



M2 EEA
« Systèmes Electroniques Intégrés »

HMEE327

virazel@lirmm.fr



Test de Systèmes Intégrés Digitaux et Mixtes

Arnaud Virazel

virazel@lirmm.fr



Plan

- Chapitre 1 : *Introduction*
- Chapitre 2 : *Modélisation de Fautes*
- Chapitre 3 : *Simulation de Fautes*
- Chapitre 4 : *Analyse de Testabilité*
- Chapitre 5 : *ATPG*
- Chapitre 6 : *Test de Circuits Séquentiels*
- Chapitre 7 : *Test de Fautes de Délais*
- Chapitre 8 : *Test de Mémoires*



Chapitre 1

Introduction

Arnaud Virazel

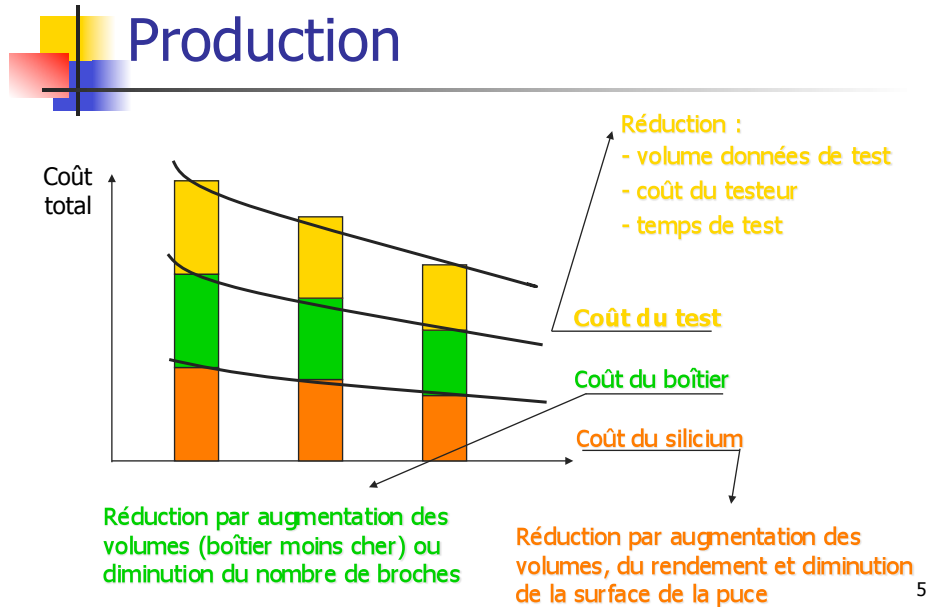
virazel@lirmm.fr



Vérification et Test

- Vérifier la correction de la conception
- Effectuée par simulation, émulation matérielle, preuve formelle
- Effectuée une fois avant la fabrication
- Responsable de la qualité de la conception
- Vérifier la correction du matériel fabriqué
- Deux étapes :
 - Génération du programme de test réalisée **une fois** pendant la conception
 - Application du test : tests électriques appliqués au matériel
- Le test électrique est :
 - appliqué à **chaque matériel** produit
 - responsable de la qualité du circuit

Réduction du Coût de Production



Pourquoi Tester ?

- Clauses contractuelles au niveau :
 - d'une inspection d'entrée,
 - de performances de fiabilité
- Imposé par un environnement de plus en plus compétitif où la qualité des produits et leur fiabilité sont parmi les points importants mis en avant par la clientèle (consommateurs)

Quand Tester ?

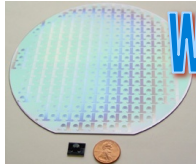
Circuit



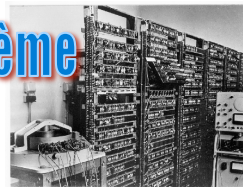
Plaque



Wafer



Système



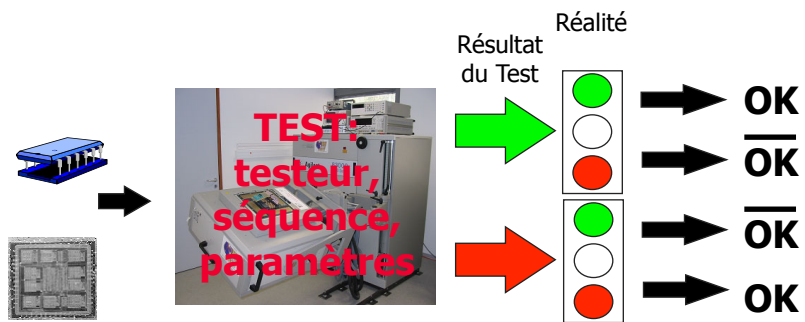
7

Quand Tester ?

Dispositif sous Test			
Wafer	Boîtier	Plaque	Système
Test du processus technologique	Test paramétrique	Test d'entrée	Test du système
Test des puces	Test logique	Test de la plaque nue	Test en utilisation
	Vieillessement	Test de la plaque montée	
	Test complet sur échantillon	Vieillessement	

8

Le Test est un Filtre



9

Le Test met en Œuvre des Matériels Coûteux (M\$)



10



Définition du Test

- Le test est possible quand on peut appliquer un **stimulus connu** à une entité dans un **état connu** et que la **réponse connue** peut être évaluée
 - **stimulus connu** : avoir accès aux entrées de l'entité et appliquer une valeur connue
⇒ **CONTROLABILITE**
 - **état connu** : déterminisme de l'influence des entrées sur le circuit
 - **réponse connue** : avoir accès aux sorties de l'entité et comparer la réponse du circuit à une valeur réputée bonne
⇒ **OBSERVABILITE**

11



Définitions et Terminologie

- Quelque chose fait que le système ne fonctionne pas :
 - **Erreur** : comportement erroné observé
 - **Faute** : déviation de la structure par rapport aux spécifications
 - **Défaut** : déviation de la réalisation physique par rapport aux spécifications de fabrication
 - **Panne** : mauvais fonctionnement en opération
 - **Test** : détection du défaut
 - **Diagnostic** : détection et localisation du défaut

12



Chapitre 2

Modélisation de Fautes

Arnaud Virazel

virazel@lirmm.fr



Plan

- **Généralités**
- Caractérisation des défauts
- Modélisation des défauts
 - Collage
 - Court circuit
 - Fautes de délais
- Equivalence de fautes
- Notions de test

2



Comment Obtenir une Séquence de Test ?

- Utiliser une **séquence fonctionnelle** produite pour vérifier le fonctionnement par simulation lors de la phase de conception
 - facile à trouver
 - plus difficile à valider
 - encore plus difficile à améliorer

3



Comment Obtenir une Séquence de Test ?

- Appliquer une **séquence exhaustive** des entrées
 - Beaucoup trop long
 - Exemple
 - Un circuit possédant 64 entrées
 - $2^{64} = 18^{18}$ vecteurs
 - Application de la séquence à 1 GHz
 - Environ 585 années sont nécessaires pour appliquer la séquence
- ⇒ Test structurel – Utilisation d'un modèle de faute

4

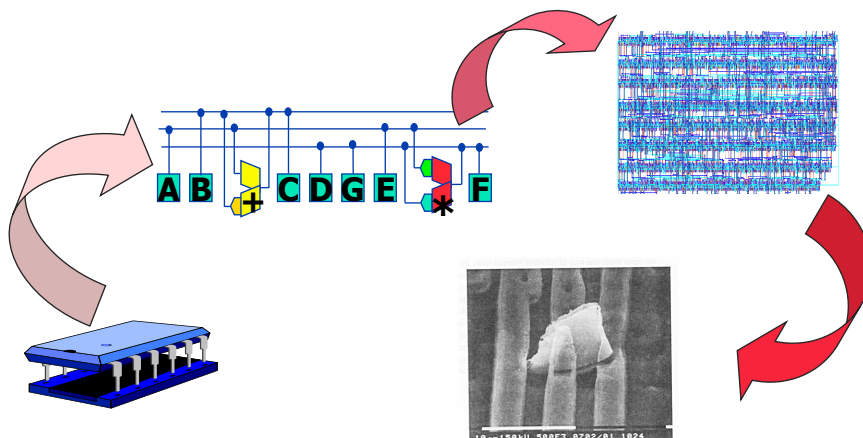
Comment Obtenir une Séquence de Test ?

- **Test Structurel** : Utiliser un **modèle de faute**, l'appliquer au circuit et essayer de détecter toutes les fautes
 - Validité du modèle
 - taux de couverture et "defect level"
 - Outils de génération

5

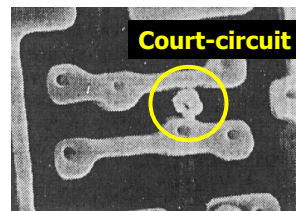
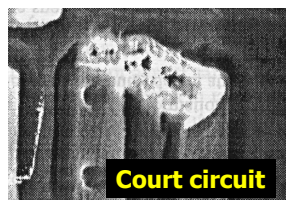
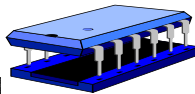
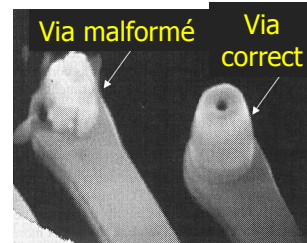
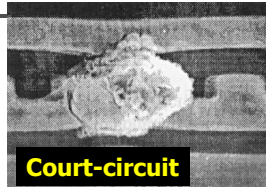
Que tester ?

- Défaillances dans le processus de fabrication



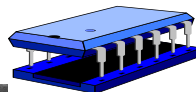
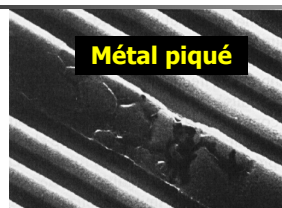
6

Défaillances dans le Processus de Fabrication

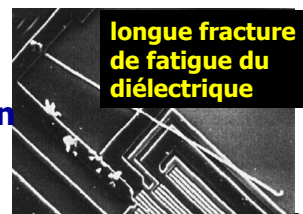


7

Défaillances dans le Processus de Fabrication



Phénomènes d'électromigration



8



Plan

- Généralités
- **Caractérisation des défauts**
- Modélisation des défauts
 - Collage
 - Court circuit
 - Fautes de délais
- Equivalence de fautes
- Notions de test

9



Mécanismes de Défaillance

- Défauts du wafer (contamination, micro-crevasse)
- Erreurs humaines (interactions humaines avec le processus de production)
- Défaillance d'équipements (utilisation de maintenance préventive)
- Impact de l'environnement (contaminations diverses, vibrations, ..)
- Instabilités du processus technologique de production



DEFAUTS GLOBAUX

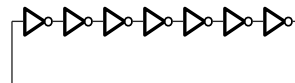
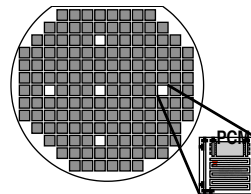


DEFAUTS LOCAUX

10

Monitoring des Défauts Globaux

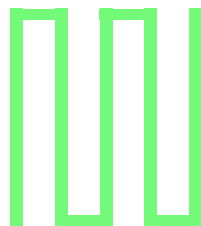
- Utilisation de PCM ("Process Control Monitor")
 - composés de structures de base (transistors, lignes de conducteur, chaîne de vias, ...)
 - distribués sur le wafer (éventuellement placés sur les lignes de découpe)
- Utilisation d'oscillateur en anneau
 - monitoring de paramètres de haut niveau
 - fréquence d'oscillation fonction des paramètres de plus bas niveau du processus technologique
 - résultats peuvent être mal interprétés



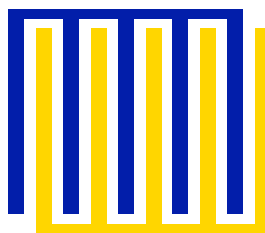
11

Monitoring des Défauts Locaux

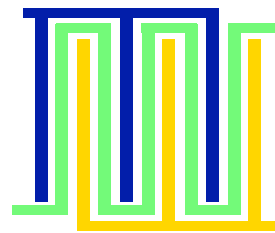
- Monitoring en ligne par inspection à différentes étapes du processus de fabrication (surfscan, réflectométrie, évaluation d'image, ...)
- Moniteurs d'oxyde de grille (combinaison de condensateurs de différentes formes et tailles, mesure de courant de fuite et de capacités)
- Moniteurs d'interconnexions



Méandre



Double peigne



Méandre+ peignes

12



Plan

- Généralités
- Caractérisation des défauts
- **Modélisation des défauts**
 - Collage
 - Court circuit
 - Fautes de délais
- Equivalence de fautes
- Notions de test

13

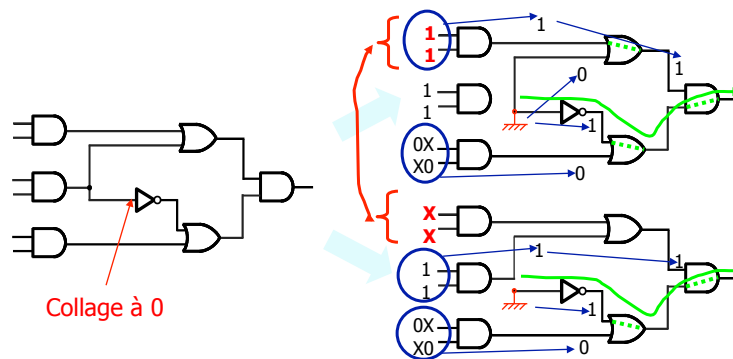


Le Modèle de Collage Simple

- Avec le modèle de collage, une (et une seule) ligne de la description est collée de manière permanente à la valeur 0 ou à la valeur 1
- On utilise parfois uniquement le collage des ports d'entrée et de sortie des modules (ces deux modèles sont équivalents sauf en cas de divergence)

14

Le Modèle de Collage Simple



15

Principales Caractéristiques et Avantages

- Il permet de représenter de nombreux défauts physiques différents
- Il est indépendant de la technologie
- Il permet d'utiliser l'algèbre booléenne pour trouver les vecteurs de test
- Les vecteurs de test générés avec ce modèle de collage détectent aussi d'autres défauts
- L'ensemble des fautes obtenu avec ce modèle est limité
- Le Taux de Couverture (**TC**) associé à ce modèle de fautes est une métrique admise entre fournisseurs et clients

$$TC = \frac{\text{Nb de fautes détectées}}{\text{Nb de fautes total}}$$

- Le modèle de collage peut être utilisé pour modéliser d'autres types de fautes

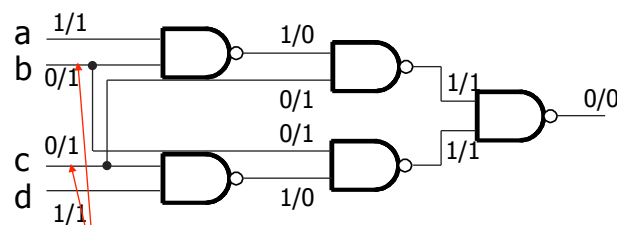
16

Le Modèle de Collage Multiple

- Extension du modèle de collage avec plusieurs lignes collées simultanément
- Avec n sites possibles pour une faute de collage simple il y a $3^n - 1$ fautes de collage multiple possibles ($2n$ fautes de collage simple)
- Les fautes de collage multiple ont été introduites à cause des masquages (phénomène peu courant en pratique)

17

Le Modèle de Collage Multiple



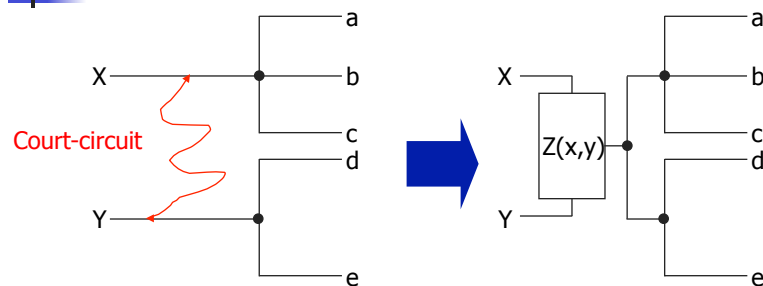
Collage multiple à 1

a	b	c	d
1	1	1	1
0	1	1	1
1	1	1	0
1	0	0	1
1	0	1	0
0	1	0	1

Séquence de test complète pour les fautes de collage

18

Le Modèle de "Bridging Fault"



- $Z(x,y) = z$
 - $Z(0,1) = 0$ correspond au modèle de **ET cablé**
 - $Z(0,1) = 1$ correspond au modèle de **OU cablé**

19

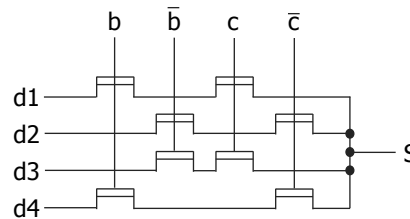
Caractéristiques Principales

- Introduit à l'origine pour les technologies TTL et ECL
- Généralement appliqué sur une description à portes, le court-circuit est défini entre les sorties de portes et/ou les entrées primaires (pas de différence entre racine et branches de divergence)
- S'il existe un chemin fonctionnel entre les deux extrémités du court-circuit, celui-ci crée une boucle de contre-réaction (transformant par exemple un circuit combinatoire en circuit séquentiel avec possibilité d'oscillations)
- Les fautes de court-circuit multiples ont également été introduites (uniquement au niveau théorique)
- Le modèle de "bridging fault" n'est pas très bien adapté pour les technologies CMOS pour lesquelles la fonction $Z(x,y)$ ne correspond pas toujours à une valeur logique

20

Inadaptation du Modèle de Collage

- Problèmes dus à la modélisation au niveau porte
 - Utilisation du transistor interrupteur dans une logique à relais

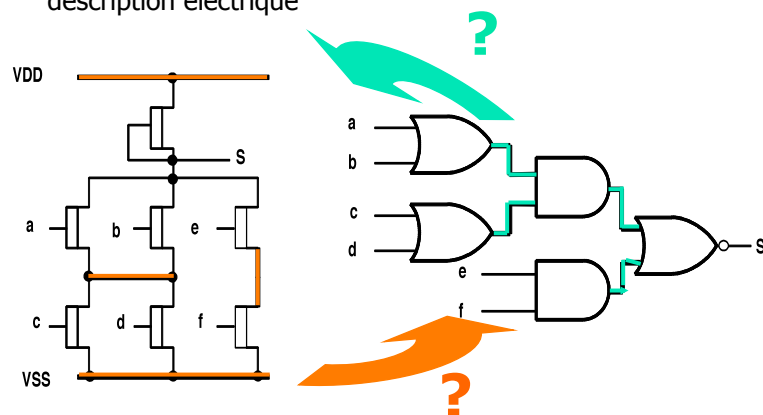


Fonction multiplexage

21

Inadaptation du Modèle de Collage

- Non correspondance entre la description à porte et la description électrique



22

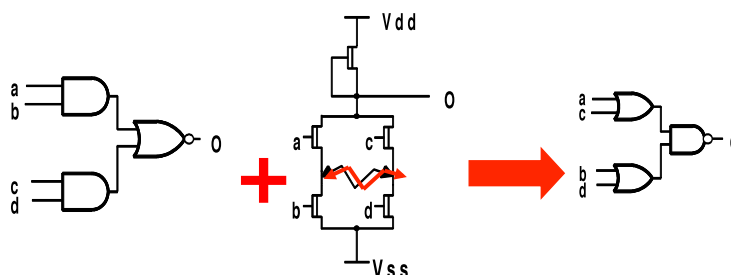
Inadaptation du Modèle de Collage

- Problèmes dus à la nature des défauts
 - Les conséquences électriques principales des défauts physiques dans les technologies CMOS se manifestent le plus souvent par des courts-circuits et des circuits ouverts
 - Ces perturbations peuvent entraîner des dysfonctionnements du circuit tels que
 - modifications de la fonction logique réalisée par le circuit,
 - effet mémoire
 - comportement analogique
 - Dans la plupart des cas, ces dysfonctionnements ne sont pas modélisables par des fautes de collage

23

Inadaptation du Modèle de Collage

- Modification de la fonction logique



<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>O</i>	<i>Collage détectés</i>	<i>O (avec le CC)</i>
0	1	0	1	1	Sa1(a) Sa1(c)	1
1	0	1	0	1	Sa1(b) Sa1(d)	1
1	1	0	X	0	Sa0(a) Sa0(b)	0
0	X	1	1	0	Sa0(c) Sa0(d)	0

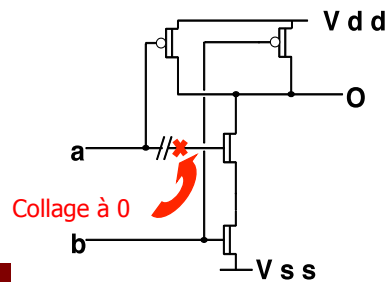
24

Inadaptation du Modèle de Collage

■ Effet mémoire



	<i>a</i>	<i>b</i>	<i>O</i>	<i>O (avec la faute)</i>
V1	0	0	1	1
V2	0	1	1	1
V3	1	0	1	1
V4	1	1	0	?



