [10] L. Pierre, "Tutorial sur ACL2," http://deptinfo.unice.fr/~lpierre/Tutorial_T1/.
- [11] B. Brock, M. Kaufmann, and J. Moore, "ACL2 Theorems about Commercial Microprocessors," in *Proc. FMCAD'96*, 1996.
- [12] J. Sawada and W. A. Hunt, "Results of the verification of a complex pipelined machine model," in *Proc. CHARME'99*, September 1999.
- [13] M. Kaufmann, P. Manolios, and J. Moore, *Computer Aided Reasoning: ACL2 Case Studies*. Kluwer Academic Pub., 2000.