ATPG
Principe et généralités

Mounir BENABDENBI
Mounir.Benabdenbi@lip6.fr

Laboratoire d'Informatique de Paris 6 (LIP6)

Principe (circuit combinatoire)

- Pour mettre en évidence une panne de collage à 0 (resp. 1) sur une ligne L, il est nécessaire que le vecteur de test soit tel que :

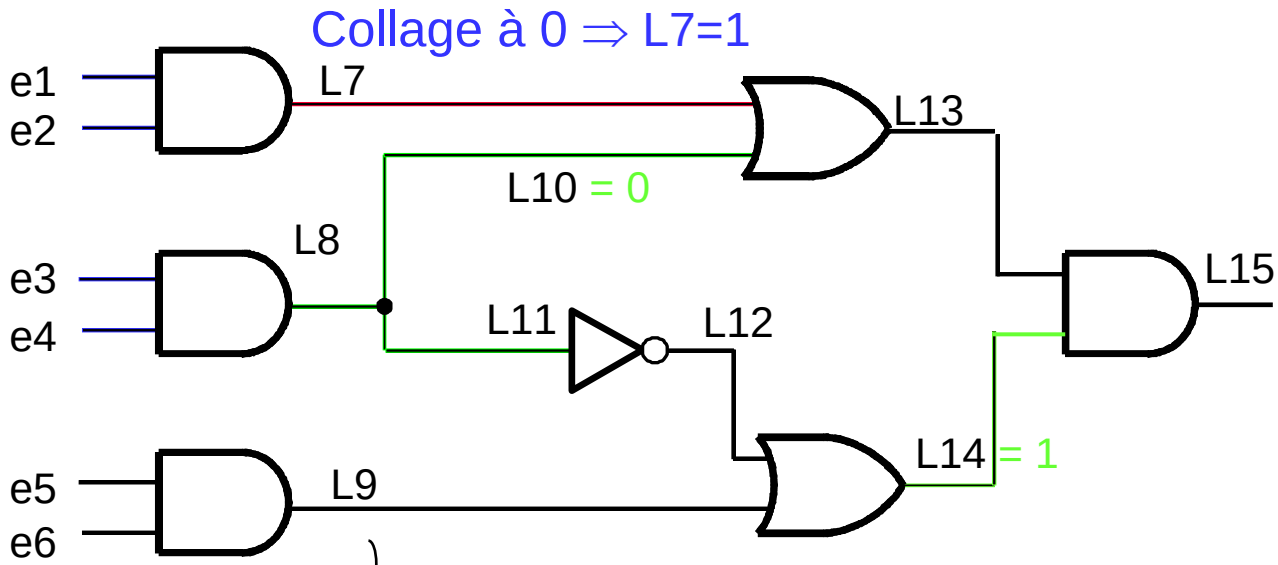
  1/ il produise la valeur 1 (resp. 0) sur la ligne L

  2/ qu'une sortie du circuit indique sans ambiguïté que la ligne L est à la valeur 0 ou à la valeur 1

- Les trois actions nécessaires sont donc :
  - **la sensibilisation** de la faute
    - positionnement à l'inverse du collage
  - **la propagation** de l'erreur vers une SP
    - par chemin simple
    - par chemins multiples
  - **la justification** des valeurs de sensibilisation et de propagation par assignation des EPs

Exemple

(Sensibilisation simple)