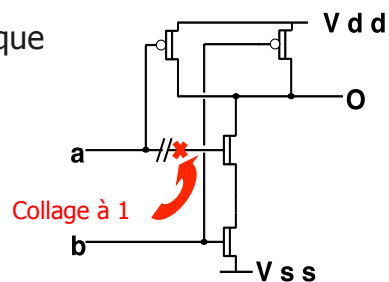
25

Inadaptation du Modèle de Collage

■ Comportement analogique



	<i>a</i>	<i>b</i>	<i>O</i>	<i>O (avec la faute)</i>
V1	0	0	1	1
V2	0	1	1	?
V3	1	0	1	1
V4	1	1	0	0



Valeur intermédiaire
dépendant de la
résistance passante
des transistors

26

Inadaptation du Modèle de Collage

Valeur intermédiaire dépendant de la
résistance passante des transistors

Valeur logique erronée :

- Possibilité de détection par un test logique classique
- Augmentation du courant consommé ("IDDQ testing")

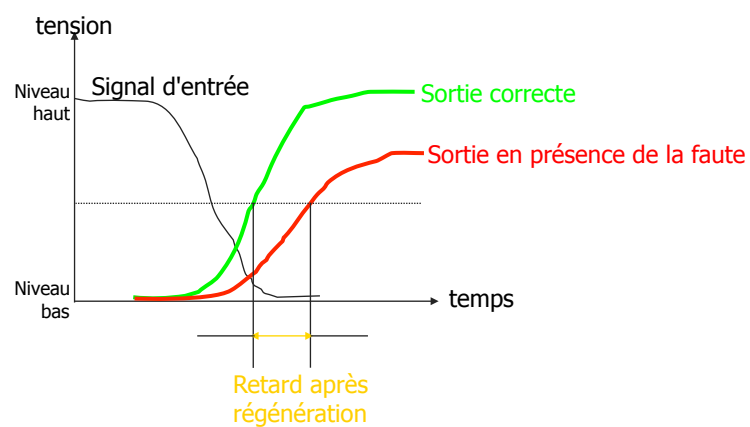
Valeur logique correcte :

- Pas d'erreur logique détectable en statique
- Augmentation du courant consommé ("IDDQ testing")
- Fautes de délais après régénération du signal par les éléments aval

27

Inadaptation du Modèle de Collage

Faute de délai



28



Plan

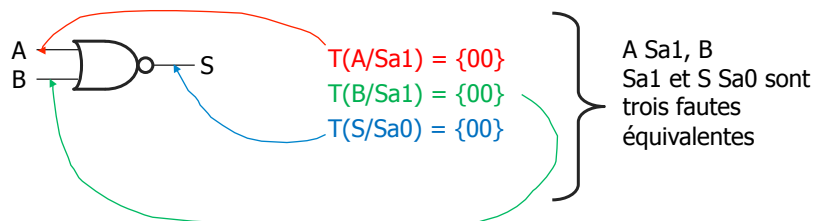
- Généralités
- Caractérisation des défauts
- Modélisation des défauts
 - Collage
 - Court circuit
 - Fautes de délais
- **Equivalence de fautes**
- Notions de test

29



Définition - Equivalence

- Soit $T(f_i)$ l'ensemble de tous les tests qui détectent la faute f_i
- Deux fautes f_i et f_j sont **équivalentes** si et seulement si $T(f_i) = T(f_j)$
- Tout test qui détecte f_i détecte f_j et inversement

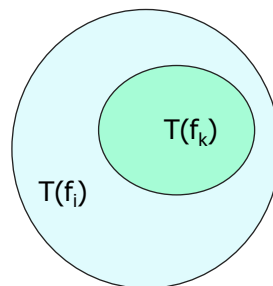


30



Définition - Implication

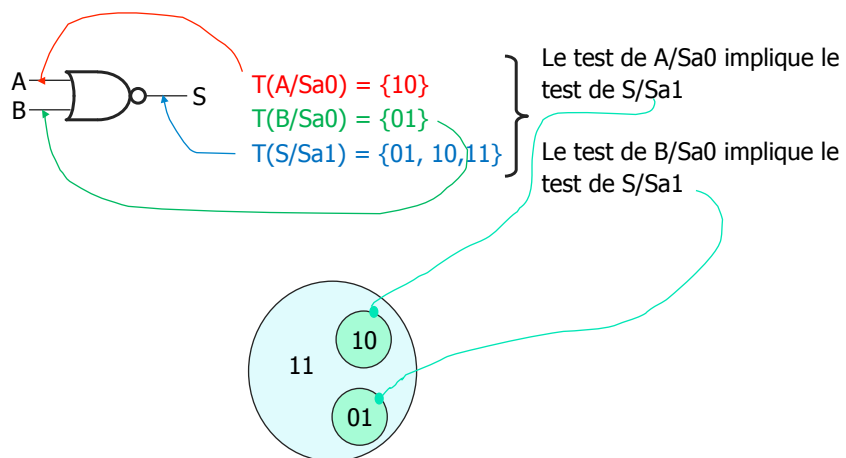
- Une faute f_i **domine** la faute f_k si et seulement si $T(f_k) \subset T(f_i)$
- Tout test qui détecte f_k détecte f_i
- On dit aussi que le test de f_k **implique** le test de f_i



31



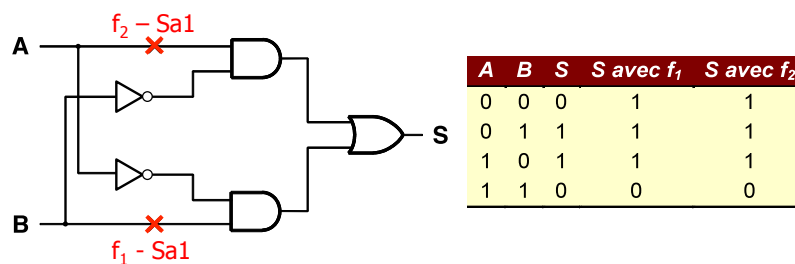
Exemple d'Implication



32

Equivalence de Fautes

- La détermination des fautes équivalentes est un problème NP-difficile
- Uniquement les équivalences structurelles locales peuvent être prises en compte



33

Niveau Porte

Entrées/sortie			Détection de collage (#)					
A	B	S	Sa0(A)	Sa0(B)	Sa1(A)	Sa1(B)	Sa0(S)	Sa1(S)
0	0	1			#	#	#	
0	1	0		#				#
1	0	0	#					#
1	1	0						#

Généralisation

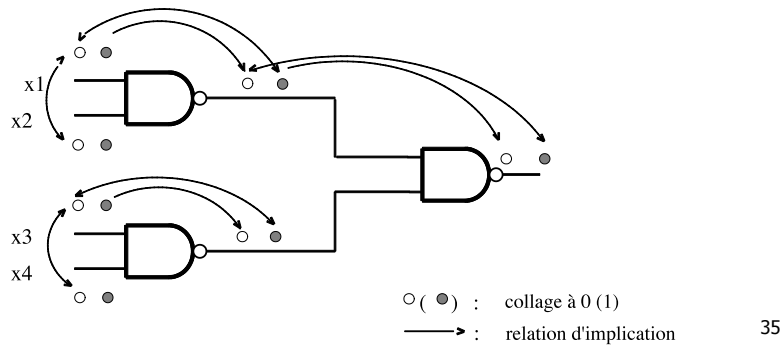
- Pour une porte avec une valeur prioritaire c et une inversion i, toute faute de collage à c d'une entrée est équivalente au collage à $c \oplus i$ (équivalence stricte) de la sortie
- Pour une porte avec une valeur prioritaire c et une inversion i, tout test de la faute de collage à \bar{c} d'une entrée implique le test au collage à $c \oplus i$ (implication seulement) de la sortie

34

Niveau Porte

Théorème 1

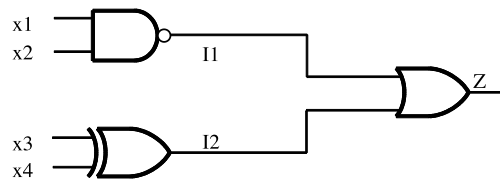
- Dans un arbre (pas de divergence) combinatoire réalisé avec des portes conventionnelles, tout test détectant toutes les fautes de collage des entrées primaires détecte toutes les fautes de collage



35

Niveau Porte

Problème avec les portes OU exclusif



Ensemble de test complet pour le collage des entrées												Détection des collages sur les entrées (#)	
x ₁	x ₂	x ₃	x ₄	SA0(x ₁)	SA1(x ₁)	SA0(x ₂)	SA1(x ₂)	SA0(x ₃)	SA1(x ₃)	SA0(x ₄)	SA1(x ₄)	I ₂	
1	0	1	1				#					0	
1	1	1	1	#		#		#		#		0	
0	1	1	1		#							0	
1	1	0	0	#		#			#		#	0	

36

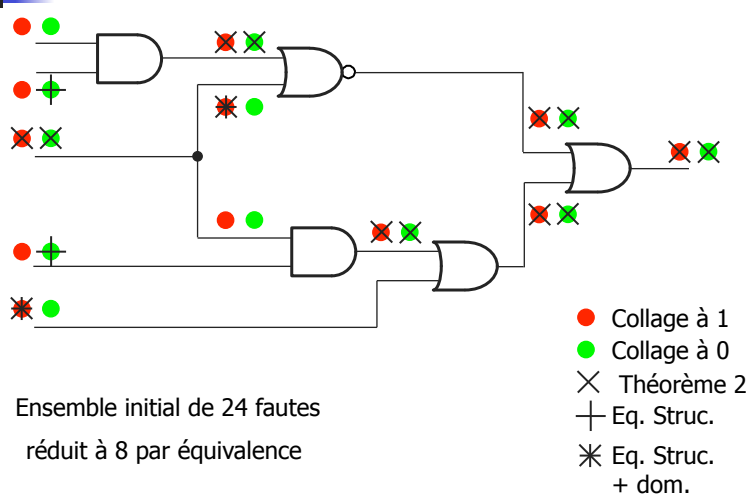
Niveau Porte

Théorème 2

- Dans un circuit combinatoire réalisé avec des portes conventionnelles, tout test détectant toutes les fautes de collage des entrées primaires et des branches de divergences détecte toutes les fautes de collage
- Les entrées primaires et les branches de divergence sont appelés "*checkpoints*"
- L'ensemble des "*checkpoints*" peut encore être réduit en utilisant des relations au niveau porte

37

Exemple



38



Plan

- Généralités
- Caractérisation des défauts
- Modélisation des défauts
 - Collage
 - Court circuit
 - Fautes de délais
- Equivalence de fautes
- **Notions de test**

39

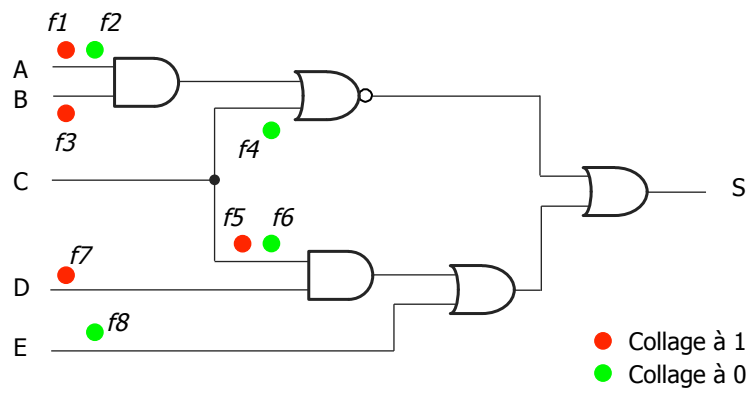


Notions de Test

- Sensibilisation
 - Appliquer la valeur logique sensibilisant la faute
⇒ 0(1) pour un Sa1 (0)
- Propagation
 - Ouvrir un chemin de propagation non-masquant (valeur non prioritaire sur les portes) afin d'observer l'effet de faute sur un point observable (une sortie primaire)
- Justification
 - Justifier jusqu'aux entrées primaires l'ensemble des valeurs logiques fixées durant les étapes de sensibilisation et propagation

40

Exemple



41



Chapitre 3

Simulation de Fautes

Arnaud Virazel

virazel@lirmm.fr

1



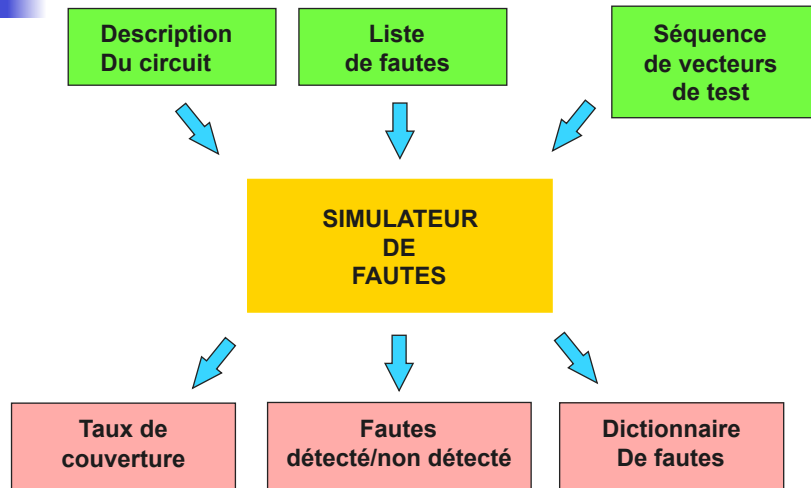
Objectifs

de la Simulation de Fautes

- Détermine le comportement du circuit en présence de chacune des fautes
- Déterminer les fautes détecté par une séquence de vecteur de test
- Grand nombre de fautes + PB complexe
 - Faute unique, logique et permanente
 - Certains simulateurs prennent aussi en compte les modèles nécessaires au traitement des technologies MOS (réseaux de transistors et fautes de transistors collés passant, collés ouverts, les courts-circuits, les coupures).

