Test March pour la Détection des Fautes Dynamiques dans les Décodeurs de Mémoires SRAM

Luigi Dilillo, Patrick Girard, Serge Pravossoudovitch, Arnaud Virazel, Simone Borri, Magali Bastian Hage-Hassan

▶ **To cite this version:**

Luigi Dilillo, Patrick Girard, Serge Pravossoudovitch, Arnaud Virazel, Simone Borri, et al.. Test March pour la Détection des Fautes Dynamiques dans les Décodeurs de Mémoires SRAM. JNRDM'04 : 7ièmes Journées Nationales du Réseau Doctoral de Microélectronique, May 2004, Marseille, France. pp.495-497. lirmm-00108644

HAL Id: lirmm-00108644

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00108644

Submitted on 23 Oct 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Test March pour la Détection des Fautes Dynamiques dans les Décodeurs d'Adresse de Mémoire SRAM

Luigi Dilillo Patrick Girard Serge Pravossoudovitch Arnaud Virazel

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier – LIRMM
Université de Montpellier II / CNRS
161, rue Ada – 34392 Montpellier Cedex 5, France
Email: <nom>@lirmm.fr
URL: http://www.lirmm.fr/~w3mic

Simone Borri Magali Hage-Hassan

Infineon Technologies France
2600, Route des Crêtes – 06560 Sophia-Antipolis, France
Emails: simone.borri@infineon.com magali.hage@infineon.com
URL: http://www.infineon.com

Résumé

Cet article présente de nouveaux éléments March adaptés à la détection des fautes de circuit ouvert dans les décodeurs d'adresse appelées ADOFs pour "Address Decoder Open Fault". La détection de telles fautes est l'objectif premier de cette étude car elles sont réputées pour être difficile à détecter. Dans cet article nous proposons d'adapter l'algorithme de test présenté dans [1, 2] aux formalismes March. Les nouveaux éléments issus de cette étude sont caractérisés par un ordre particulier du parcours d'adresse ainsi que des données particulières en lecture/écriture.

1. Introduction

Les besoins croissants en performance, capacité de traitement de données, faible consommation d'énergie et coût de production, entraînent l'émergence de nouveaux concepts de design tels que les systèmes sur puce ou SoC pour "System-on-Chip". Ainsi, un grand nombre de fonctionnalités peuvent être adressées sur un SoC en comparaison d'un système sur carte, plus complexe et coûteux. Afin de satisfaire les besoins liés à la conception d'un SoC, les ressources mémoire nécessaire sont en croissante augmentation. Ceci est confirmé par la "SIA roadmap" qui prévoit une occupation mémoire proche de 94% de la surface totale du SoC d'ici les dix prochaines années [3]. Il est donc primordial d'apporter des solutions de test efficaces pour les mémoires car elles vont jouer un rôle majeur sur le rendement de fabrication.

Généralement, pour déceler la présence de pannes dans une mémoire des algorithmes de type March [4, 5] sont généralement les plus utilisés en raison de leur complexité linéaire. Cependant, ils sont construits essentiellement pour détecter les fautes dites statiques; fautes ayant besoin d'une seule opération pour être sensibilisées.

Les procédés de conception actuels de mémoire permettent d'obtenir une densité d'intégration importante et une amélioration des performances, mais au détriment de l'occurrence de nouvelles défaillances physiques. Les fautes résultantes à ces défaillances sont appelées fautes dynamiques. En référence à la classification présentée dans [6], une faute est dynamique quand plus d'une opération appliquée séquentiellement est nécessaire pour la sensibiliser, notamment plusieurs lectures ou écritures. Parmi les fautes dynamiques connues, nous concentrons notre étude sur les ADOFs, faute dont l'origine physique est un circuit ouvert dans le plan parallèle des portes logiques du décodeur d'adresse.