Sensibilisation

Propagation

Justification

Tout vecteur de test / L7 = 1, L10 = 0, L14 = 1

⇒ L15 = 1 dans le circuit sain

L15 = 0 dans le circuit fautif

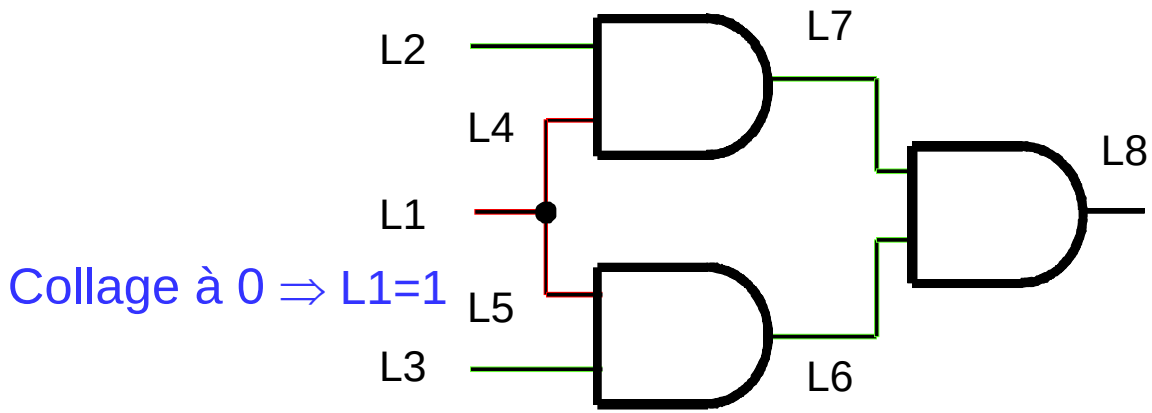
⇒ c'est un vecteur de test

⇒ (e3 = 0 ou e4 = 0) et e1 = e2 = 1

⇒ ex : vecteur de test CA0/L7 (1,1,0,X,X,X)

1 chemin de propagation unique : (L7,L13,L15)

Sensibilisation multiple nécessaire



Collage à 0 ⇒ L1=1

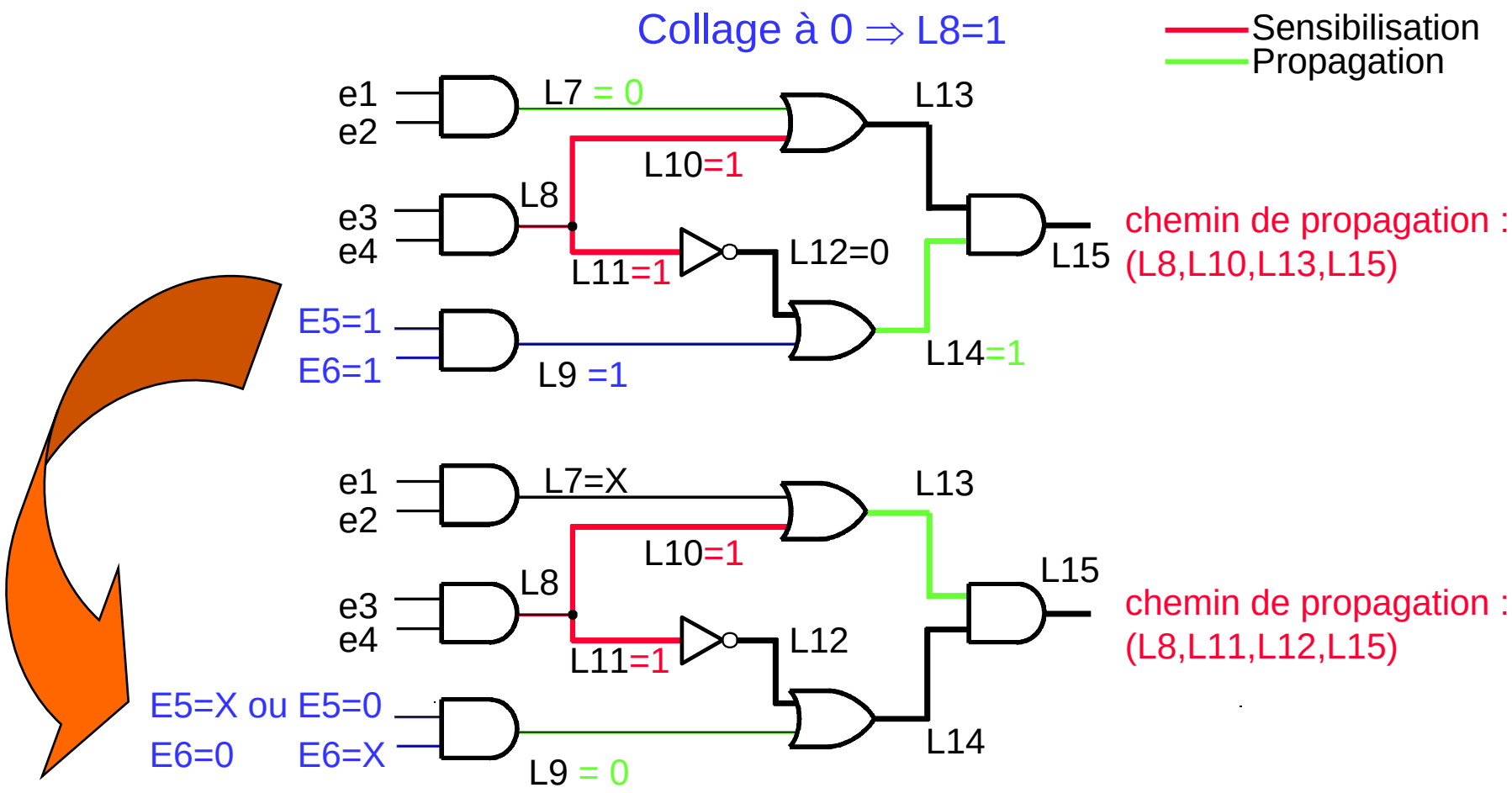
2 chemins de propagation : (L1,L4,L7,L8) et (L1,L5,L6,L8)
Il doivent être sensibilisés simultanément