2

Objectifs de la Simulation de Fautes



3

Objectifs de la Simulation de Fautes

- Déterminer le Taux de couverture d'une séquence de test
- Établir la liste des fautes non détectées
- Dresser le dictionnaire de fautes (but : diagnostic), c-a-d la liste des fautes détectées, les vecteurs détectant chacune de ces fautes et les réponses du circuit

4



Classification

- Description du circuit
 - Fonctionnelle, structurelle
- Modelés de fautes
 - Collage, délai, court-circuit ...
- Codage
 - Deux valeurs (0,1)
 - Trois valeurs (0, 1, X)
 - Quatre valeurs (0, 1, X, Z)

5



Classification

- Timing
 - Zero-delay : aucune retard est considéré durant la simulation
 - Unit-delay : à chaque porte est associé un retard

6



Principales Techniques de Simulation de Fautes

- Simulation de fautes série
- Simulation de fautes parallèle
- Simulation de fautes déductive
- Simulation de fautes concurrente

7



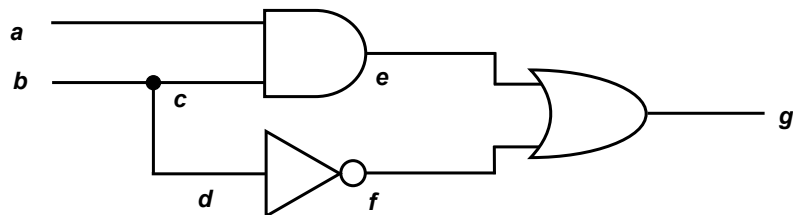
Simulation de Fautes Série

- Simulation logique du circuit sain + n simulations indépendantes des n circuits fautifs
- Avantages :
 - Simple. Ne nécessite pas de développement d'un simulateur de fautes spécialisé. Peut être implanté à partir d'un simulateur logique classique.
 - Permet de simuler n'importe quel type de faute pouvant être prise en compte par le simulateur logique.
 - « Consomme » très peu d'espace mémoire
- Problèmes :
 - très lent (n+1 simulations consécutives pour une liste de n fautes)

8

Exercice

- Déterminer les fautes détectées par les vecteurs $V(a,b) = (1,1)$ et $(0,0)$



9

Simulation de Fautes Parallèle

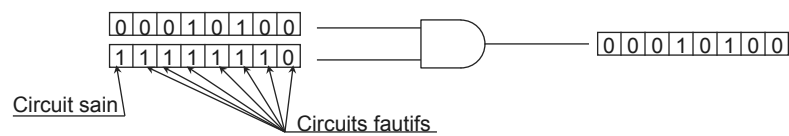
- Principe
 - Tire partie du traitement parallèle (par mot) de l'information binaire dans les ordinateurs
 - Si l'ordinateur hôte manipule des mots de n bits, on simule en parallèle le fonctionnement du circuit sain et de $n-1$ circuits fautifs
 - ⇒ Le nombre de fautes manipulées lors d'une simulation est fonction de la longueur des mots machines
 - ⇒ Éventuellement plusieurs phases
 - A chaque nœud on associe un mot de n bits
 - le premier bit représente l'état du circuit sain, chaque autre bit représente l'état d'un circuit fautif
 - L'évaluation des différents éléments du circuit s'effectue en utilisant les opérateurs logiques de l'ordinateur hôte

10



Simulation de Fautes Parallèle

■ Exemple



11



Simulation de Fautes Parallèle

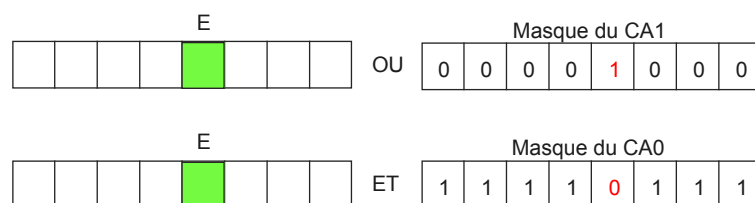
- Les fautes sont prises en compte (ou injectées) par l'intermédiaire de masques logiques
 - Le Sa1 d'une équipotentielle (E) est injecté en effectuant le OU du mot de E et du masque du Sa1
 - Le Sa0 d'une équipotentielle (E) est injecté en effectuant le ET du mot de E et du masque du Sa0

12



Simulation de Fautes Parallèle

- Les masques d'injection de fautes sont pris en compte lors de l'évaluation de la sortie d'un élément



13



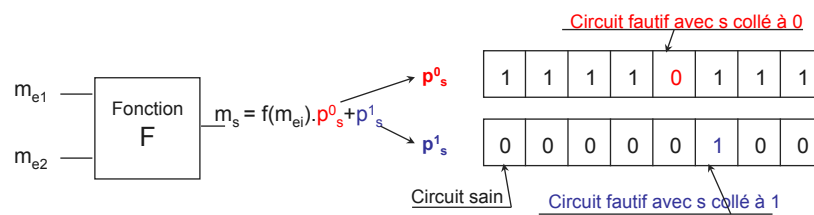
Simulation de Fautes Parallèle

- Exemple d'une porte AND ($c = a \cdot b$)
 - ma et mb sont les deux mots affectés aux entrées
 - $p0$ et $p1$ sont les deux masques correspondant au Sa0 et au Sa1 de la sortie c
 - mot affecté à la sortie c
 - $mc = ma \cdot mb$ si les deux fautes de collage de la sortie c ne sont pas pris en compte
 - $mc = ma \cdot mb \cdot p0$ si seul le Sa0 de la sortie c est pris en compte
 - $mc = ma \cdot mb + p1$ si seul le Sa1 de la sortie c est pris en compte
 - $mc = ma \cdot mb \cdot p0 + p1$ si les deux collages sont pris en compte

14

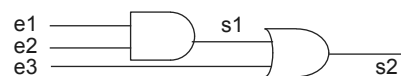
Simulation de Fautes Parallèle

■ Cas général



15

Exemple



■ Mot machine de 5 bit

sain	S1 Sa0	S1 Sa1	S2 Sa0	S2 Sa1
------	--------	--------	--------	--------

16

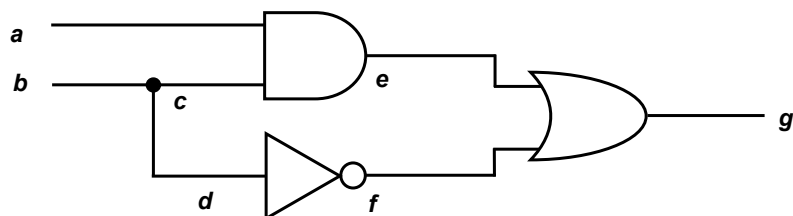
Exemple

- Injection de fautes
 - $s1 \text{ Sa}0 \text{ } p0(s1) = 10111$ $s2 \text{ Sa}0 \text{ } p0(s2) = 11101$
 - $s1 \text{ Sa}1 \text{ } p1(s1) = 00100$ $s2 \text{ Sa}1 \text{ } p1(s2) = 00001$
- Vecteur de test V et mots d'entrée
 - $V(e1, e2, e3) = 110$
 $m(e1) = 11111, m(e2) = 11111$ et $m(e3) = 00000$
- Evaluation
 - $m(s1) = m(e1) \cdot m(e2) \cdot p0(s1) + p1(s1) = 10111$
 - $m(s2) = (m(s1) + m(e3)) \cdot p0(s2) + p1(s2) = 10101$
- Conclusion
 - valeur circuit sain $s2 = 1$
 - $V = 110$ met en évidence les $\text{Sa}0$ de $s1$ et $s2$

17

Exercice

- Déterminer les fautes détectées par le vecteur $V(a,b) = (1,1)$
- On suppose avoir $n = 15$



18



Simulation de Fautes Dédutive

- Permet de simuler un nombre de fautes théoriquement infini (contrairement à la simulation parallèle : n-1 fautes)
- On simule uniquement le circuit sain et on en déduit les comportements des circuits défectueux
- Problème
 - Nécessite d'une quantité de mémoire très importante
- On associe à chaque équipotentielle :
 - 1/ la valeur dans le circuit sain
 - 2/ Une liste de fautes telles que le comportement du circuit sain est différent du comportement du circuit en présence d'une de ces fautes

19



Simulation de Fautes Dédutive

- Déduction des listes de fautes en sortie de portes

- si toutes les entrées sont à une valeur non prioritaire,

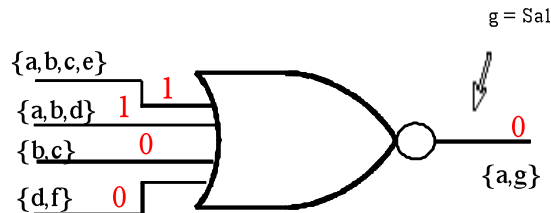
$$\text{Liste de sortie} = \left(\bigcup_{\substack{Li \\ \text{Toutes les} \\ \text{entrées}}} \right) \cup \left(\text{Sortie collée à la valeur prioritaire de sortie} \right)$$

- si certaines entrées sont à une valeur prioritaire,

$$\text{Liste de sortie} = \left(\bigcap_{\substack{Li \\ \text{entrées} \\ \text{prioritaires}}} \right) \cap \left(\bigcup_{\substack{Lj \\ \text{entrées non} \\ \text{prioritaires}}} \right) \cup \left(\text{Sortie collée à la valeur non prioritaire de sortie} \right)$$

20

Exemple

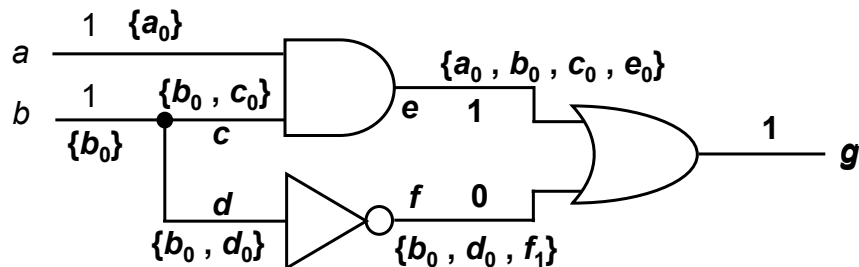


Liste de départ: $\{a, b, c, d, e, f, g, h\}$
 $\{a, b, c, e\} \setminus \{a, b, d\} = \{a, b\}$
 $\{b, c\} \oplus \{d, f\} = \{b, c, d, f\}$
complément de $\{b, c, d, f\} = \{a, e, g, h\}$
 $\{a, b\} \setminus \{a, e, g, h\} = \{a\}$
 $\{a\} \oplus \{g\} = \{a, g\}$

21

Exemple

- L_k est une liste de fautes pour la ligne k
- k_n est un San de la ligne k



$L_g = (L_e \cap \overline{L_f}) \cup \{g_0\} = \{a_0, c_0, e_0, g_0\}$
Fautes détecté par le vecteur (1, 1)

22



Simulation de Fautes Concurrente

- Définition 1 : L'activité d'une faute (taux de contamination d'un circuit par une faute à un instant donné)
 - $A = n / N$ avec n = nombre d'éléments affectés et N = nombre total d'éléments
- Définition 2 : L'activité moyenne est la moyenne des activités de toutes les fautes simulées pendant toute la durée de la simulation
- En pratique
 - L'activité moyenne varie de 1% à 5%
 - Donc 95% à 99% du circuit reste insensible à la faute
- Principe de la simulation concurrente
 - On ne simule que les portions de circuits défectueuses qui présentent une différence avec le circuit sain

23

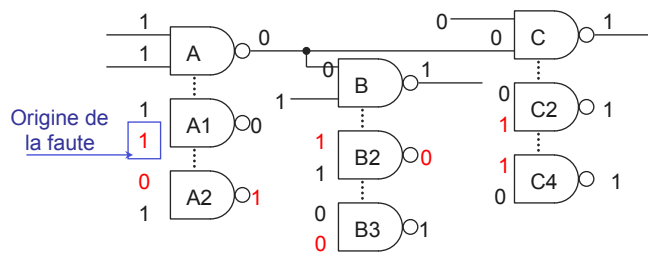


Simulation de Fautes Concurrente

- On associe une liste de configurations fautives à chaque élément du circuit
- Les liste de configurations fautives mémorisent les configurations logiques des éléments (valeurs d'entrée et de sortie)

24

Simulation de Fautes Concurrente



Remarque : les configurations des éléments de circuits fautifs qui sont identiques à celles du circuit sain ne sont pas mémorisées (ex : B1, C1)
(exception pour l'origine de la faute (ex : A1))

25

Simulation de Fautes Concurrente

- Les éléments du circuit sont évalués par la même procédure, suite
 - à un changement dans le circuit sain
 - à un changement sur un élément d'un certain circuit fautif (seul les éléments du circuit fautif sont affectés)
- Suivant les états logiques propagés au cours d'une simulation, l'effet d'une faute peut apparaître ou disparaître

26



Simulation de Fautes Concurrente

- On doit insérer ou retirer des configurations défectueuses dans la liste des configurations fautives de chaque élément
 - ⇒ Gestion dynamique des listes de configurations fautives
- Divergence
 - Création et insertion d'une configuration fautive
- Convergence
 - Suppression d'une configuration fautive lorsque l'effet de la faute devient invisible

27



Conclusion

- Simulation parallèle (liste de valeurs)
 - + place mémoire et rapidité
 - blocs complexes (doivent être transformés en logique booléenne) et fautes complexes
- Simulation déductive (liste de fautes)
 - + un seul passage
 - blocs complexes, simulation de plus de trois états trop complexe et espace mémoire difficilement prévisible en pointe
- Simulation concurrente (liste de configurations fautives)
 - + rapidité, simulation simultanée, modèle de fautes
 - espace mémoire, algorithme complexe

28



Chapitre 4

Analyse de Testabilité

Arnaud Virazel

virazel@lirmm.fr



Analyse de Testabilité

- Mesures utilisées comme critères de choix
 - Générateurs automatiques de vecteurs de test
 - Outils de conception en vue du test
- Testabilité
 - Caractéristique d'un circuit qui influence divers coûts associés au test (longueur, complexité de génération d'une séquence, ...)
 - En pratique : Basée sur les concepts de **contrôlabilité** et d'**observabilité**

2

Mesures de Testabilité

■ Contrôlabilité

- Indique la difficulté relative à positionner une ligne à 0 ou à 1 à partir des Entrées Primaires (EP)

■ Observabilité

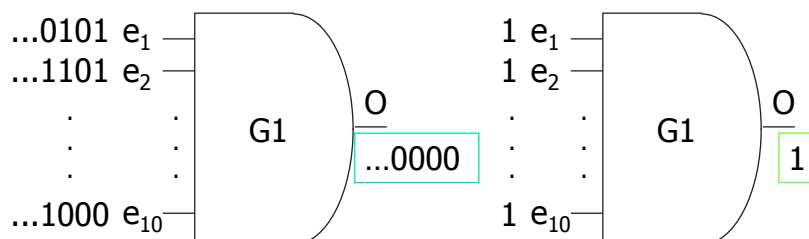
- Indique la difficulté relative à propager une erreur à partir d'une ligne vers les Sorties Primaires (SP)

3

Exemple

Test du collage à 0 de la sortie de G1

- Génération de vecteurs aléatoires
 - Contrôle à 1 de la sortie de G1 très difficile (probabilité de $1/2^n$)
- Génération de vecteurs déterministes
 - Contrôle à 1 de la sortie de G1 évident



4



Différentes Méthodes

- Obtention des mesures dans le meilleur cas pour un test déterministe
 - SCOAP et dérivées
- Obtention des mesures à partir de calculs probabilistes pour un test aléatoire ou pseudo aléatoire
 - COP

5



SCOAP

Sandia Controlability and Observability Analysis Program

- Basée sur une analyse structurelle
- Circuit = cellules(comb., séq.) + nœuds(comb., séq.)
- Valeurs des mesures croissantes avec l'effort de test requis

6



■ Six mesures associées à chaque nœud N :

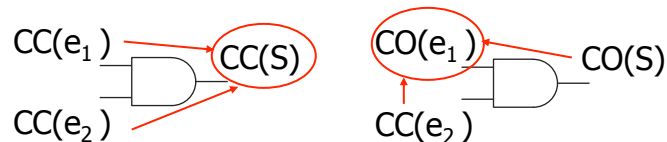
- $CC0(N)$ = contrôlabilité combinatoire à 0 = # minimum de nœuds comb. qu'il faut contrôler pour amener la valeur 0 sur N
- $CC1(N)$ = contrôlabilité combinatoire à 1 = # minimum de nœuds comb. qu'il faut contrôler pour amener la valeur 1 sur N
- CO = observabilité combinatoire = # minimum de nœuds comb. qui doivent être contrôlés pour que l'effet de la faute sur N soit propagé vers une SP.
- $SC0$ = contrôlabilité séquentielle à 0 = # minimum de nœuds séq. qu'il faut contrôler pour amener la valeur 0 sur N
- $SC1$ = contrôlabilité séquentielle à 1 = # minimum de nœuds séq. qu'il faut contrôler pour amener la valeur 1 sur N
- $S0$ = observabilité séquentielle = # minimum de nœuds séq. qui doivent être contrôlés pour que l'effet de la faute sur N soit propagé vers une SP.

7



■ Evaluation :

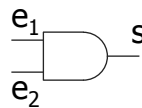
- Contrôlabilité en sortie d'une cellule fonction des contrôlabilités de ses entrées
- Observabilité en entrée d'une cellule fonction de l'observabilité en sortie et des contrôlabilités des autres entrées



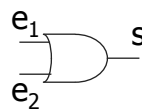
8



SCOAP - Cellules Combinatoires



- $CC0(s) = \min(CC0(e1), CC0(e2)) + 1$
- $CC1(s) = CC1(e1) + CC1(e2) + 1$
- $CO(e1) = CC1(e2) + CO(s) + 1$



- $CC0(s) = CC0(e1) + CC0(e2) + 1$
- $CC1(s) = \min(CC1(e1), CC1(e2)) + 1$
- $CO(e1) = CC0(e2) + CO(s) + 1$

9



Portes Logiques Classiques

Contrôlabilité
Combinatoire à 0

AND2	$CC0(OUT) = \min(CC0(IN1), CC0(IN2)) + 1$
NAND2	$CC0(OUT) = CC1(IN1) + CC1(IN2) + 1$
OR2	$CC0(OUT) = CC0(IN1) + CC0(IN2) + 1$
NOR2	$CC0(OUT) = \min(CC1(IN1), CC1(IN2)) + 1$
INV	$CC0(OUT) = CC1(IN) + 1$
BUF	$CC0(OUT) = CC0(IN) + 1$

Contrôlabilité
Combinatoire à 1

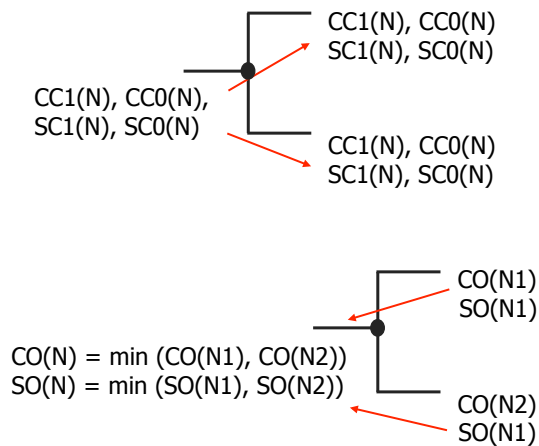
AND2	$CC1(OUT) = CC1(IN1) + CC1(IN2) + 1$
NAND2	$CC1(OUT) = \min(CC0(IN1), CC0(IN2)) + 1$
OR2	$CC1(OUT) = \min(CC1(IN1), CC1(IN2)) + 1$
NOR2	$CC1(OUT) = CC0(IN1) + CC0(IN2) + 1$
INV	$CC1(OUT) = CC0(IN) + 1$
BUF	$CC1(OUT) = CC1(IN) + 1$

Observabilité
Combinatoire

AND2	$CO(IN1) = CC1(IN2) + CO(OUT) + 1$
NAND2	$CO(IN1) = CC1(IN2) + CO(OUT) + 1$
OR2	$CO(IN1) = CC0(IN2) + CO(OUT) + 1$
NOR2	$CO(IN1) = CC0(IN2) + CO(OUT) + 1$
INV	$CO(IN1) = CO(OUT) + 1$
BUF	$CO(IN1) = CO(OUT) + 1$

10

SCOAP - Divergences



11

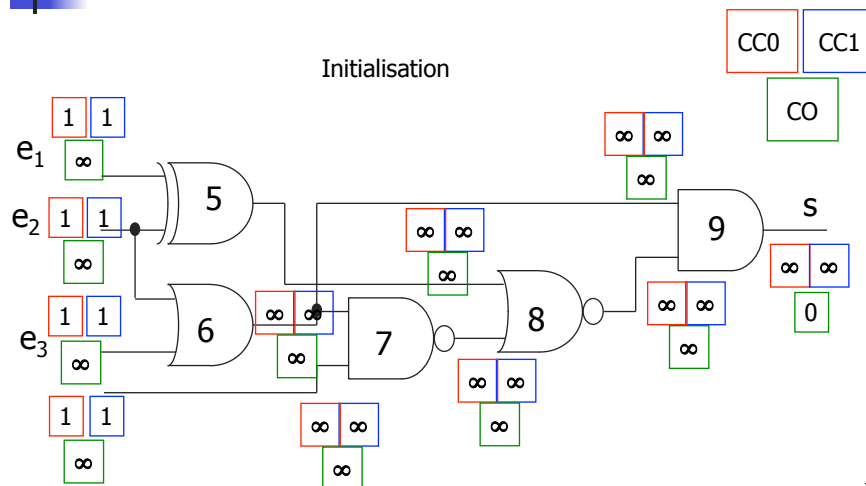
SCOAP - Calcul des Valeurs

- Initialisation des nœuds :
 - Si $N = EP \rightarrow CC0(N) = CC1(N) = 1$
 $\rightarrow SC0(N) = SC1(N) = 0$
 - Si $N = SP \rightarrow CO(N) = SO(N) = 0$
 - Sinon $\rightarrow CC0 = CC1 = SC0 = SC1 = \infty$
- Phase 1 :
 - Calcul des contrôlabilités des EPs aux SPs,
 - Itération jusqu'à stabilisation
- Phase 2 :
 - Calcul des observabilités des SPs aux EPs