Dans cet article nous adaptons un algorithme dédié au test des ADOFs, proposé dans [1, 2], aux formalismes March. Cette adaptation est obtenue en décomposant le test des ADOFs en phase de sensibilisation et d'observation. Ces deux phases de test nous permettent de définir un ordre d'adressage spécifique et des données particulières en lecture/écriture embarquées dans des éléments March. Ces nouveaux éléments garantissent la détection des ADOFs pour une complexité inférieure à l'algorithme originel.

Le reste du papier est organisé comme suit. La section 2 donne quelques détails du fonctionnement d'un décodeur d'adresse sain ainsi que le comportement fautif en présence d'une ADOF. La section 3 présente les conditions de test des ADOFs et introduit les nouveaux

éléments March permettant de les détecter. Les conclusions sont discutées en section 4.

2. Décodeur d'adresse et ADOF

Le schéma de la Figure 1 présente un décodeur de ligne ou WL pour "word line" à deux bits. Il utilise des portes NOR pour la sélection des WLs. Les portes NAND et INV remplissent respectivement les fonctions de synchronisation et d'amplification. Un autre décodeur d'adresse similaire est utilisé pour la sélection des colonnes ou BL pour "bit line". Une telle structure est adoptée dans l'architecture SRAM synchrone produite par la société Infineon en technologie 0,13µm.

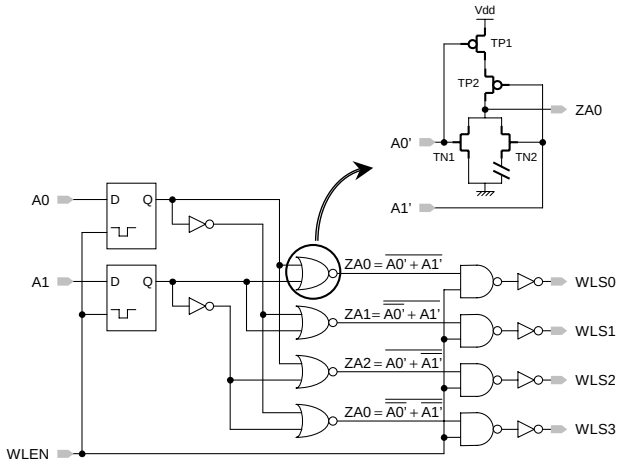

Figure 1: Décoder d'adresse de "word line"

Le décodeur d'adresse est contrôlé par les entrées d'adresse (A0 et A1) ainsi qu'un signal de synchronisation (WLEN). Le fonctionnement normal de ce décodeur consiste à sélectionner une seule WL à la fois en fonction des valeurs sur les entées A0 et A1. Par exemple :

1. <A0, A1> = <0, 0>
   ⇒ WLS0 est activée.
2. transition montante sur A1 : <A0, A1> = <0, 1>
   ⇒ WLS2 est activée et WLS0 est désactivée.

Parmi l'ensemble des localisations possibles pour un défaut de circuit ouvert, le plan parallèle des transistors de la porte NOR permet d'obtenir un comportement fautif modélisé par une faute dynamique et plus particulièrement une ADOF. En référence à la porte NOR du schéma de la Figure 1, un tel défaut peut être situé au niveau du drain, de la source ou de la grille des transistors TN1 ou TN2. A titre d'exemple, nous avons inséré un défaut de circuit ouvert au noeud source du transistor TN2. Le défaut peut produire un comportement fautif de la porte NOR empêchant la désactivation de WLS0 dû à un effet mémoire des capacités du noeud et d'entrée de la porte suivante. Dans ce cas, deux WLs peuvent être actives en même temps. En conséquence, deux cellules mémoire, au lieu d'une seule, sont sélectionnées pour une opération de lecture ou écriture. Considérant le défaut de la Figure 1, la séquence suivante met en évidence le comportement fautif du décodeur :

1. < A0, A1 > = < 0, 0 >
   ⇒ WLS0 est activé;
2. transition montante sur A1: <A0, A1> = <0, 1>
   ⇒ WLS2 est activé et WLS0 reste activé.