Sensibilisation

Propagation

Tout vecteur de test / L1 = 1, L2 = 1, L3 = 1, L7 = 1, L6 = 1
⇒ L8 = 1 dans le circuit sain
L8 = 0 dans le circuit fautif
⇒ c'est un vecteur de test

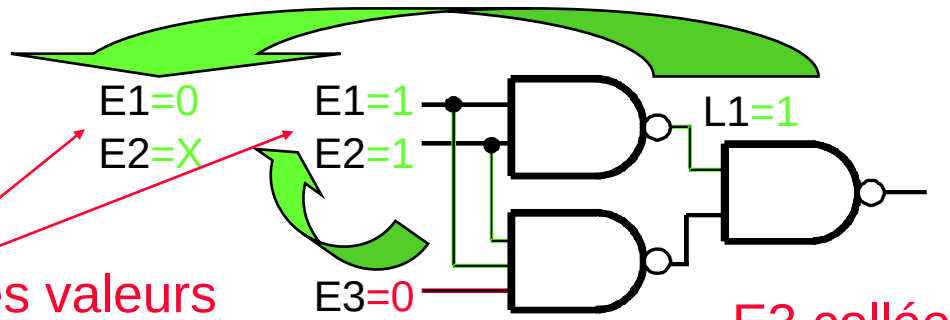
Sensibilisation multiple impossible



Les chemins de propagation ne peuvent être sensibilisés simultanément

Génération de vecteurs de test : les difficultés

- divergence et reconvergence de lignes
- nombre de chemins
- redondance et indétectabilité
  - Def1 : S'il n'existe pas de vecteur de test pour mettre en évidence une faute, la faute est indétectable
  - Def2 : Si Contexte = collages sur les interconnexions d'un circuit, alors si la faute est indétectable on dit que la connexion est redondante (elle peut être supprimée)



Incompatibilité des valeurs de propagation

E3 collée à 1 est indétectable

Génération Automatique de Séquences de Test

Mounir BENABDENBI
Mounir.Benabdenbi@lip6.fr

Laboratoire d'Informatique de Paris 6 (LIP6)

Génération de vecteurs de test

- Généralités
- Test Déterministe
  - Gestion des fautes
  - Principes et Concepts élémentaires
  - Génération au niveau structurel
    - circuit combinatoire (D-algorithme, 9V-algorithme, PODEM, FAN, ...)
    - circuit séquentiel
  - Génération au niveau fonctionnel
    - Mémoires
    - PLAs
- Test Exhaustif
- Test Aléatoire

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
- 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 Tc, efficacité du test (test déterministe), qualité de test

Taux de couverture : Tc

- Taux de couverture de fautes
  - Tc = (# de fautes détectées par la séquence de test) / (# de fautes total (dans le modèle))
  - connu par simulation de faute
  - borne inférieure du Taux de couverture des fautes physiques (TcΦ)
  - utilisé dans l ’industrie
  - l’expérience montre que Tc élevé $\Rightarrow$ TcΦ élevé
  - Tc et niveau de défaut DL :
    - Y = rendement de fabrication (proba. qu’un circuit fabriqué soit sain)
    - Hypothèse : Y=0,5 Si Tc = 99%, DL = 100ppm
      Si Tc = 95%, DL = 350ppm