12

Example

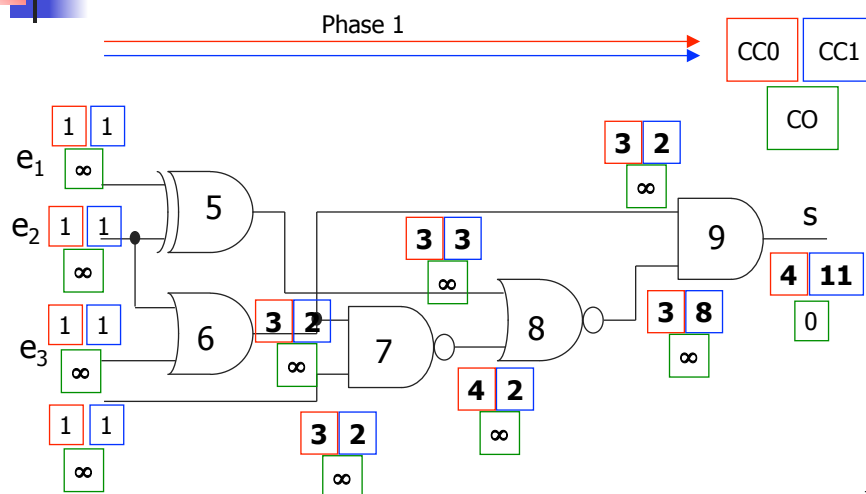
Initialisation



13

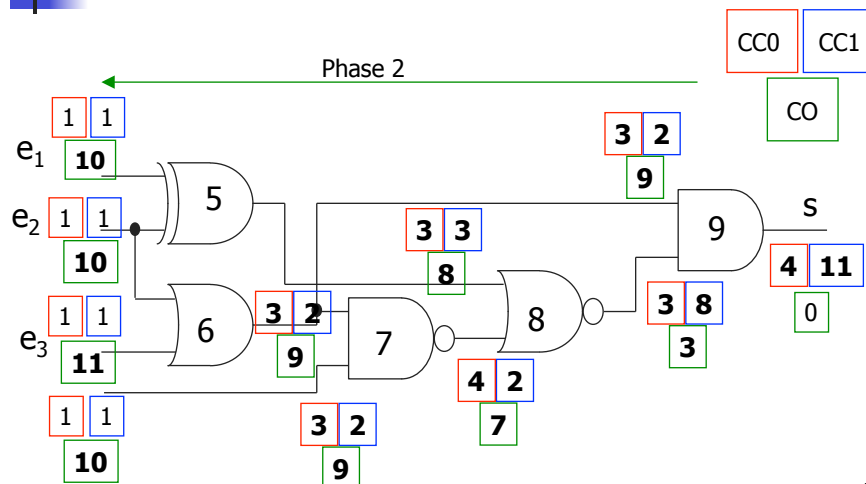
Example

Phase 1



14

Exemple



15

Les dérivés de SCOAP (1)

- COMET(Controllability and Observability Measurement for Testability)
 - Possibilité d'utiliser des macrocellules et non plus uniquement des portes logiques (\Rightarrow gain de temps / SCOAP)
 - Prise en compte de cellules bidirectionnelles
 - Mesure de 'prédictabilité' : quantifie la facilité avec laquelle le circuit peut être initialisé dans un état connu
- ITTAP
 - Insertion interactive de points de test
 - Mesure supplémentaire qui donne la longueur d'une séquence de test aléatoire pour contrôler ou observer la valeur d'un nœud

16

Les dérivés de SCOAP (2)

■ ARCOP

- Testabilité d'un nœud = difficulté de détecter un collage sur le nœud
 $TESTABILITE(N \text{ collé à } 0) = CC1(N) + CO(N)$
 $TESTABILITE(N \text{ collé à } 1) = CC0(N) + CO(N)$

■ DTA (Daisy Testability Analyser)

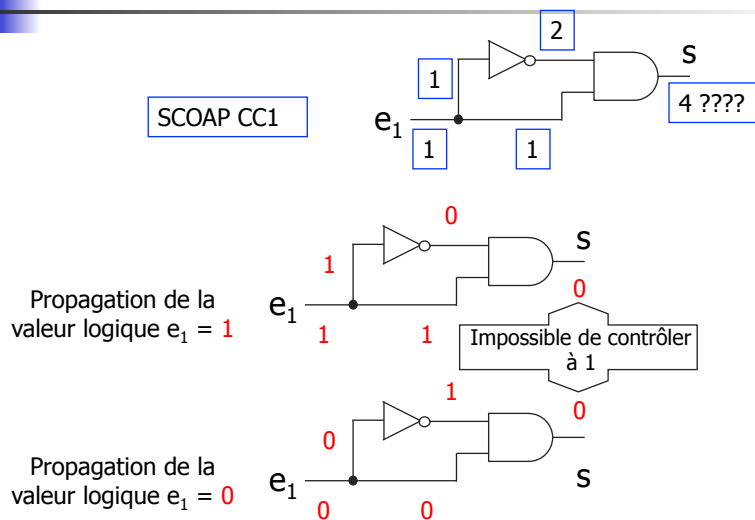
- Insertion interactive de points de test (comme ITTAP)
- possibilité d'utiliser des macrocellules et non plus uniquement des portes logiques (RAM, ROM, PLA, blocs représentés par des fonctions booléennes) \Rightarrow gain de temps / SCOAP

■ FUNTAP

- Généralisation de la méthode au niveau fonctionnel (ex : chemins de données)

17

Inconvénients de SCOAP (1)

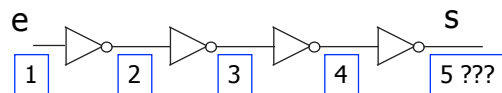


18

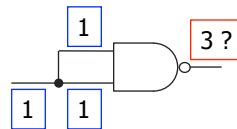
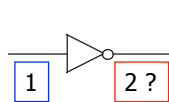
Inconvénients de SCOAP (2)

SCOAP CC1

SCOAP CC0



Pourtant il est aussi facile de contrôler s que e



La façon de réaliser une même fonction influence les résultats

19

COP

- Basée sur une Analyse structurelle
- Mesures basées sur une approche probabiliste
 - proportion de vecteurs d'entrée qui peuvent contrôler et observer le nœud
- Limitées aux circuits combinatoires
- Normalisées [0 1]

20



■ Cinq mesures associées à chaque nœud N :

- $C1(N)$ = contrôlabilité à 1 de N
= (# vecteurs d'entrée donnant 1 sur N) / 2^n (n = # bits)
- $C0(N)$ = contrôlabilité à 0 de N
= (# vecteurs d'entrée donnant 0 sur N) / 2^n
= $1 - C1(N)$
- $O(N)$ = observabilité de N
= (# de 1 et de 0 sur N observables sur 1 SP) / 2^n
- $O1(N)$ = observabilité à 1 de N
= probabilité de détecter un collage à 0 de la ligne N
= $C1(N) \cdot O(N)$
- $O0(N)$ = observabilité à 0 de N
= probabilité de détecter un collage à 1 de la ligne N
= $C0(N) \cdot O(N)$

$C1(N), O(N) \Rightarrow C0(N), O1(N), O0(N)$

21



■ Porte AND

- $C1(s) = C1(e1) \cdot C1(e2)$
- $O(e1) = C1(e2) \cdot O(s)$

■ Porte OR

- $C1(s) = 1 - (1 - C1(e1)) \cdot (1 - C1(e2))$
- $O(e1) = (1 - C1(e2)) \cdot O(s)$

■ Porte NOT

- $C1(s) = 1 - C1(e)$
- $O(e) = O(s)$

22



Calcul des Mesures

- Initialisation : $C1(EPs) = 0,5$ et $O(SP) = 1$
- Passe avant (EPs vers SPs) pour les contrôlabilités
- Passe arrière (SPs vers EPs) pour les observabilités
- Racine (B) et b branches de divergences (B_i) :

$$O(B) = 1 - \prod_{i=1}^b (1 - O(B_i))$$

23



Conclusions

- Mesures présentées :
 - Hypothèse d'indépendance entre les aspects contrôlabilité et observabilité
- Observabilité et Contrôlabilité : mauvaise image de la testabilité
 - qui est induite par le recouvrement entre l'ensemble des vecteurs d'entrée assurant la contrôlabilité d'une panne et l'ensemble des vecteurs d'entrée assurant l'observabilité de cette même panne
- Erreur d'interprétation possible MAIS néanmoins largement utilisé dans l'industrie
 - Aide à la génération de vecteurs de test (prise de décision)
 - Aide à la conception en vue du test (insertion de points de test)

24



Chapitre 5

ATPG

Arnaud Virazel

virazel@lirmm.fr

1



Génération de vecteurs de test

- Généralités
- Test Aléatoire
- Test Déterministe
 - Gestion des fautes
 - Principes et Concepts élémentaires
 - Génération au niveau structurel
 - Circuit combinatoire (D-algorithme, PODEM)
 - Circuit séquentiel (Chapitre 6)
 - Génération au niveau fonctionnel
 - Mémoire (Chapitre 8)

2



Généralités

- Hypothèses (dans ce chapitre) :
 - Test appliqué hors du fonctionnement normal du circuit (off-line)
 - Vecteurs mémorisés dans un testeur qui se charge d'appliquer la séquence au circuit (test externe)
 - Résultats par comparaison complète des réponses du circuit avec les réponses attendues

3



Généralités

- Principaux aspects :
 - Le coût de la Génération de Vecteurs de Test
 - La qualité du test généré
 - Le coût d'application du test généré
- Evaluation (dans le contexte d'un modèle de fautes) :
 - Taux de couverture T_c

4



Taux de couverture : Tc

- Taux de couverture de fautes

$$Tc = \frac{\text{\# de fautes détectées par la séquence de test}}{\text{\# de fautes total (dans le modèle)}}$$

- Connu par simulation de faute
- Utilisé dans l'industrie

- Efficacité

$$Eff = \frac{\text{\# de fautes détectées par la séquence de test}}{\text{\# de fautes testables}}$$

5



Test aléatoire

- Vecteurs de test générés de façon aléatoire
- Coût du développement peu élevé

6



Test aléatoire

- Qualité du test :
 - Dépend de la longueur de la séquence
 - Simple obtenir $T_c \approx 60\% - 80\%$
 - Séquence longue pour un T_c raisonnable ($\approx 95\%$)
 - Séquence très longue pour un T_c élevé ($> 98\%$)
- Coût d'application élevé car séquences longues :
 - Temps d'application et mémoire nécessaire élevés

7



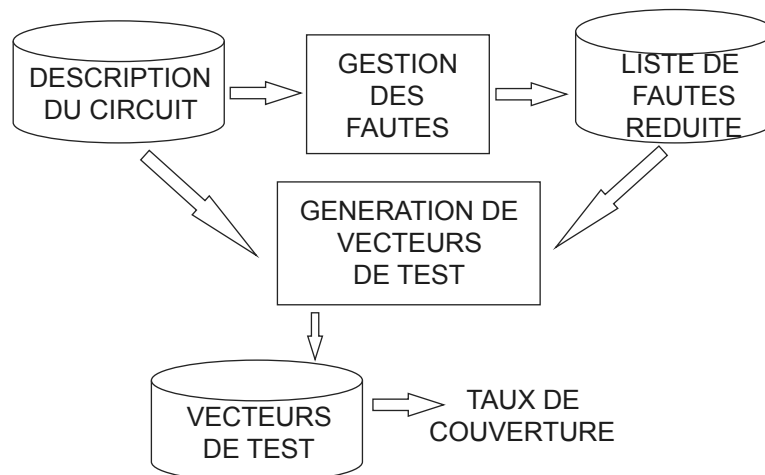
Test Déterministe

- Il peut être manuel ou automatique
- En général, appliqué à des sous ensembles fonctionnels du circuit (approche boule de neige «start-small», «diviser pour régner»)
- Basé le plus souvent sur le modèle de collage

8

Génération Automatique de vecteurs de test

■ ATPG : **A**utomatic **T**est **P**attern **G**eneration



9

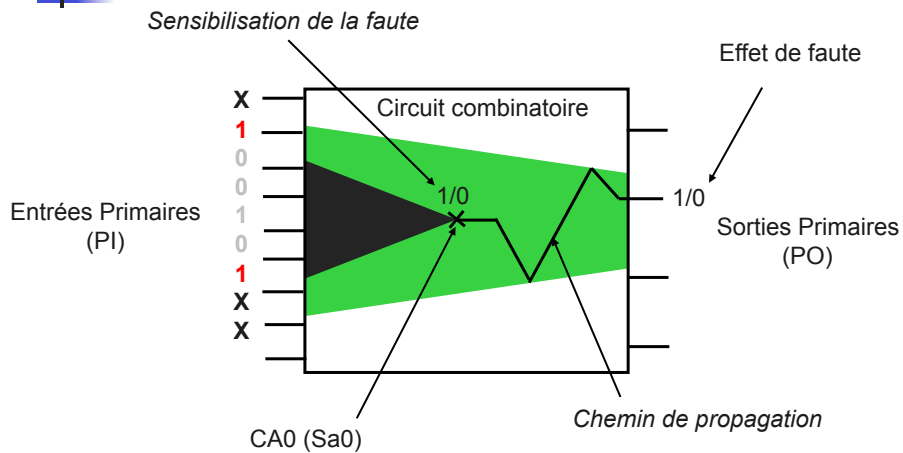
Génération au niveau structurel

■ Test des circuits combinatoires :

- Idée de base : sensibilisation de chemins
- Technique :
 - Choix d'un chemin sensible de l'origine de la panne jusqu'à une sortie primaire = phase de propagation aval ("*forward-trace phase*")
 - Déterminer un vecteur d'entrée qui satisfasse toutes les assignations effectuées dans la phase précédente = phase de propagation amont ("*backward-trace phase*")

10

Génération au niveau structurel



11

Le D-algorithme : Principe

■ Principe de base

1. définir le test d'une faute f en termes d'E/S de la porte fautive
2. déterminer tous les chemins sensibilisables du site de la faute à toutes les SPs du circuit (*propagation aval*)
3. construire le vecteur de test sur les EPs qui réalise toutes les assignations effectuées en 1/ et 2/ (*propagation amont*)

12



Le D-algorithme : Principe

- Algèbre à 5 valeurs
 - 0, 1, X, D (1/0), \bar{D} (0/1)
 - D → ligne = 1 dans le circuit sain, 0 dans le circuit fautif
 - D → ligne = 0 dans le circuit sain, 1 dans le circuit fautif
 - Dans ce cours \bar{D} est parfois noté D^*
- 3 notions
 - D-cube primitif
 - Couverture singulière
 - D-cube de propagation

13



Le D-algorithme

- D-cube primitif d'une faute affectant une porte
 - Permet de mettre en évidence une faute en sortie d'une porte sous forme de D (\bar{D}) en appliquant certaines valeurs sur les entrées

e1	e2	S
1	1	D

D-cube primitif du CA0
sur la sortie d'une porte AND

14

Le D-algorithme

■ Couverture singulière d'une porte

- Table de vérité réduite utilisant les valeurs (0, 1, X)

e1	e2	S
0	x	0
x	0	0
1	1	1

Couverture singulière
d'une porte AND à 2 entrées

15

Le D-algorithme

■ D-cube de propagation d'une porte

- Spécifie les valeurs à appliquer sur les entrées excepté une (ou plus) tel qu'un changement sur cette entrée (ou ces entrées) induit un changement en sortie de la porte.
- Propage $D(\bar{D})$ de la (les) entrée(s) vers la sortie

e1	e2	S
D	1	D cube simple
1	D	D "
D	D	D cube multiple
\bar{D}	1	\bar{D} cube simple
1	\bar{D}	\bar{D} "
\bar{D}	\bar{D}	\bar{D} cube multiple

D-cube de
propagation
AND à 2 entrées

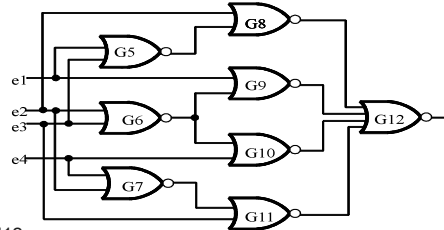
16

D-algorithme : exemple

- 0/ Construire le tableau des D-cubes de propagation du circuit

	Lignes											
	e1	e2	e3	e4	I5	I6	I7	I8	I9	I10	I11	I12
a	0	x	D	x	D*	x	x	x	x	x	x	x
b	D	x	0	x	D*	x	x	x	x	x	x	x
c	x	0	D	x	x	D*	x	x	x	x	x	x
d	x	D	0	x	x	D*	x	x	x	x	x	x
e	x	0	x	D	x	x	D*	x	x	x	x	x
f	x	D	x	0	x	x	D*	x	x	x	x	x
g	x	0	x	x	D	x	x	D*	x	x	x	x
h	x	D	x	x	0	x	x	D*	x	x	x	x
9	0	x	x	x	x	D	x	x	D*	x	x	x

.....
Tableau des D-cubes de propagation du circuit



17

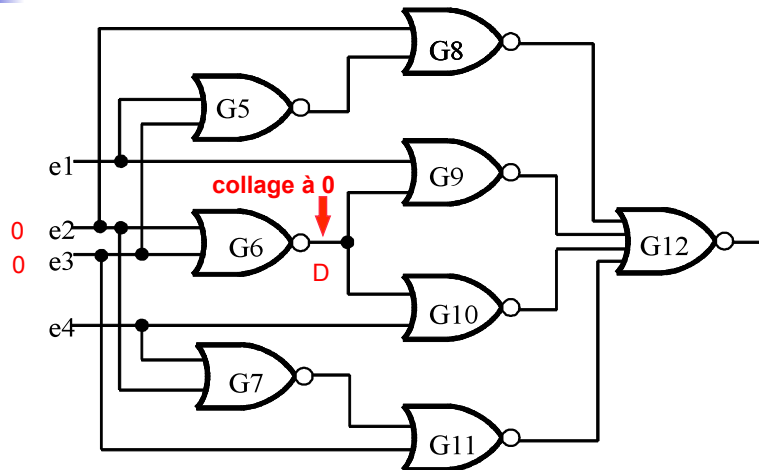
D-algorithme : exemple

- 1/ Choisir un D-cube primitif (C0) pour la faute considérée (Sa0 - I6):

e1	e2	e3	e4	I5	I6	I7	I8	I9	I10	I11	I12
x	0	0	x	x	D	x	x	x	x	x	x

18

D-algorithme : exemple



19

D-algorithme : exemple

- 2/ Propagation de D jusqu'à la sortie l12 (phase de propagation aval)