Les chronogrammes de la Figure 2 résument le comportement sain et fautif (courbe grisée) du décodeur de WL. Pour ce dernier, le défaut de circuit ouvert sur TN2 empêche la mise à zéro ("pull-down") du noeud ZA0, qui reste donc au niveau logique '1' en raison de l'effet mémoire. Dans cet exemple, nous avons d'abord activé WLS0, (adresse <0, 0>) et puis WLS2 (adresse <0, 1>). Il est à noter que les deux adresses diffèrent d'un seul bit. Le changement de plus d'un bit entre les deux adresses masquerait l'effet de faute. Par exemple, après l'adresse <0, 0>, l'application de l'adresse <1, 1> positionne le '0' logique sur le nœud ZA0 via le transistor sain TN1 masquant ainsi le comportement fautif de TN2. En conséquence, une des conditions nécessaire pour tester les ADOFs est l'utilisation d'un ordre d'adressage avec distance de Hamming de 1 (Hd = 1), *i.e.* chaque adresse doit présenter une transition d'un seul bit en comparaison avec la précédente.

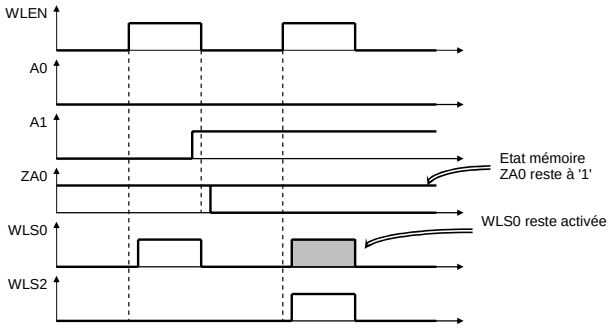

Figure 2: Fonctionnement sain et fautif d'un décodeur de WL

3. Solution de test des ADOFs

Dans [1] est proposé une solution algorithmique permettant de sensibiliser et observer l'ensemble des ADOFs pouvant affecter les décodeurs d'adresse. Cet algorithme, nommé algorithme de Sachdev, est composé des trois phases suivantes :

a. '0' est écrit dans une certaine cellule X d'adresse Adx;
b. '1' est écrit dans une cellule Y d'adresse Ady, avec Hd = 1 entre Adx et Ady;
c. la cellule X est lue; un '0' est prévu.

Les phases b et c sont réitérées n fois, pour toutes les cellules dont les adresses ont Hd = 1 avec l'adresse de la cellule X; n est le nombre de bits du décodeur. En présence d'une ADOF, pendant la phase b, un '1' est écrit dans la cellule Y mais aussi dans la cellule X car elle reste sélectionnée anormalement. La phase c permet l'observation de l'effet de faute. En effet, un '0' est attendu mais un '1' est lu en présence d'une ADOF. L'algorithme de Sachdev est donc efficace pour la sensibilisation et l'observation des ADOFs et sa complexité est $(2n+1)x2^n$. Cependant, cet algorithme est dédié uniquement au test des ADOFs et nécessite une structure de génération propre. L'objectif de cette étude consiste donc à implanter l'algorithme de Sachdev sous forme d'éléments March qui

peuvent facilement être rajoutés à un algorithme March existant. Ainsi, une seule structure de génération sera nécessaire.

Les trois phases de l'algorithme de Sachdev peuvent être graphiquement illustrées sous forme de motif comme sur le schéma in Figure 3, où entre Ad0 et Ad1 Hd = 1, d = donnée (0 ou 1) et $\overline{d}$ sa valeur opposée.

Figure 4: Motif de Sachdev

L'adaptation March de ce motif est obtenue en le réitérant pour toutes les adresses possibles, comme représenté sur le schéma de la Figure 4. Au lieu d'effectuer la sensibilisation et l'observation séparément pour chaque cellule, elles sont réalisées par phase :

- Sensibilisation pour toutes les cellules par une écriture en série avec les données alternatives d et $\overline{d}$.
- Observation par lecture des données écrites.