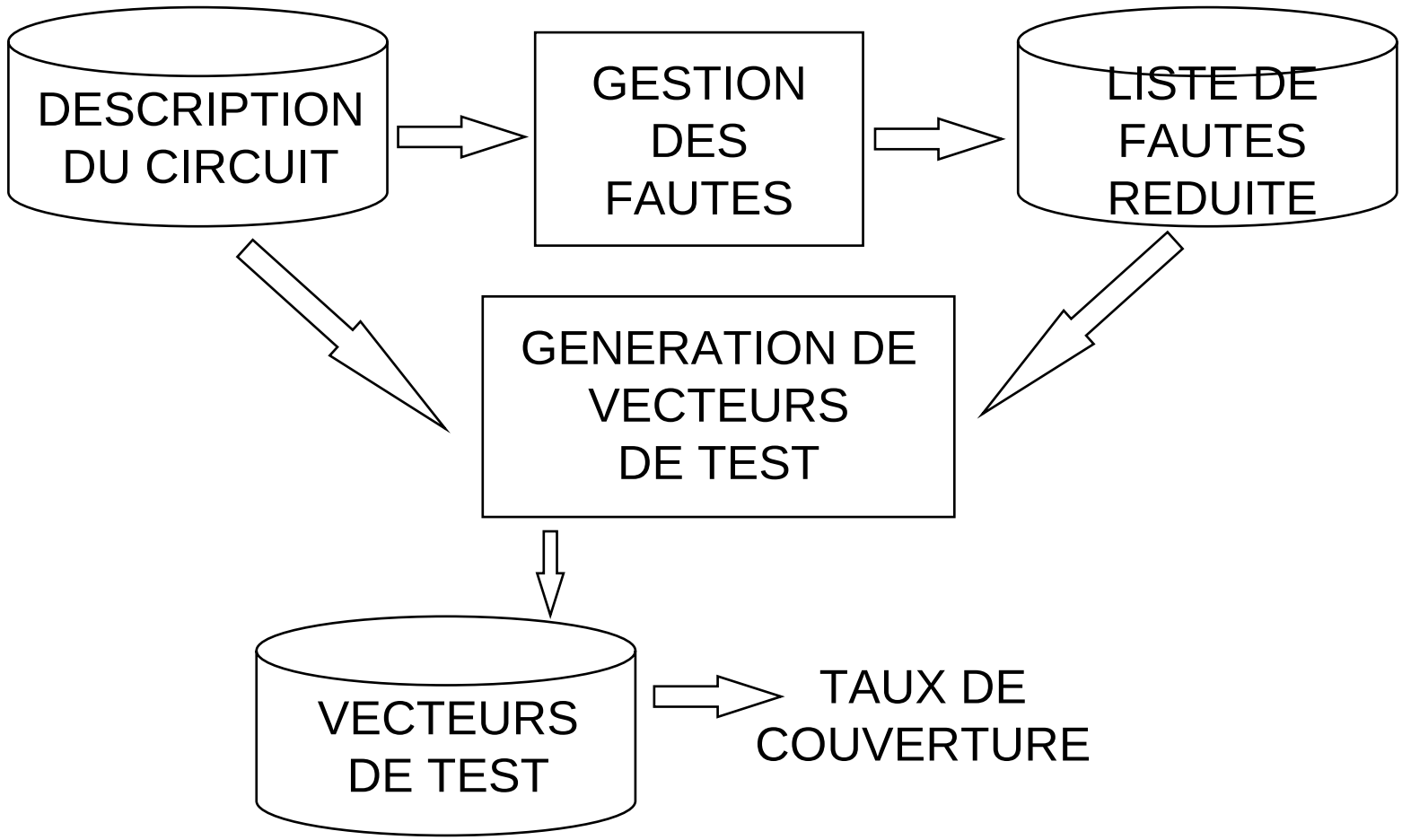
Test Déterministe

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

Génération Automatique de vecteurs de test

(**ATPG** : **A**utomatic **T**est **P**attern **G**eneration)

☑ Architecture d'un système d'ATPG



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")
  - Inconvénient potentiel : ne fournit pas toujours de vecteur de test alors que celui-ci existe (cas des chemins de sensibilisation multiples) => D-Algorithme

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
  - 3/ construire le vecteur de test sur les EPs qui réalise toutes les assignations effectuées en 1/ et 2/
- Algèbre à 5 valeurs
  - 0, 1, X, D (1/0), $\overline{D}$ (0/1)
  - D/ligne $\Rightarrow$ ligne=1 dans le circuit sain, 0 dans le circuit fautif
  - $\overline{D}$/ligne $\Rightarrow$ ligne=0 dans le circuit sain, 1 dans le circuit fautif
  - *Dans ce cours $\overline{D}$ est parfois noté D**
- 3 notions
  - D-cube primitif, couverture singulière, D-cube de propagation