Cube	D-intersection												Porte activée	Porte à activer	
	e1	e2	e3	e4	l5	l6	l7	l8	l9	l10	l11	l12			
C0	0	0				D							G6	G9,G10	I
C1=C0 ∩ (9)	0	0	0			D		D*					G9	G12	II
C2=C0 ∩ (10)		0	0	0		D				D*			G10	G12	
C3=C1 ∩ (12)		0	0	0		D		0	D*0	0	D		G12	∅	
C4=C2 ∩ (12)			0	0	0	D		0	0	D*0	D		G12	∅	
C5=C0 ∩ (9) ∩ (10)		0	0	0	0	D				D* D*			G9,G10	G12	III
C6=C5 ∩ (12)			0	0	0	0	D		0	D* D*0	D		G12	∅	

I : Cube initial (C0) pour la faute de sortie de G6 collée à 0

II : 2 Chemins simples (D-cube simple) conduisant au cubes C3 et C4

III : 1 Chemin multiple (D-cube multiple) conduisant au cube C6

20

D-algorithme : exemple

- Règles d'intersection des D-cubes de propagation

\cap	0	1	x	D	D*
0	0	-	0	-	-
1	-	1	1	-	-
x	0	1	x	D	D*
D	-	-	D	D	-
D*	-	-	D*	-	D*

⇒ conflit

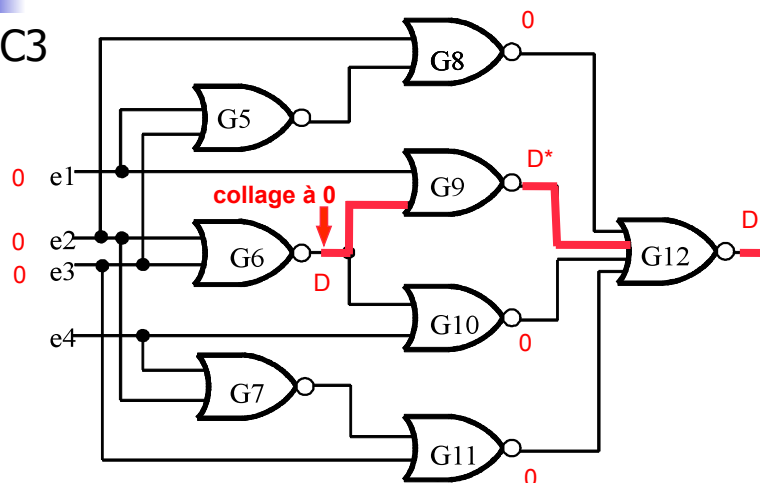
Opération de
D-intersection (\cap)

	e1	e2	e3	e4	I5	I6	I7	I8	I9	I10	I11	I12
C0	x	0	0	x	x	D	x	x	x	x	x	x
(9)	0	x	x	x	x	D	x	x	D*	x	x	x
\cap	0	0	0	x	x	D	x	x	D*	x	x	x

21

D-algorithme : exemple

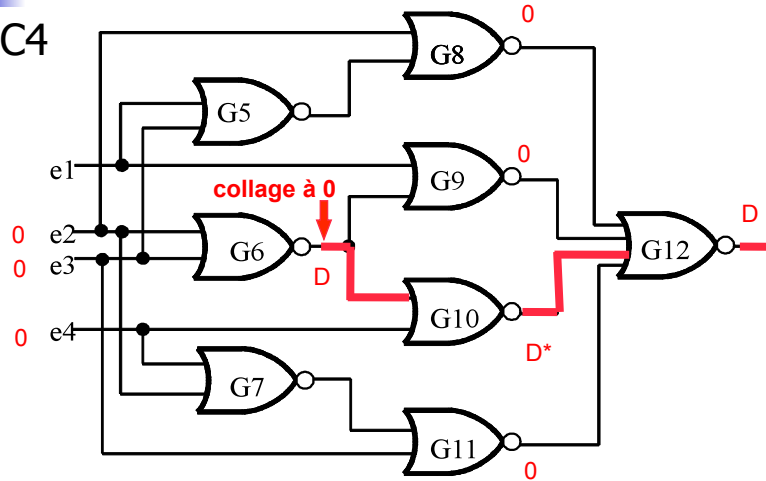
- C3



22

D-algorithme : exemple

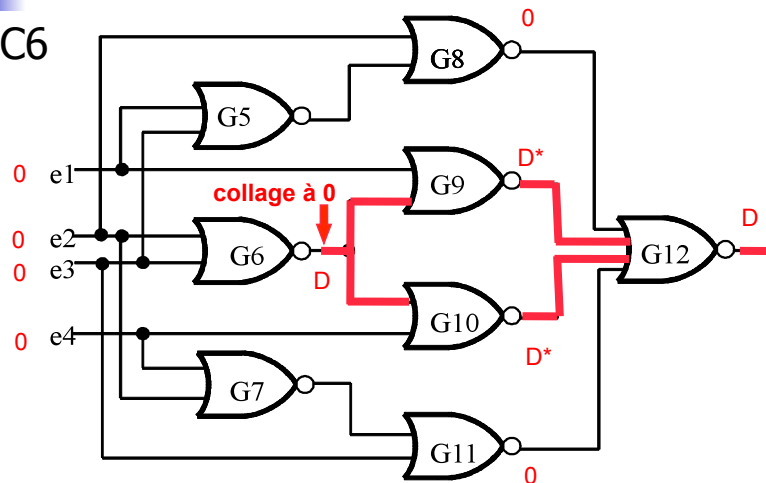
■ C4



23

D-algorithme : exemple

■ C6



24

D-algorithme : exemple

- 3/ construire le vecteur de test sur les EPs qui réalise toutes les assignations effectuées en 1/ et 2/ (phase de propagation amont)

25

D-algorithme : exemple

3/ Phase de justification du cube final C3 :
cube \cap couverture singulière

Cube	D-intersection											
	e1	e2	e3	e4	l5	l6	l7	l8	l9	l10	l11	l12
C3	0	0	0	x	x	D	x	0	D*	0	0	D
P11(a) couverture singulière NOR G11	x	x	x	x	x	1	x	x	x	0	x	
P11(b) couverture singulière NOR G11	x	x	1	x	x	x	x	x	x	0	x	
C3 \cap P11(a)	0	0	0			D	1	0	D*	0	0	D
P10(a) NOR G10				1		x				0		
P10(b) NOR G10				x		1				0		
C3 \cap P11(a) \cap P10(a)	0	0	0	1		D	1	0	D*	0	0	D
P8(a) NOR G8	1				x				0			
P8(b) NOR G8		x			1			0				
C3 \cap P11(a) \cap P10(a) \cap P8(b)	0	0	0	1	1	D	1	0	D*	0	0	D
P7 NOR G7	0		0				1					

0 Valeur à justifier
↔ Incompatibilité

Incompatibilité sur l'EP e4
 Impossible de tester le CA0/l6 par le chemin simple de propagation correspondant à C3 (G9, G12)

26

D-algorithme : Exemple

3/ Phase de justification du cube final C6 :
cube \cap couverture singulière

Cube	D-intersection											
	1	2	3	4	5	6	7	8	9	10	11	12
C6	0	0	0	0		D		0	D	D	0	D
P11(a)		1						x			0	
P11(b)		x					1				0	
C6 \cap P11(b)	0	0	0	0		D	1	0	D	D	0	D
P8(a)		1						x			0	
P8(b)		x			1						0	
C6 \cap P11(b) \cap P8(b)	0	0	0	0	1	D	1	0	D	D	0	D
P7				0			1					
C6 \cap P11(b) \cap P8(b) \cap P7	0	0	0	0	1	D	1	0	D	D	0	D
P5			0			1						

Problèmes avec le D-algorithme

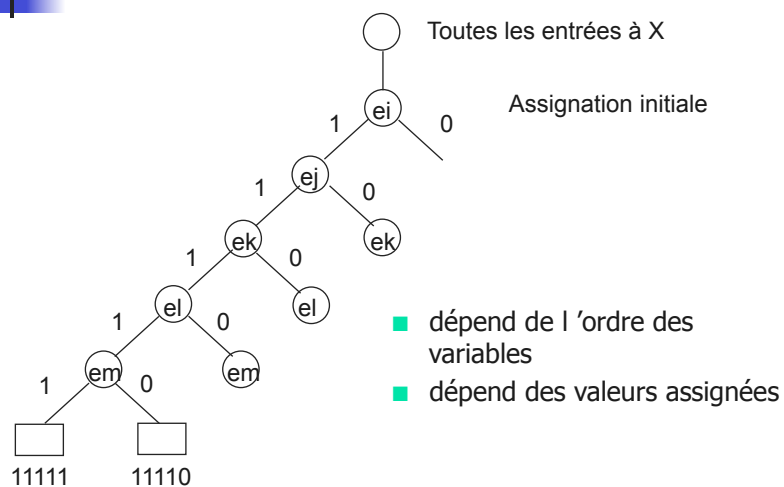
- Temps d'exécution trop important
(beaucoup de chemins + fautes redondantes)
- Amélioration des temps de calcul
 - 9V- algorithme (même stratégie mais sensibilisation simultanée de tous les chemins de propagation)
 - Changement de stratégie avec PODEM
(parcours dans un arbre)

PODEM : Path Oriented Decision Making

- Le test des circuits combinatoires est vu comme un PB de parcours dans un arbre
- Fondement : algorithme d'énumération dans lequel toutes les combinaisons d'entrée sont implicitement (mais exhaustivement) examinées comme vecteur de test d'une faute donnée
- La structure du circuit est utilisée pour guider les essais successifs des combinaisons d'entrée

29

Principe de l'algorithme d'énumération



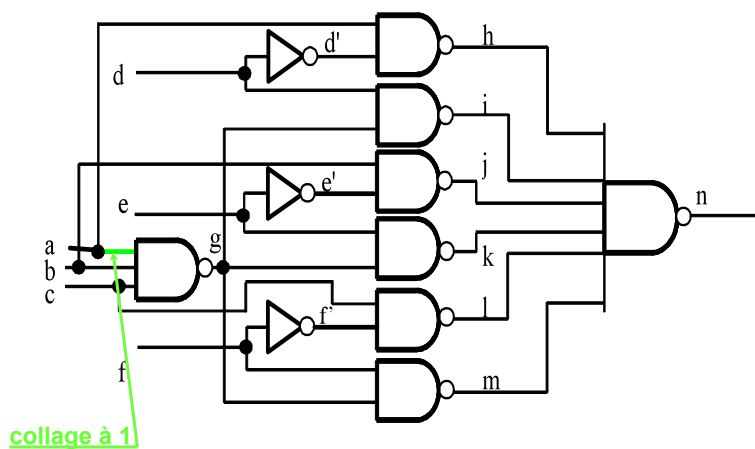
30

Algorithme Général

1. détermination de l'objectif à atteindre
 - propagation du D vers les SPs
2. «remontée» de l'objectif vers les EPs (plus simple que D-algorithme car utilisation d'heuristiques)
 - Si justification des entrées d'une porte dont la sortie est à une valeur prioritaire, choisir l'entrée la plus facilement contrôlable à la valeur prioritaire = parmi plusieurs solutions, choix de celle qui a le plus de chance d'être satisfaite
 - Si justification des entrées d'une porte dont la sortie est à une valeur non prioritaire, choisir l'entrée la plus difficile à contrôler à la valeur non prioritaire = parmi plusieurs problèmes à résoudre choix de celui réputé comme le plus difficile (afin d'éviter un échec global après la résolution de problèmes plus faciles)
3. Implications des assignations faites en 2
4. si test trouvé sortir
5. sinon
 - 5.1. si objectif atteint alors modification de l'objectif retour en 1
 - 5.2. sinon
 - si test encore possible alors retour en 2 pour nouvelle assignation
 - sinon retour en 1 pour modification de l'objectif

31

Exemple



32

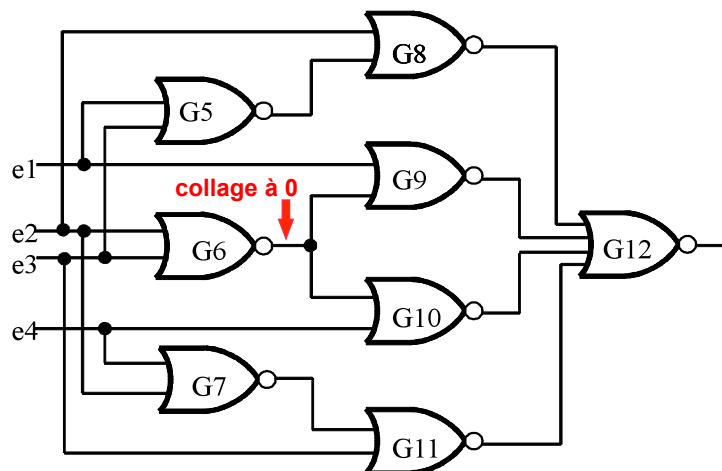
Exemple

Objectif	Assignation des EP	Implications	D-frontière	Commentaires
a=0	a=0	h=1	g	
b=1	b=1		g	
c=1	c=1	g=D	i, k, m	
d=1	d=1	d'=0 i=D*	k, m, n	
j=1	e=1	e'=0 k=D*	m, n	
l=1	f=1	f=0 m=D* n=D		
				Succès

Remarque : Limite de propagation de la valeur fautive D = D-frontière

33

PODEM : Exercice



34



- C'est une amélioration de PODEM sur un certain nombre d'heuristiques relatives aux divergences (Fan out) d'où son nom
- Différences majeures :
 - Arrêt des justifications des valeurs sur les sommets de cônes (nœud sans divergence en amont) car leur justification sera toujours possible => gain de temps si échec sur justification de valeurs sur d'autres nœuds
 - Propagation amont multiple (sur tous les chemins possibles) => pied de divergence se voit affecté comme objectif celui le plus souvent remonté à travers les branches de divergence
 - Sensibilisation obligée => assignation des valeurs nécessaires de propagation sur toutes les portes "obligées" c'est à dire faisant nécessairement partie du chemin de propagation

35



Chapitre 6

Test de Circuits Séquentiels

Arnaud Virazel

virazel@lirmm.fr

1



Test de circuits séquentiels :

Les difficultés

- Le test d'une faute nécessite en général plusieurs vecteurs
 - Problèmes structurels
 - les cycles
 - la profondeur séquentielle
 - les chemins reconvergers
 - la sortance des bascules
- } Idem combinatoire

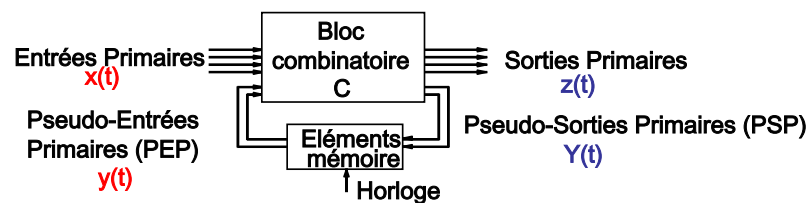
2

Test de circuits séquentiels : Les difficultés

- Problèmes comportementaux
 - la densité d'états valides
 - la fonctionnalité

3

Modélisation du circuit



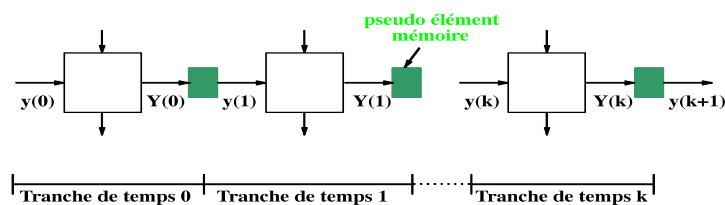
4

Modélisation du circuit

- Représentation par tranche de temps :
 - Le circuit est vu comme une succession de circuits (combinatoires) dans le temps
 - Les éléments de mémorisation du circuit original sont modélisés comme des **pseudo-éléments combinatoires** de mémorisation qui font le lien entre les différentes copies temporelles de la partie combinatoire

5

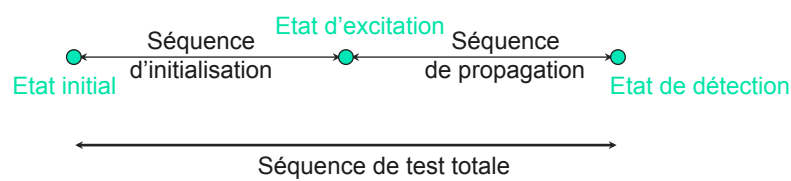
Modélisation du circuit



6

Séquence de test

- Nombre de vecteurs de la séquence égal au nombre de tranche de temps nécessaires



7

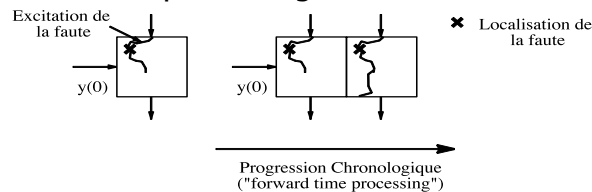
Génération automatique

- Les générateurs basés sur l'analyse topologique du circuit
- Les générateurs utilisant la simulation du circuit

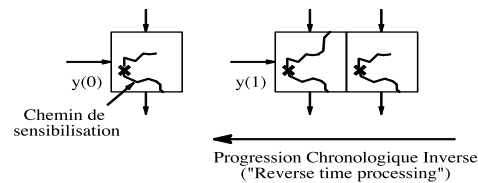
8

Analyse topologique du circuit

■ «Forward time processing»



■ «Backward time processing»



Analyse topologique du circuit

- En général utilisation couplée des techniques de «forward» et «backward time processing»
- Utilisation du « D algorithme » pour la partie combinatoire
- Procédures spécifiques pour améliorer :
 - l'étape de justification (justification d'états)
 - l'étape de sensibilisation



Les générateurs basés sur la simulation de fautes

- Choix d'un vecteur d'essai et simulation
- Des fonctions de coût évaluent si le vecteur est intéressant pour la détection ou pas (nbre de bascules non spécifiées, nbre de portes restant à traverser pour l'erreur, ...)