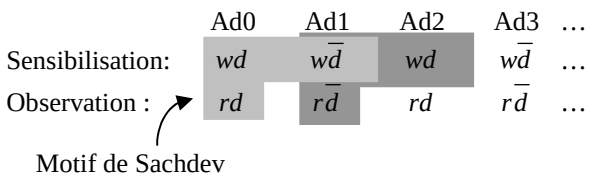

Figure 4: Adaptation March de l'algorithme de Sachdev

Nous obtenons ainsi les éléments March présentés dans la Figure 5, où A est une valeur logique partant de '0', pour la première adresse, et est inversée à chaque nouvelle adresse.

$$\{ \Uparrow (wA) \Uparrow (rA) \}$$ A valeur logique alternée
M M

Figure 5: Nouveaux éléments March pour tester les ADOF

Pour ces nouveaux éléments March, la succession des adresses doit toujours respecter la condition Hd = 1 entre deux adresses successives. Cette condition peut être satisfaite en exploitant le premier des six degrés de liberté ou "Degrees of Freedom" (DOF) [7] des tests March :

> **DOF I** : *Tout ordre arbitraire d'adresse peut être défini en tant qu'ordre croissant $\Uparrow$, à condition que toutes les adresses apparaisses exactement une fois ($\Downarrow$ est l'inverse de $\Uparrow$). Les propriétés de détection de faute sont indépendantes de l'ordre d'adresse utilisé [7].*

La possibilité de changer la donnée pendant l'exécution d'un élément March est justifiée par le quatrième DOF :

> **DOF IV :** *Les données, pour une opération de lecture/écriture, ne doivent pas nécessairement être équivalentes pour toutes les adresses mémoire tant que la probabilité de détection des fautes n'est pas changée [7].*

Ces nouveaux éléments March permettent de détecter les ADOFs car ils reproduisent les étapes nécessaires à la sensibilisation et l'observation des fautes. De plus, ils peuvent directement être ajoutés à un test March existant en augmentant ainsi ses propriétés de détection de faute. Nos éléments March ont aussi l'avantage d'être moins complexe que l'algorithme d'origine. En effet, la complexité des nouveaux éléments March est $2x2^n$ comparée à $(2n+1)x2^n$ pour l'algorithme de Sachdev, n étant le nombre de bit d'adresse.

4. Conclusion

Cette étude a portée sur les fautes dynamiques qui peuvent affecter les décodeurs d'adresse de mémoire SRAM. En présence d'une ADOF, le comportement fautif du décodeur se matérialise par une double sélection de WL ou BL.

La solution proposée consiste à traduire un algorithme dédié au test de ces fautes en élément March directement utilisable dans un test March existant. La détection des ADOFs est donc assurée par ces nouveaux éléments dont la complexité est moindre par rapport aux méthodes antérieures. Ces éléments utilisent un ordre d'adressage particulier et des données alternées lors de la phase d'écriture ou lecture.

Références

[1] M. Sachdev, "*Test and Testability Techniques for Open Defects in RAM Address Decoders*", Proc. IEEE European Design & Test Conference, 1996, pp.428-434.

[2] M. Sachdev, "*Open Defects in CMOS RAM Address Decoders*", IEEE Design & Test of Computers, vol.14, n.2, Apr-Jun 1997, pp. 26-33.

[3] Semiconductor Industry Association (SIA), "*International Technology Roadmap for Semiconductors (ITRS)*", 2003 Edition.

[4] A.J. van de Goor, "*Using March Tests to Test SRAMs*", *IEEE Design & Test of Computers*, vol.10, n.1, Jun 1993, pp.8-14.

[5] R.D. Adams, "*High Performance Memory Testing*", Kluwer Academic Publishers, Sept. 2002.

[6] Z. Al-Ars and A.J. van de Goor, "*Static and Dynamic Behavior of Memory Cell Array Opens and Shorts in Embedded DRAMs*", Proc. Design, Automation and Test in Europe, 2001, pp. 496-503.

[7] D. Niggemeyer, M. Redeker and J. Otterstedt, "*Integration of Non-classical Faults in Standard March Tests*", Records of the IEEE Int. Workshop on Memory Technology, Design and Testing, 1998.