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 ($\overline{D}$) en appliquant certaines valeurs sur les entrées
- Couverture singulière d'une porte
  - Table de vérité réduite utilisant les valeurs (0, 1, X)
- D-cube de propagation d'une porte
  - Spécifie les valeurs à appliquer sur les entrées exceptée une (ou plus) tel qu'un changement sur cette entrée (ou ces entrées) induit un changement en sortie de la porte.

e1	e2	S
1	1	D

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

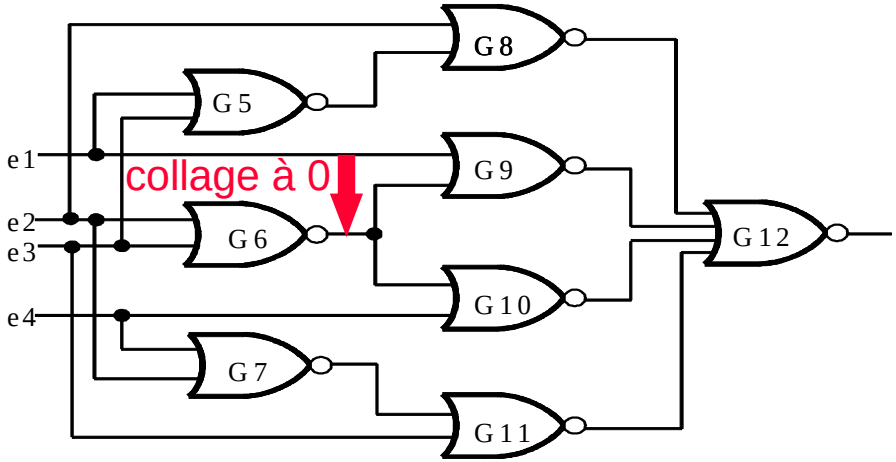
e1	e2	S
0	x	0
x	0	0
1	1	1

Couverture singulière
AND à 2 entrées

e1	e2	S	
D	1	D	cube simple
1	D	D	’’
D	D	D	cube multiple
$\overline{D}$	1	$\overline{D}$	cube simple
1	$\overline{D}$	$\overline{D}$	’’
$\overline{D}$	$\overline{D}$	$\overline{D}$	cube multiple

D-cube de
propagation
AND à 2 entrées

D-algorithme : exemple



Lignes

Cubes

	e1	e2	e3	e4	l5	l6	l7	l8	l9	l10	l11	l12
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**

…..(il y en a d'autres)