Les générateurs basés sur la simulation de fautes

- Eventuellement modification graduelle du vecteur pour améliorer les fonctions de coût
- Permettent de travailler sur des circuits plus «gros»
- Les algorithmes génétiques appartiennent à cette famille
- En général couplé avec des algorithmes topologiques



Génération au niveau fonctionnel

- Vérifier la fonctionnalité pour laquelle le circuit a été conçu (Test fonctionnel)
- Parfois la seule solution envisageable si l'on ne connaît pas la structure interne du circuit
- Très difficile de garantir un taux de couverture



Génération au niveau fonctionnel

- Ne permet pas de détecter certaines défaillances, en particulier lorsque les fonctionnalités ne sont pas vérifiées avec tout l'espace des données manipulées
- Mais il existe des méthodes de test fonctionnel mieux adaptées que le test déterministe lorsque les structures sont très régulières (Mémoires)



Chapitre 7

Test de Fautes de Délais

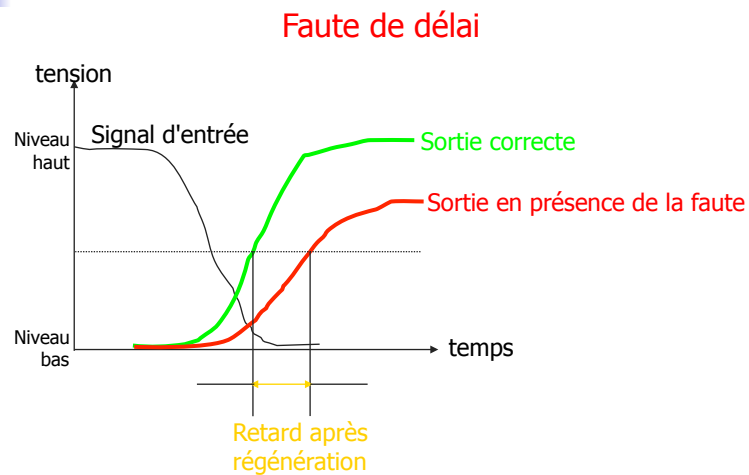
Arnaud Virazel
virazel@lirmm.fr



Plan

- Modélisation
- Le test des fautes de délais
- Le Scan et les fautes de délais

Principe



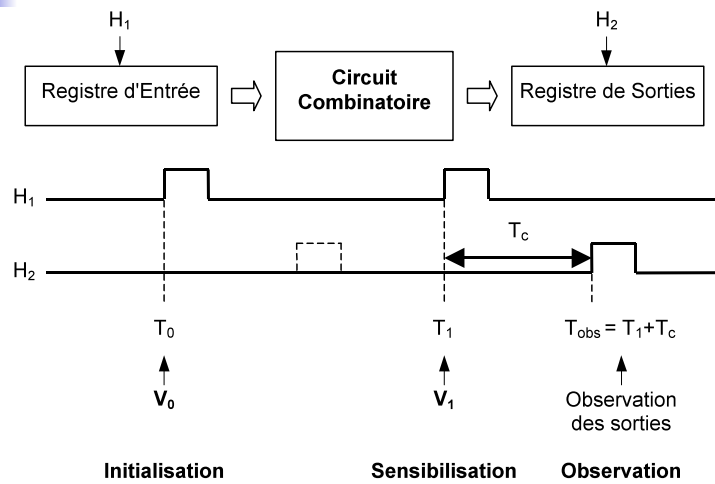
3

Définition

- Du fait de structures plus complexes et de vitesse d'opération plus élevées, les fautes de délais prennent de plus en plus d'importance dans les nouvelles technologies
- Lorsqu'un circuit fonctionne à une certaine fréquence et qu'il présente un dysfonctionnement à fréquence plus élevée (fréquence toujours permise par les spécifications), on dit qu'il est le siège d'une faute de délai
- Le test d'une faute de délai nécessite **l'application de deux vecteurs afin de provoquer une transition** au site de la faute et de propager l'erreur vers les sorties primaires

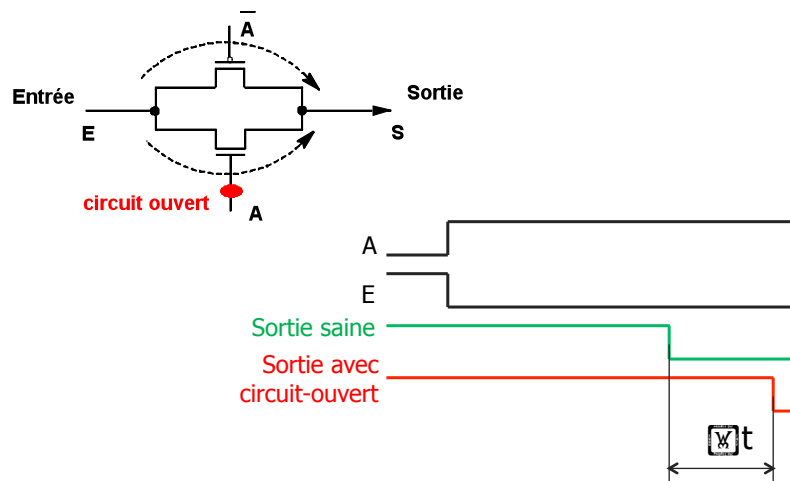
4

Principe du Test



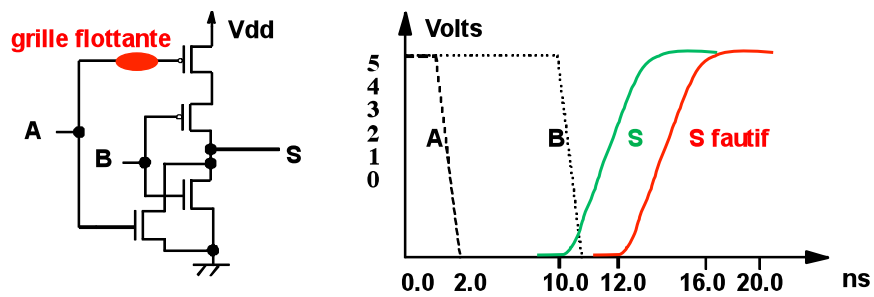
5

Origines Physiques



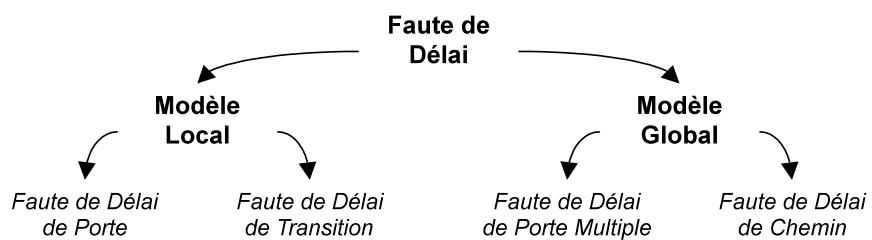
6

Origines Physiques



7

Classification



8



Faute de Transition

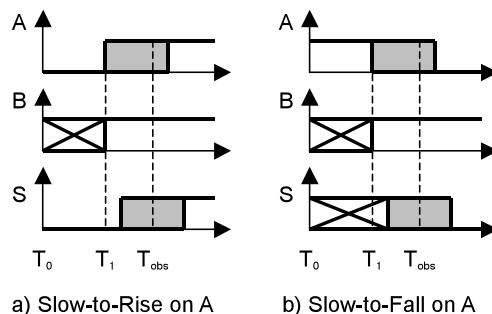
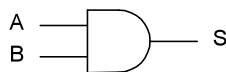
- Le retard induit par la panne est supposé être suffisamment long pour provoquer une erreur quel que soit le chemin utilisé pour la propager
- Le nombre de fautes possibles dans le circuit reste limité au nombre de connexions du circuit voire le double si l'on considère les pannes "lent-à-monter" et "lent-à-descendre "
- L'inconvénient majeur est que seul l'aspect qualitatif des fautes est pris en compte, ce qui limite son utilisation au cas particulier des fautes de délai de grande taille
- Le modèle de faute de transition est très souvent utilisé dans les applications industrielles car il est simple à manipuler et, de plus, il est très proche du modèle de faute de collage d'un point de vue génération de vecteurs de test

9



Faute de Transition

- Test des fautes de transition sur une porte AND



10



Faute de Délai de Chemin

- La faute de délai peut être répartie sur tout le chemin (prise en compte de défauts non locaux tels que dérives, désalignement, ..)
- La réponse de la sortie est observée à l'intervalle de fonctionnement nominal
- Le modèle de chemin est efficace avec des fautes de n'importe quelle taille
- la présence d'autres fautes n'affectent (théoriquement) pas la possibilité de tester un chemin donné. Ceci permet donc de gérer les fautes multiple.
- Le modèle de chemin est réaliste et parfaitement efficace si **tous** les chemins sont testés

11



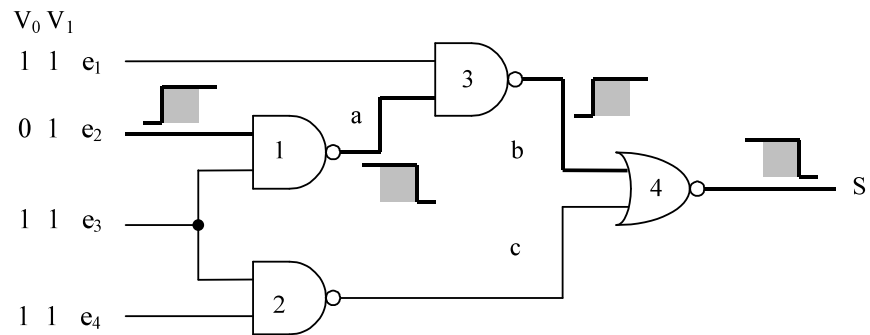
Faute de Délai de Chemin

- Le nombre de chemins structurels dans un circuit réel est en général important
- Le nombre de chemins fonctionnels est en général plus faible mais reste important
- Nécessité de choisir un sous-ensemble plus réduit
- Par exemple les chemins structurels les plus longs
- **ATTENTION** s'il existe une faute non sensibilisée par le sous-ensemble choisi, elle ne sera jamais détectée

Circuit	Nombre total de chemins structurels	Nombre de plus longs chemins structurels
C432	83926	27
C499	9440	18
C880	8642	35
C1355	4173216	41
C1908	729057	48
C2670	679884	44
C3540	28676671	58
C5315	1341305	56
C6288	>10 ²⁰	218
C7552	726493	49

12

Faute de Délai de Chemin



13

Plan

- Modélisation
- Le test des fautes de délais
- **Le scan et les fautes de délais**

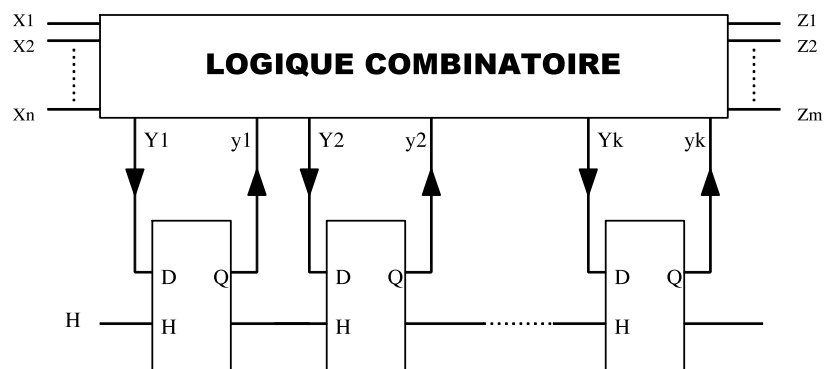
14

Modification du schéma d'horloge

- LOC - Launch-Off-Capture
 - V1 → Vecteur chargé en série
 - V2 → Réponse de V1
- LOS – Launch-Off-Shift
 - V1 → Vecteur chargé en série
 - V2 → Décalage de V1

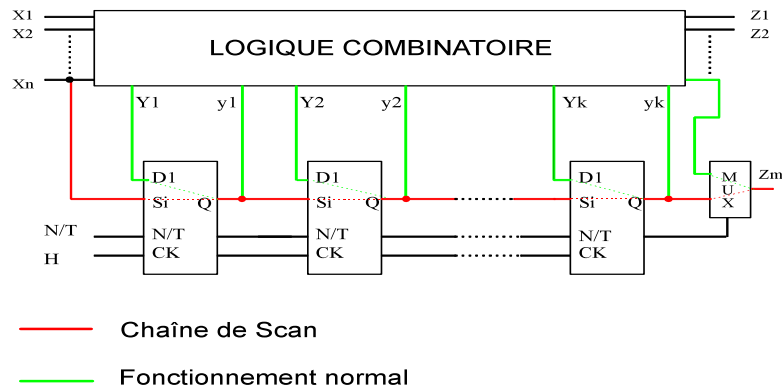
15

Le «Scan Path» *La structure de départ*



16

Le «Scan Path» La nouvelle structure



17

Le «Scan Path» La Procédure de Test

0) Test de la chaîne de scan (séquence de k fois 01)

- 1) Positionner le circuit en mode test ($N/T=1$),
- 2) Entrer par décalage le vecteur de test $\{y_1, \dots, y_k\}$ à l'intérieur des bascules,
- 3) Positionner les valeurs de test correspondantes sur les entrées primaires X_i ,
- 4) Positionner l'entrée (N/T) à la valeur logique zéro et après un temps nécessaire à la stabilisation des sorties de la partie combinatoire vérifier les différentes sorties Z_k ,
- 5) Appliquer une impulsion d'horloge sur l'entrée H ,
- 6) Positionner l'entrée (N/T) à la valeur logique 1 et sortir en série le contenu du registre à décalage (des bascules) par l'intermédiaire de la sortie Z_m et le comparer avec les résultats attendus (simultanément à la sortie du vecteur d'observation (contenu des bascules), on peut entrer le nouveau vecteur de test par l'intermédiaire de l'entrée X_n).

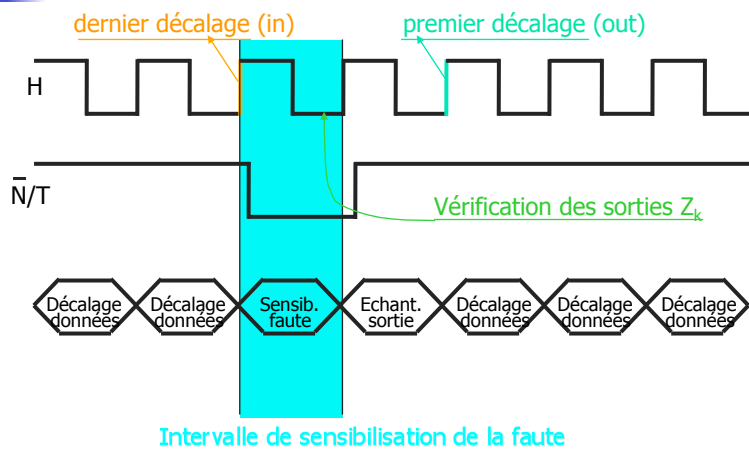
Rq1 : simultanément à la sortie du vecteur d'observation (contenu des bascules), on peut entrer le nouveau vecteur de test par l'intermédiaire de l'entrée X_n).

Rq2 : la technique de scan transforme le circuit séquentiel en circuit combinatoire durant le mode test

18

Le «Scan Path»

Le timing



19

Le Scan et le Test de Délai

■ Problématique

- Application d'une paire de vecteur ($V1$, $V2$)
- $V1$ et $V2$ sont indépendants

■ Solutions

- Modification de la chaîne de Scan
- Modification du schéma d'horloge

20



Modification de la chaîne de scan

- Doubler la taille de la chaîne de scan
- Entrelacer V1 et V2 avant le chargement
- Charger l'ensemble V1 V2
- Un décalage permet d'appliquer V2

- Test équivalent au test de délai de la partie combinatoire
- Trop coûteux en surface

21



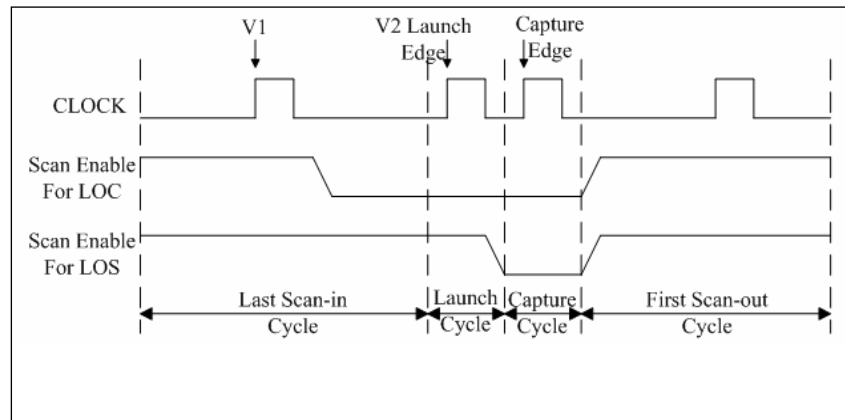
Modification du schéma d'horloge

- LOC - Launch-Off-Capture
 - V1 → Vecteur chargé en série
 - V2 → Réponse de V1

- LOS – Launch-Off-Shift
 - V1 → Vecteur chargé en série
 - V2 → Décalage de V1

22

LOC et LOS



23

Conclusion

- Test des fautes de transition
 - LOC très utilisé
 - LOS plus efficace
- Test des fautes de délai de chemin
 - Un ensemble de chemin critique
 - Un ensemble de chemin couvrant un maximum de chemin

24



Chapitre 8

Test de Mémoires

Arnaud Virazel

virazel@lirmm.fr

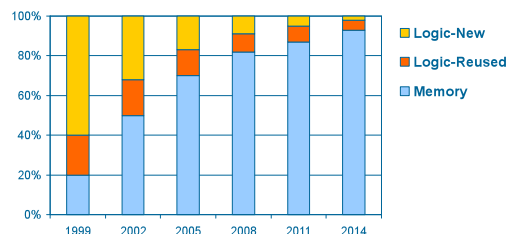


Plan

- **Introduction**
- Modélisation de la mémoire
- Mécanismes de défaillances et modélisation de fautes
- Algorithmes de test pour RAM
- Test des ROMs
- BIST mémoire

Introduction

- Les mémoires (particulièrement les RAM) sont au premier rang de la conception des circuits commerciaux
- Les DRAMs sont les moteurs du développement technologique dans l'industrie des semi-conducteurs
- Les mémoires sont les coeurs les plus utilisés dans les SoC



3

Principaux types de mémoires vives RAMs

- RAM dynamique (DRAM)
 - La meilleure densité
 - Temps d'accès lent (typiquement 20ns)
 - L'information est stockée comme la charge d'un condensateur et doit être rafraîchie régulièrement
- RAM statique (SRAM)
 - Les plus rapides (typiquement 2ns)
 - L'information est stockée dans des latches réalisés avec des inverseurs rebouclés