Tableau des D-cubes de propagation du circuit

$\overline{D} = D^*$

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

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

e1 e2 e3 e4 l5 l6 l7 l8 l9 l10 l11 l12

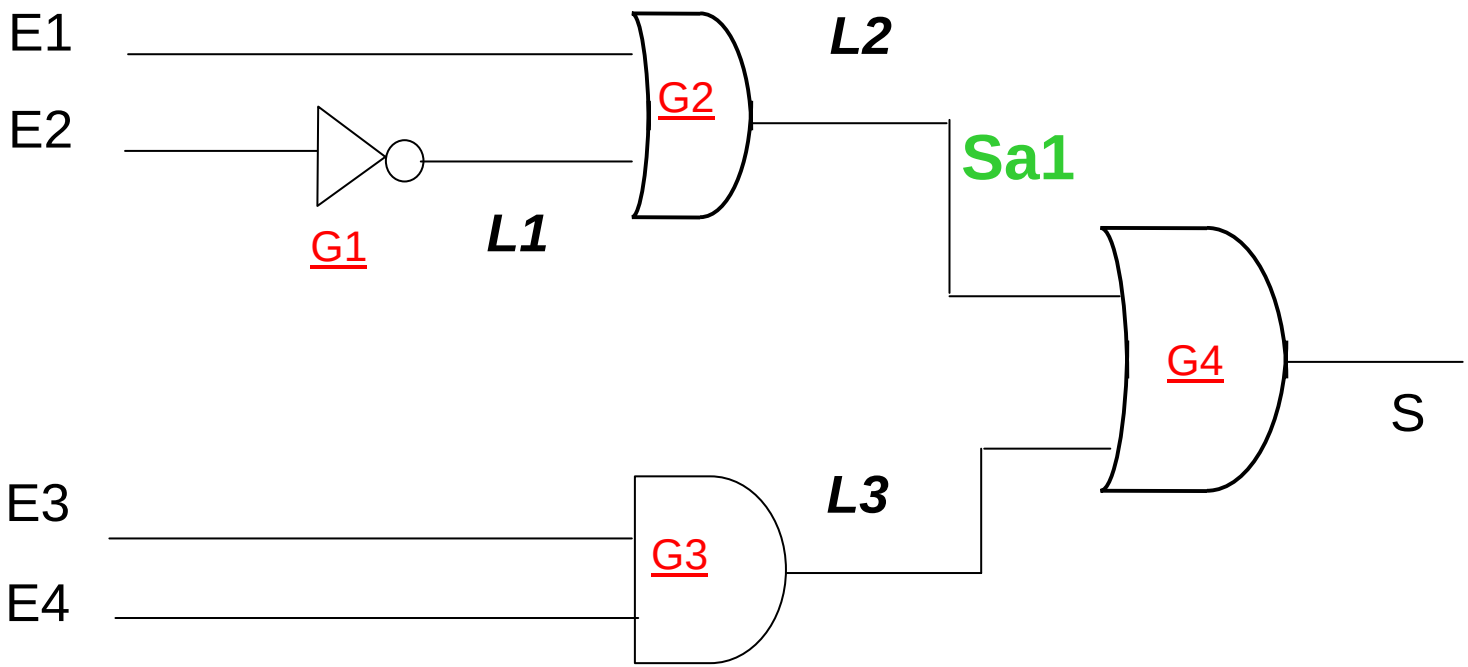
x 0 0 x x D x x x x x x

2/ Propager le D jusqu'à une sortie : règles d 'intersection des D-cubes de propagation

∩	0	1	x	D	D*
0	0	∅	0	Ψ	Ψ
1	∅	1	1	Ψ	Ψ
x	0	1	x	D	D*
D	Ψ	Ψ	D	μ	λ
D*	Ψ	Ψ	D*	λ	μ

Opération de D-intersection (∩)

D-algorithme : exemple



On veut trouver le vecteur permettant de détecter le collage a 1 de l'equipotentielle L2 (Sa1 de L2)

Collage a 1 de L2 donc il faut mettre un 0 sur L2 : donc 0 sur circuit sain et 1 sur circuit fautif

=> D* à propager jusqu'à la sortie (je choisis de propager D* dans les D-cubes qui suivent)

D-algorithme : exemple
Tableau de propagation des D-cubes

Cube	**E1**	**E2**	**E3**	**E4**	**L1**	**L2**	**L3**	**S**	**Porte**
A		D			D*				G1 NOT
B	0				D*	D*			G2 OR
C	D*				0	D*			G2 OR
D			**1 car porte and**	D			D		G3 AND
E			D	1			D		G3 AND
F						**0 car porte or**	D*	D*	G4 choix du D* car Sa1 a tester, on aurait pu remplacer par D pour un autre collage
G						D*	0	D*	G4 OR
H= polynome primitif Sa1 de L2	0				0	D*			
I= H inter F (propagation vers S)	0				0	D*	0	D*	
JUSTIFICATION pour verifier que le vecteur est valable (pas de conflit)	0	**1**	**Xou0**	**0ouX**	**0**	D*	**0**	D*	

D-algorithme : exemple
Tableau des couvertures singulières

	E1	**E2**	**E3**	**E4**	**L1**	**L2**	**L3**	**S**	**Porte**
		0			1				G1 NOT
		1			0				G1 NOT
	1				X	1			G2 OR
	X				1	1			G2 OR
	0				0	0			G2 OR
			0	x			0		G3 AND
			x	0			0		G3 AND
			1	1			1		G3 AND
						1	x	1	G4 OR
						X	1	1	G4 OR
	0				0	0	0	0	G4 OR

D-algorithme : exemple

Marche a suivre pour trouver le(s) vecteur(s) en utilisant l'algorithme D:

1. Ecrire le tableau de propagation des D-cubes pour toutes les portes jusqu'au cube G.
   Dans ce cas j'ai rempli le tableau avec D* pour me simplifier la propagation vers S car vu que c'est Sa1 de L2 qu'on veut tester c'est un D* a propager.

2. Ecrire le tableau des couvertures singulieres

3. Ecrire le D-cube primitif correspondant au collage **(phase de sensibilisation),** ici c'est le cube H

D-algorithme : exemple

Marche a suivre pour trouver le(s) vecteur(s) en utilisant l'algorithme D:

4. En se servant des D-cubes de propagation et du tableau des couvertures singulieres, propager le D* jusqu'à la sortie.

   On doit avoir sur la sortie un D ou un D*, ni 1 ni 0 sur S. On traverse une porte en faisant l'intersection des D-cubes qui la concerne. On utilise pour cela le tableau des intersections fourni en cours : D* inter D*= D* pour obtenir le cube I qui nous donne D* sur S **(phase de propagation)**

5. On cherche a remonter vers les entrées pour leur affecter les valeurs correspondantes aux nœuds du circuit.
   Si un conflit de valeurs apparaît sur l'entrée alors le vecteur n'est pas bon, il faut trouver un autre chemin de sensiblisation ou sensibliser plusieurs chemins simultanément. **(phase de justification)**

Le vecteur 0 0 0 0 permet donc de détecter un collage a 1 de L2.

Problèmes avec le D-algorithme

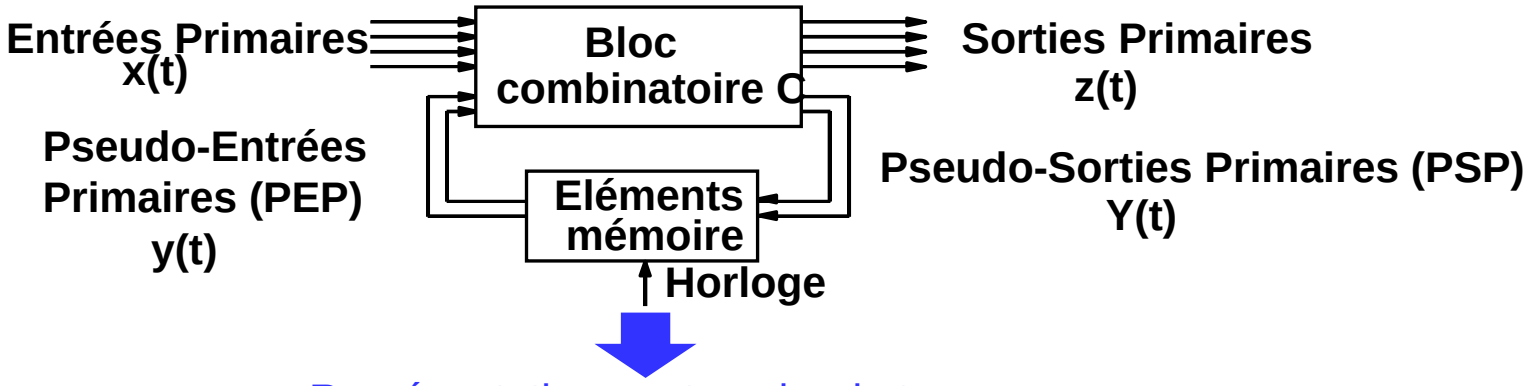
- temps d'exécution trop important (beaucoup de chemins + fautes redondantes)
- amélioration des temps de calcul
  - 9V- algorithme (même stratégie mais sensibilisation simultanée de tous les chemins de propagation)
  - changement de stratégie avec PODEM (parcours dans un arbre)

Test de circuits séquentiels : les difficultés

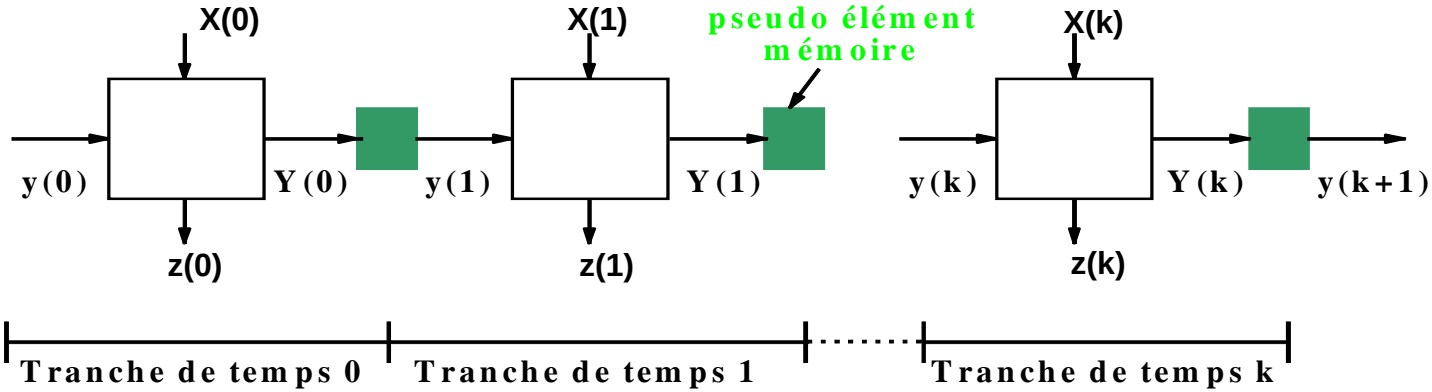
- Le test d'une faute nécessite en général plusieurs cycles
- problèmes structurels
  - les cycles
  - la profondeur séquentielle
  - les chemins reconvergents
  - la sortance des bascules