4



Principaux types de mémoires mortes ROMs

- Les mémoires mortes (ROM)
 - L'information est stockée par la présence ou l'absence d'un transistor lors de la fabrication
 - L'information persiste même en cas de non alimentation du circuit
- Les mémoires mortes effaçables et programmables (EPROM)
 - Programmables dans l'application
 - Effaçables entièrement en appliquant des rayons ultraviolets
- Les mémoires mortes effaçables électriquement et programmables (EEPROM, FLASH)
 - Les mots peuvent être effacés sélectivement par des moyens électriques

5



Le Test de Mémoires

- Le test de mémoires doit prouver que le circuit sous test se comporte comme il a été conçu, il consiste donc en :
 - des test paramétriques relatifs au niveau de courants/tensions et aux délais sur les broches d'E/S du circuit
 - un test fonctionnel
 - Modélisation de fautes fonctionnelles

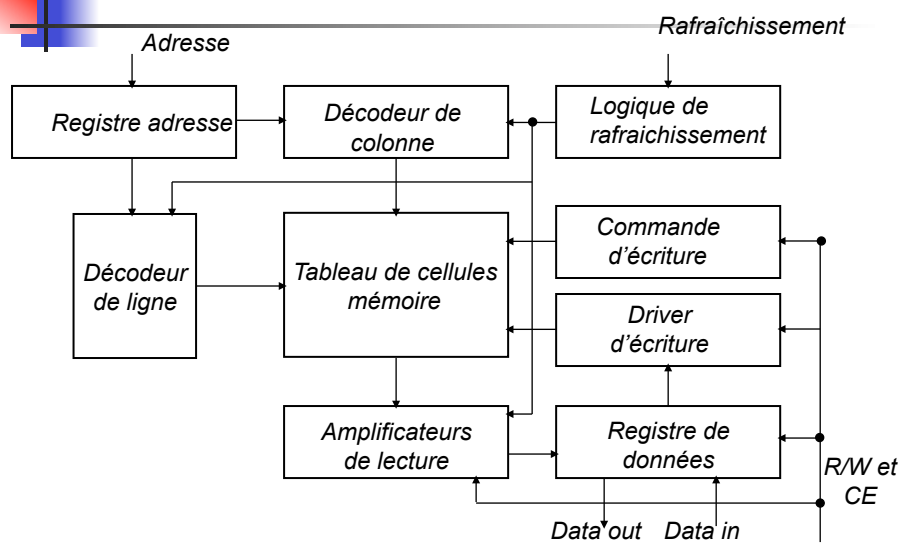
6

Plan

- Introduction
- **Modélisation de la mémoire**
- Mécanismes de défaillances et modélisation de fautes
- Algorithmes de test pour RAM
- Test des ROMs
- BIST mémoire

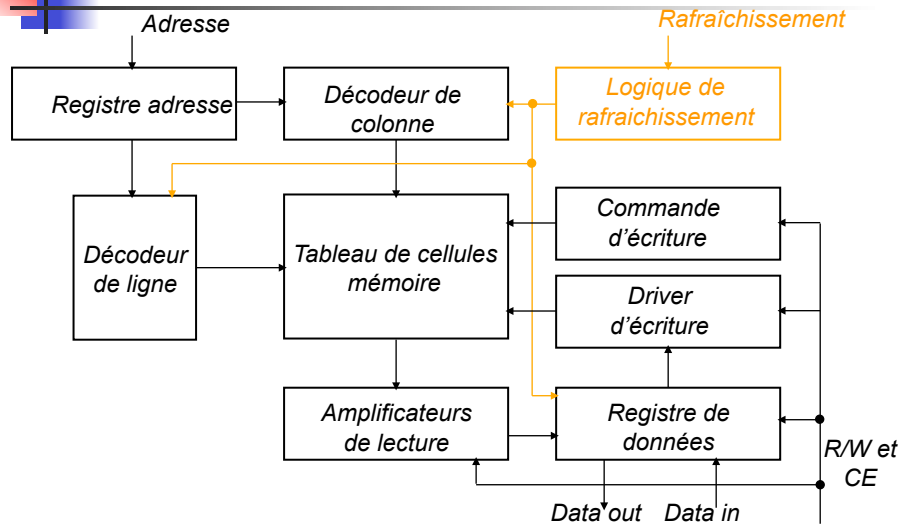
7

Modèle fonctionnel de RAM

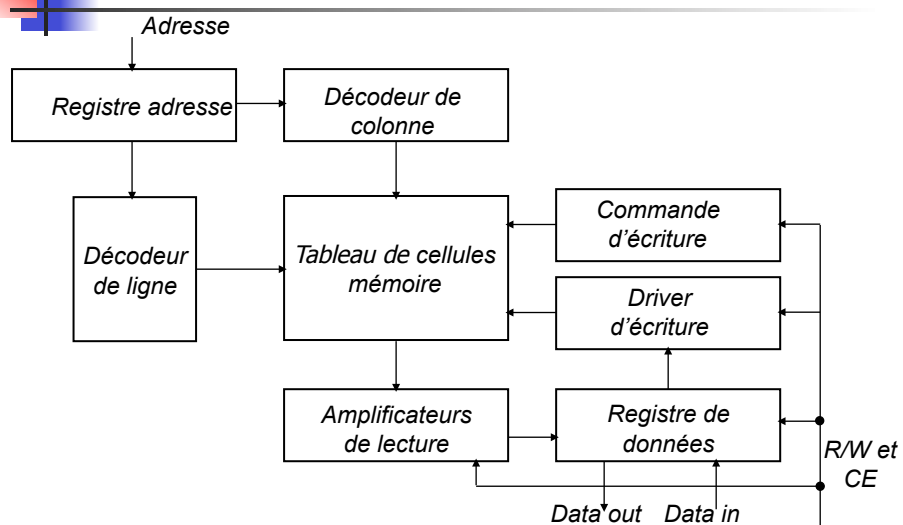




Modèle fonctionnel de SRAM

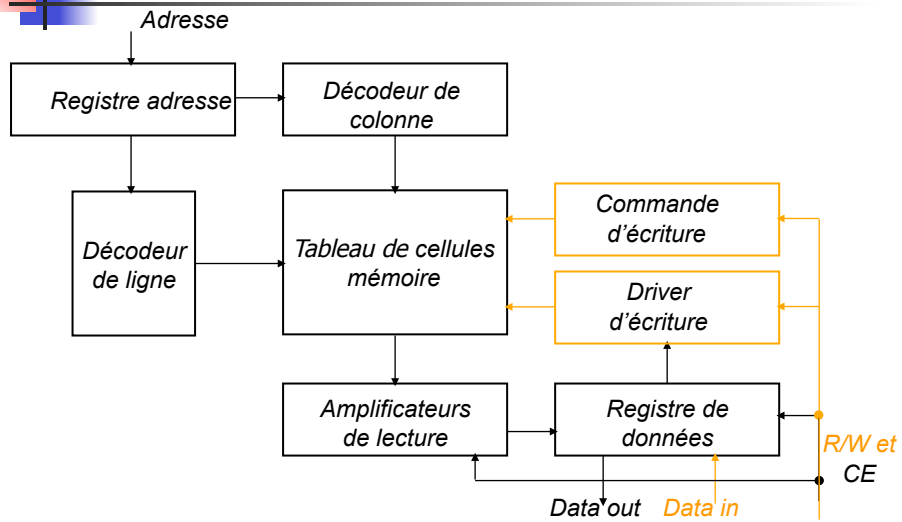


Modèle fonctionnel de SRAM

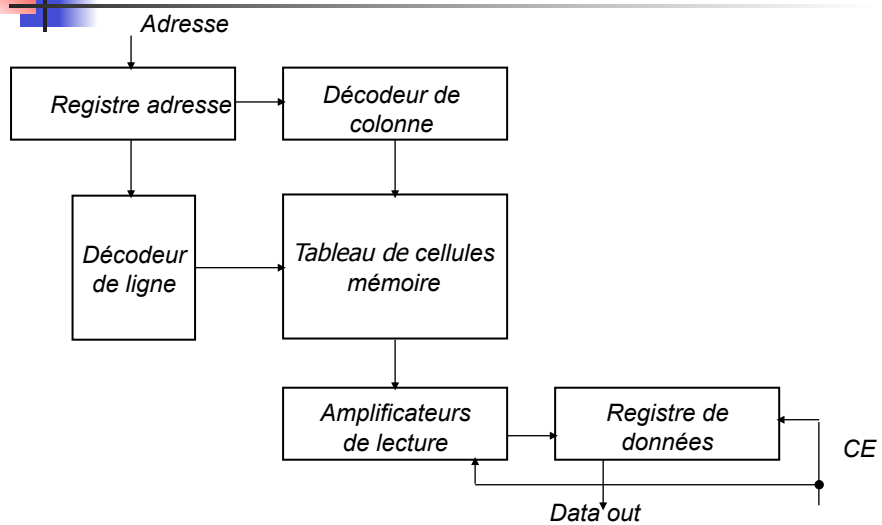




Modèle fonctionnel de ROM



Modèle fonctionnel de ROM





Plan

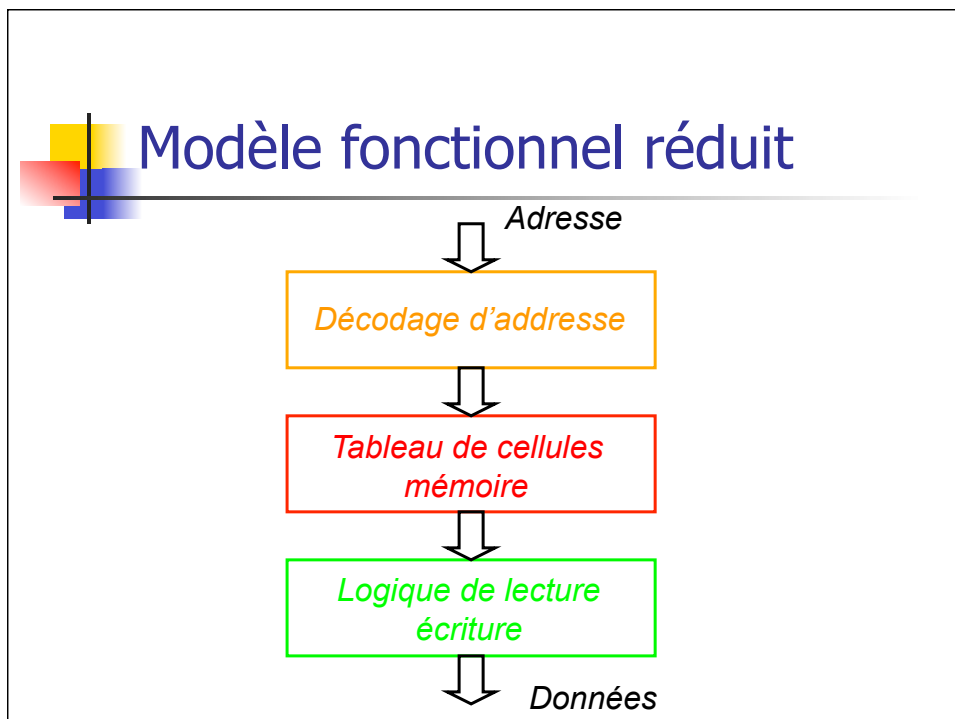
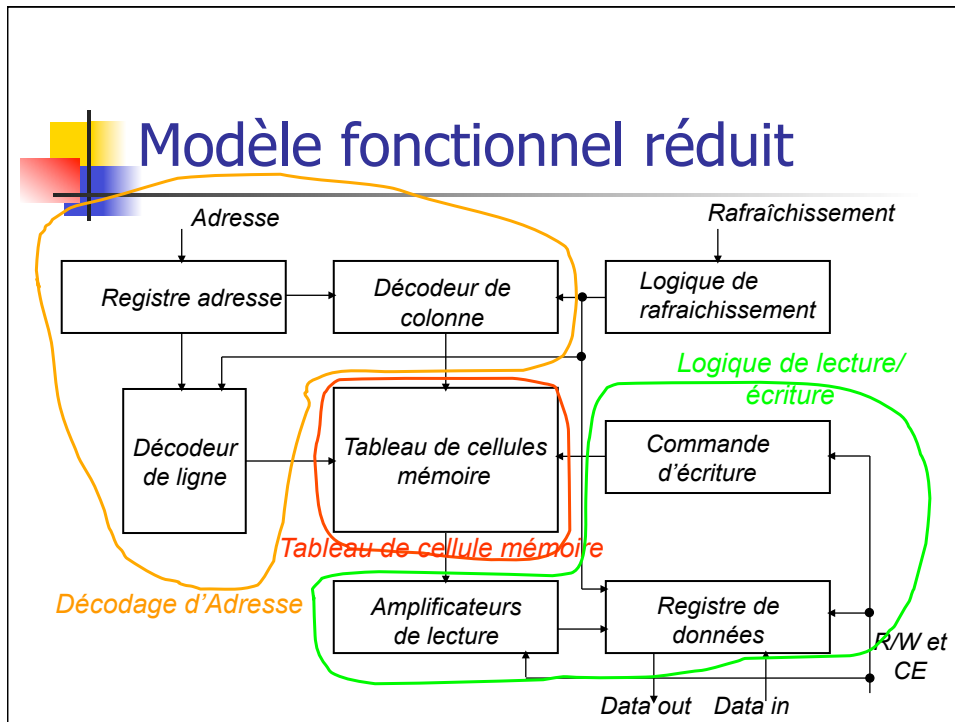
- Introduction
- Modélisation de la mémoire
- **Mécanismes de défaillances et modélisation de fautes**
- Algorithmes de test pour RAM
- Test des ROMs
- BIST mémoire

13



Fautes Fonctionnelles

- | | |
|--|--|
| ■ Collage de cellules | ■ Collage de ligne d'adresse |
| ■ Collage de Driver | ■ Ouverture de ligne d'adresse |
| ■ Collage de ligne | ■ Court-circuit entre lignes d'adresse |
| ■ Collage de ligne de sélection | ■ Circuit ouvert de décodeur |
| ■ Collage de ligne de donnée | ■ Mauvais accès |
| ■ Circuit ouvert de ligne de donnée | ■ Accès Multiple |
| ■ Court-circuit entre lignes de donnée | ■ Une Cellule peut être mise à 0 et pas à 1 (ou vice versa) |
| ■ Crosstalk entre ligne de donnée | ■ Interaction entre cellules dépendant de la configuration de la mémoire |

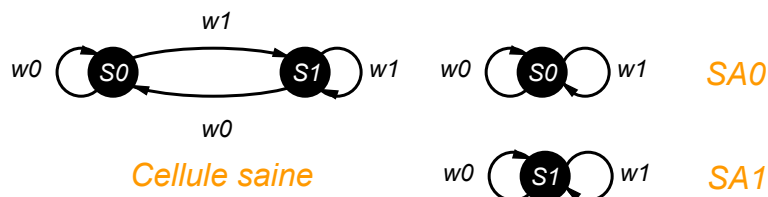


Fautes fonctionnelles réduites

- SAF : fautes de collage
 - TF : fautes de transition
- } Une seule cellule impliquée
-
- CF : fautes de couplages
- } 2 cellules impliquées ou plus
-
- NPSF : fautes de voisinage
- } K cellules impliquées

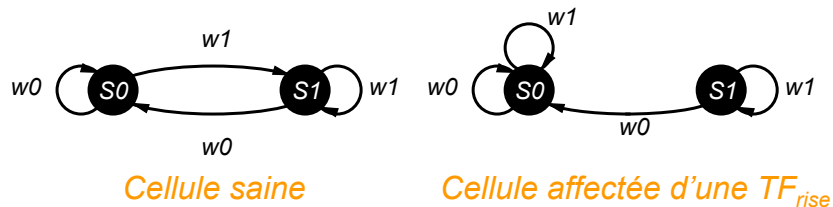
Stuck-at-Fault

- La valeur logique d'une ligne ou d'une cellule est toujours 0 (SA0) ou 1 (SA1)
- Pour détecter les SAF des cellules mémoires
 - SA0 : Write 1 Read 1 (w1 r1)
 - SA1 : Write 0 Read 0 (w0 r0)



Faute de Transition

- Une cellule ne peut pas effectuer une transition $0 \rightarrow 1$ (TFrise) ou une transition $1 \rightarrow 0$ (TFfall)
- Pour détecter une faute de transition :
 - TFrise : w0 w1 r1
 - TFfall : w1 w0 r0



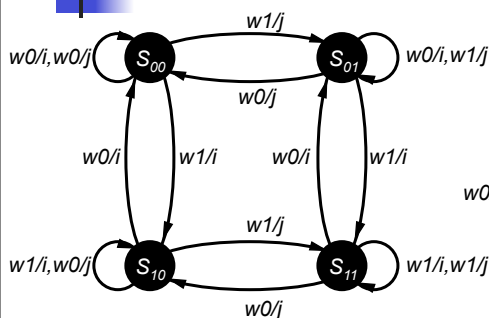
Fautes de couplage (2 cellules)

- Implique deux cellules : une cellule victime et une cellule agresseur
- Différentes sortes de fautes de couplage existent :
 - Faute d'Inversion
 - Faute Idempotente
 - Faute de couplage d'état
 - Fautes de court-circuit
 - Fautes de couplage dynamique
 -

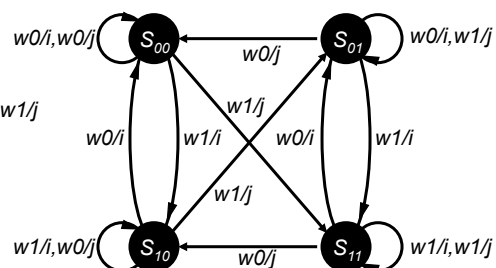
FC d'Inversion

- Faute de couplage d'inversion (CFin): Le contenu de la cellule victime est inversée par une transition de la cellule agresseur
- Suivant le type de transition ($0 \rightarrow 1$ or $1 \rightarrow 0$), il y a deux types de CFin :
 - $< \uparrow ; \Downarrow >$ $< \downarrow ; \Uparrow >$
- Pour détecter une CFin entre une cellule x (victime) et une cellule y (agresseur)
 - CFin (y rise \rightarrow x inversée) : $w0x \ w0y \ w1y \ r0x$.
 - CFin (y fall \rightarrow x inversée) : $w0x \ w1y \ w0y \ r0x$.

FC d'Inversion



Deux cellules saines

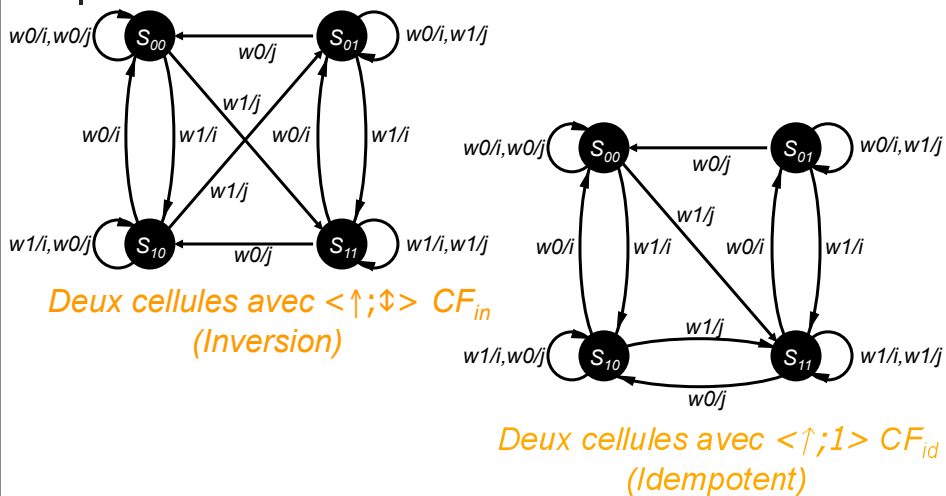


Deux cellules avec $< \uparrow ; \Downarrow >$ CFin

FC idempotente

- Faute de couplage idempotente (CFid) : La cellule victime est forcée à 0 ou 1 si la cellule agresseur a une transition $0 \rightarrow 1$ ou $1 \rightarrow 0$
- Suivant le type de transition ($0 \rightarrow 1$ ou $1 \rightarrow 0$), il y a 4 types possible de CFid :
 - $\langle \uparrow; 0 \rangle \langle \downarrow; 0 \rangle \langle \uparrow; 1 \rangle \langle \downarrow; 1 \rangle$
- Pour détecter une CFid entre une cellule x (victime) et une cellule y (agresseur)
 - CFid (y rise \rightarrow x=0): $w1x \ w0y \ w1y \ r1x$
 - CFid (y fall \rightarrow x=1): $w0x \ w1y \ w0y \ r0x$
 - CFid (y rise \rightarrow x=1): $w0x \ w0y \ w1y \ r0x$
 - CFid (y fall \rightarrow x=0): $w1x \ w1y \ w0y \ r1x$

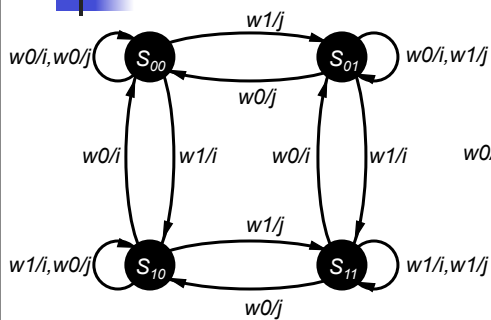
FC idempotente



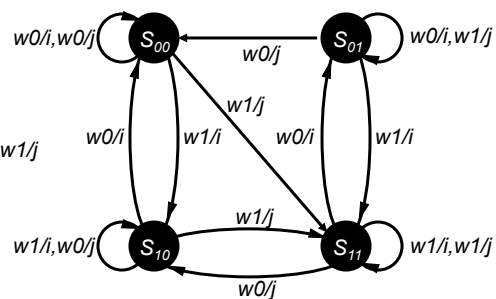
FC d'état

- Faute de couplage d'état (CFst): La cellule victime est forcée à 0 ou 1 si la cellule agresseur est dans certain état
- Il y a 4 types possible de CFst :
 $\langle 0; 0 \rangle$ $\langle 0; 1 \rangle$ $\langle 1; 0 \rangle$ $\langle 1; 1 \rangle$
- Pour détecter une CFst entre une cellule x (victime) et une cellule y (agresseur)
 - CFst ($y=0 \rightarrow x=0$): $w_{1x} w_{0y} r_{1x}$
 - CFst ($y=0 \rightarrow x=1$): $w_{0x} w_{0y} r_{0x}$
 - CFst ($y=1 \rightarrow x=0$): $w_{1x} w_{1y} r_{1x}$
 - CFst ($y=1 \rightarrow x=1$): $w_{0x} w_{1y} r_{0x}$

FC d'état



2 cellules saines



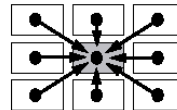
Deux cellules avec $\langle 1; 1 \rangle$ SCF

Fautes de court-circuit (BF)

- Un court-circuit entre deux lignes ou deux cellules
- C'est une faute bidirectionnelle due à un niveau logique et non pas une transition
- Deux sortes de fautes de court-circuit (BFs) :
 - AND-type (ABF): les lignes/cellules court-circuitées prennent la valeur du AND de leur valeur saine
 - OR-type (OBF): les lignes/cellules court-circuitées prennent la valeur du OR de leur valeur saine
- Pour détecter une faute de court-circuit, il est nécessaire d'écrire des valeurs différentes dans des cellules adjacentes (voir algorithme de l'échiquier).

Pattern Sensitive (PSF)

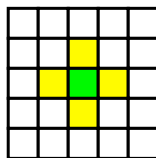
- La cellule victime est forcée à 0 ou à 1 si un certain nombre de cellules voisines présente une configuration particulière.
- Les fautes PSF sont le cas le plus général des fautes de couplage
- Les fautes de couplage NPSF (Neighborhood Pattern Sensitive Fault), réduisent le voisinage aux cellules immédiatement voisine de la cellule victime (cellule de base)



- Equivalente à une faute de couplage avec N agresseurs (jusqu'à 8 cellules adjacentes)
- Difficile à détecter
 - Pour chaque cellule, l'effet de toutes les combinaisons possibles (28) des cellules adjacentes doit être testé.

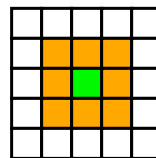
Neighborhood Pattern Sensitive (NPSF)

- En pratique deux types de fautes sont utilisés
- Le type-1 NPSF avec 4 cellules voisines
- Le type-2 NPSF avec 8 cellules voisines



cellule de base
Cellules pour
Type-1

Type-1



cellule de base
Cellules pour
Type-2

Type-2

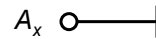
Neighborhood Pattern Sensitive (NPSF)

- **NPSF Active (ANPSF) :**
 - Changement de la cellule de base du a une transition dans les cellules voisines
 - Chaque cellule de base doit être lue dans l'état 0 et dans l'état 1 pour toutes les changements possibles dans la configuration des cellules voisines
- **NPSF Passive (PNPSF) :**
 - Le changement de la cellule de base est impossible du à une certaine configuration des cellules voisines
 - Chaque cellule de base doit être écrite et lue dans l'état 0 et dans l'état 1 pour toutes les permutations des configurations des cellules voisines
- **NPSF Statique (SNPSF) :**
 - Le contenu de la cellule de base est forcé à une certaine valeur du à une certaine configuration des cellules voisines
 - Chaque cellule de base doit être lue dans l'état 0 et dans l'état 1 pour toutes les permutations des configurations des cellules voisines

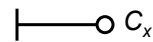
Fautes décodeur d'adresse AF

■ 4 types de fautes du décodeur d'adresse

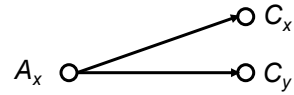
1. Une certaine adresse pointe vers aucune cellule



2. Une certaine cellule n'est jamais accédée



3. Une certaine adresse pointe vers plusieurs cellules



4. Une certaine cellule est accédée par plusieurs adresses



Fautes décodeur d'adresse AF

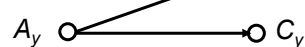
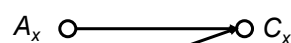
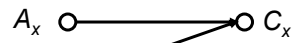
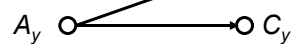
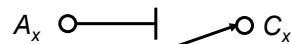
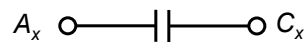
■ 4 combinaisons des fautes du décodeur d'adresse

■ Fault A: 1+2

■ Fault B: 1+3

■ Fault C: 2+4

■ Fault D: 3+4





Faute de Stuck-Open

- Une cellule n'est plus accessible due à un circuit ouvert dans une wordline (WL)
- Identique à une faute de décodeur d'adresse (2) – une cellule n'est jamais accédée



Rétention de données (DRF)

- Une cellule est incapable de retenir sa valeur après un certain temps
- Faute due par exemple à un élément de pull-up défectueux dans une cellule de SRAM
- La cellule perd sa valeur du fait des courants de fuite
- Deux différentes DRFs existent (perte du 1 et perte du 0) et peuvent être simultanément présente
- Pour détecter une DRF => on introduit un délai avant de lire le contenu de la cellule (usuellement ~ 10-100 ms)
- Peut être très aisément rajouté à tout algorithme
- Le temps de test augmente de manière dramatique !



Et bien d'autres ...

- Les fautes liées
 - sont deux ou plusieurs fautes (fautes de couplage) qui affectent la même cellule
- Les fautes dynamiques
 - Nécessitent une séquence d'opérations (c.f. paire de vecteurs)



Plan

- Introduction
- Modélisation de la mémoire
- Mécanismes de défaillances et modélisation de fautes
- **Algorithmes de test pour RAM**
- Test des ROMs
- BIST mémoire



Algorithmes de Test : notations

- \uparrow : indique un parcours des adresses en ordre croissant
- \downarrow : indique un parcours des adresses en ordre décroissant
- w0 : écrire un 0 à l'adresse courante
- w1 : écrire un 1 à l'adresse courante
- r0 : lire l'adresse courante en attendant un 0
- r1 : lire l'adresse courante en attendant un 1
- (...) : élément de l'algorithme
- $\{(...),(...),...,(...)\}$: algorithme complet



Algorithmes de Test

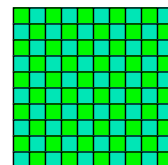
- Un test comportemental complet est définitivement trop long
- Les méthodes de test classiques et les plus anciennes avaient des temps d'exécution proportionnels à :
 - n (zero-one, checkerboard, ...)
 - n^2 et $n \cdot \log_2(n)$ (walking1/0, ping-pong, Galpat, Galcol, ...)

Zero-One

- Ce test minimal consiste à écrire des 0 et des 1 dans la mémoire
 - Etape1: écrire 0 dans toutes les cellules
 - Etape2: lire toutes les cellules (0 attendu)
 - Etape3: écrire 1 dans toutes les cellules
 - Etape4: lire toutes les cellules (1 attendu)
- Test en $O(n)$
- Taux de couverture :
 - Ne détecte pas toutes les AFs
 - SAFs détectées si le décodeur d'adresse est sain
 - Ne détecte pas toutes les TFs et CFs

Checkerboard

- Les cellules sont divisées en 2 groupes
 - Etape1: écrire 1 dans les cellules vertes et 0 dans les bleus
 - Etape2: lire (et vérifier) toutes les cellules
 - Etape3: écrire 0 dans les cellules vertes et 1 dans les bleus
 - Etape4: lire (et vérifier) toutes les cellules
- Test en $O(n)$
- Taux de couverture :
 - Ne détecte pas toutes les AFs, TFs et CFs
 - SAFs détectées si le décodeur d'adresse est sain
 - Ce test est capable de détecter les fautes de court-circuit





Temps de test

	Nombre d'opérations		
taille	n	$n \cdot \log_2 n$	n^2
1Mb	0.063	1.26	18.33
16Mb	1.01	24.16	4691.3
256Mb	16.11	451	1200959.9
2Gb	128.9	3994.4	76861433.7



Les tests March

- Le test parcourt ("march") la mémoire
- Le test est composé d'éléments de march qui sont représentés entre ()
- Les tests march sont les test les plus simples pour détecter les SAFs, TFs et les CFs



MATs++

$\{\uparrow(w0); \uparrow(r0,w1); \downarrow(r1,w0,r0)\}$

- 6n operations
- Toutes les SAFs détectées
- Toutes les AFs détectées
- Toutes les TFs détectées



Tests March : Résumé

Name	Ref.	algorithm
MATS	[NAI79]	$\{ \downarrow(w0); \downarrow(r0,w1); \downarrow(r1) \}$
MATS+	[ABA83]	$\{ \downarrow(w0); \uparrow(r0,w1); \downarrow(r1,w0) \}$
MATS++	[VAN91]	$\{ \downarrow(w0); \uparrow(r0,w1); \downarrow(r1,w0,r0) \}$
March X	[VAN91]	$\{ \downarrow(w0); \uparrow(r0,w1); \downarrow(r1,w0); \downarrow(r0) \}$
March C-	[MAR82]	$\{ \downarrow(w0); \uparrow(r0,w1); \uparrow(r1,w0); \downarrow(r0,w1); \downarrow(r1,w0); \downarrow(r0) \}$
March A	[SUK81]	$\{ \downarrow(w0); \uparrow(r0,w1,w0,w1); \uparrow(r1,w0,w1); \downarrow(r1,w0,w1,w0); \downarrow(r0,w1,w0) \}$
March Y	[VAN91]	$\{ \downarrow(w0); \uparrow(r0,w1,r1); \downarrow(r1,w0,r0); \downarrow(r0) \}$
March B	[SUK81]	$\{ \downarrow(w0); \uparrow(r0,w1,r1,w0,r0,w1); \uparrow(r1,w0,w1); \downarrow(r1,w0,w1,w0); \downarrow(r0,w1,w0) \}$
March GS	[VAN93]	$\{ \downarrow(w0); \uparrow(r0,w1,r1,w0,w1); \uparrow(r1,w0,r0,w1); \downarrow(r1,w0,w1,w0); \downarrow(r0,w1,r1,w0); \text{Del}; \downarrow(r0,w1,r1); \text{Del}; \downarrow(r1,w0,r0) \}$
March M	[MIK96]	$\{ \downarrow(w0); \uparrow(r0,w1,r1,w0); \downarrow(r0); \uparrow(r0,w1); \downarrow(r1); \uparrow(r1,w0,r0,w1); \downarrow(r1); \downarrow(r1,w0) \}$
March LR	[VAN96]	$\{ \downarrow(w0); \downarrow(r0,w1); \uparrow(r1,w0,r0,w1); \uparrow(r1,w0); \uparrow(r0,w1,r1,w0); \uparrow(r0) \}$
March U	[VAN97]	$\{ \downarrow(w0); \uparrow(r0,w1,w0,w1,r1); \uparrow(r1,w0,w1,w0,r0); \downarrow(r0,w1,w0,w1,r1); \downarrow(r1,w0,w1,w0,r0); \downarrow(r0) \}$
March LA	[VAN99]	$\{ \downarrow(w0); \uparrow(r0,w1,r1,w0); \uparrow(r0,w1); \downarrow(r1,w0,r0,w1); \downarrow(r1,w0) \}$
March SR	[VAN00]	$\{ \downarrow(w0); \uparrow(r0,w1,r1,w0); \uparrow(r0,r0); \downarrow(r1,w1); \downarrow(r1,w0,r0,w1); \downarrow(r1,r1) \}$
March SS	[HAM02]	$\{ \downarrow(w0); \uparrow(r0,r0,w0,r0,w1); \uparrow(r1,r1,w1,r1,w0); \downarrow(r0,r0,w0,r0,w1); \downarrow(r1,r1,w1,r1,w0); \downarrow(r0) \}$

Taux de couverture

Algorithme	Taux de couverture								Nbre d'op.
	S AF	AF	TF	CF in	CF d	CF dyn	SC F	Fautes liées	
MATS	All	Some							4.n
MATS+	All	All							5.n
MATS++	All	All	All						6.n
March X	All	All	All	All					6.n
March C-	All	All	All	All	All	All	All		10.n
March A	All	All	All	All				Toutes CFids liées, certaines CFins liées avec les CFids	15.n
March Y	All	All	All	All				toutes TFs liées avec les CFins	8.n
March B	All	All	All	All				Toutes les CFids liées, toutes TFs liées avec CFids ou CFins, certaines CFins liées avec CFids	17.n

Plan

- Introduction
- Modélisation de la mémoire
- Mécanismes de défaillances et modélisation de fautes
- Algorithmes de test pour RAM
- **Test des ROMs**
- BIST mémoire



Test de ROM

- Les données correctes que doit contenir la mémoire sont déjà connues
- Le modèle fonctionnel réduit est le même que pour une RAM
- Les SAF couvrent toutes les fautes du modèle réduit
 - TFs, CFs et NPSFs ne peuvent pas apparaître dans les cellules mémoires
 - TFs, CFs et NPSFs sont très peu probables dans le logique de lecture
 - Toutes les AFs sont couvertes par les SAF dans le tableau mémoire car il n'y a pas d'écriture en fonctionnement normal



Méthode classique

- Lire tous les mots de la ROM
- Compresser la sortie avec un LFSR (Linear Feedback Shift Register) de n bits : i.e. faire une division polynomiale permettant de calculer le code cyclique redondant (CRC)
- Comparer le CRC calculé avec le CRC correct stocké dans le dernier mot de la ROM
- Le parcours des adresses peut se faire deux fois en ordre croissant et décroissant pour vérifier la présence possible de lecture destructive



Plan

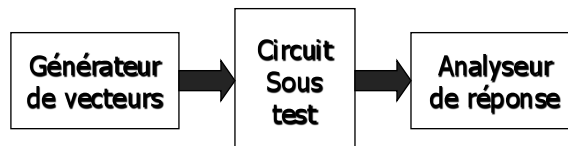
- Introduction
- Modélisation de la mémoire
- Mécanismes de défaillances et modélisation de fautes
- Algorithmes de test pour RAM
- Test des ROMs
- **BIST mémoire**

49



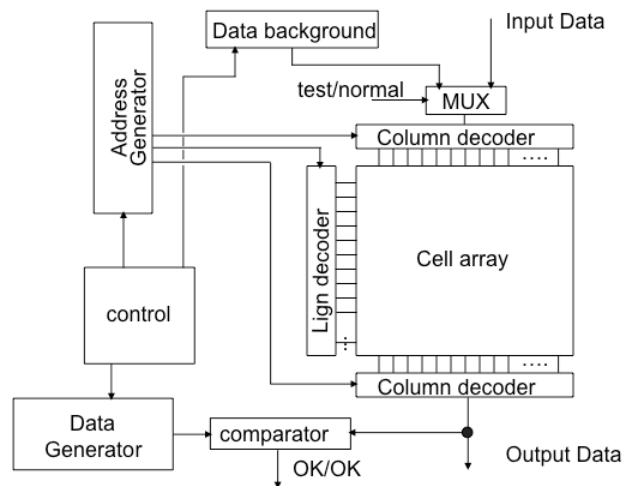
Rappels sur le BIST

- Dans les techniques de BIST (Built-In Self-Test), les vecteurs de test sont produits et les résultats analysés sur le circuit



- L'objectif étant de résoudre tout ou partie des problèmes suivants : mauvaise contrôlabilité et mauvaise observabilité, test à la vitesse nominale de fonctionnement, coût de l'ATE, réduction du temps de test, test "transparent" pour les concepteurs,

BIST de RAM basé sur des tests March



BIST de ROM