  } Idem combinatoire
- problèmes comportementaux
  - la densité d'états valides
  - la fonctionnalité

Test de circuits séquentiels
Modélisation du circuit

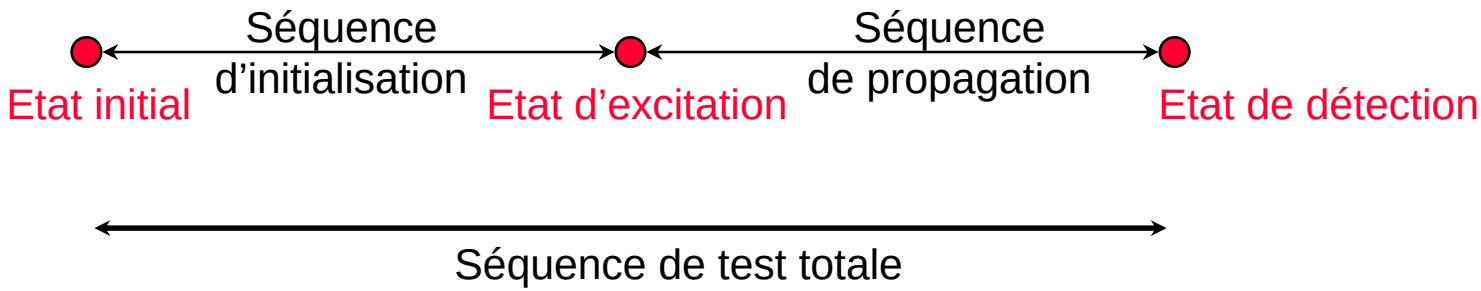


Représentation par tranche de temps :
le circuit est vu comme une succession de circuits (combinatoires) dans le temps
Les éléments de mémorisation du circuit original sont modélisés
comme des **pseudo-éléments combinatoires** de mémorisation
qui font le lien entre les différentes copies temporelles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



Séquence de test

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



Les différentes approches pour la génération automatique

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

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
  - 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, PLA)

Annexe:
Structure d'un fichier STIL (Tetramax)

```
STIL 1.0 {
   Design P2000.9;
}
Header {
}
Signals {
   "a" In;"b" In;"com" In;"s" Out;"vdd" In;"vss" In;
}
SignalGroups {
   "_pi" = '"a" + "b" + "com" + "vdd" + "vss"';
   "_po" = '"s"';
   "_default_In_Timing_" = '"a" + "b" + "com" +
"vdd" + "vss"';
   "_default_Out_Timing_" = '"s"';
}
ScanStructures {
}
Timing {
  WaveformTable "_default_WFT_" {
     Period '100ns';
     Waveforms {
       "_default_In_Timing_" { 0 { '0ns' D; } }
                 …….
  "_default_Out_Timing_" { H { '0ns' X; '40ns' H; } }
PatternBurst "_burst_" { PatList {
   "_pattern_" {
   }
}}
PatternExec {
   PatternBurst "_burst_";
}
Procedures {
   "capture" {
      W "_default_WFT_";
      F { "vss"=0; "vdd"=1; }
      "forcePI": V { "_pi"=#####; "_po"=\j X; }
      "measurePO": V { "_po"=#; }
   }
}
MacroDefs {
   "test_setup" {
      W "_default_WFT_";
      V { "vss"=0; "vdd"=1; }
   }
}
```

Annexe:
Structure d’un fichier STIL (Tetramax)

```
Pattern "_pattern_" {
   W "_default_WFT_";
   "precondition all Signals": C { "_pi"=00000; "_po"=\j X; }
   Macro "test_setup";
  "vecteur0":
   Call "capture" {
      "_pi"=01010; "_po"=H; }
  "vecteur1":
   Call "capture" {
      "_pi"=00010; "_po"=H; }
  "vecteur2":
   Call "capture" {
      "_pi"=10110; "_po"=H; }
  "vecteur3":
   Call "capture" {
      "_pi"=01110; "_po"=H; }
//  "vecteur4":
//   Call "capture" {
  //    "_pi"=10010; "_po"=H; }
}
```