Algorithme D

- Premier algorithme complet (test tous les cas possibles), développé par IBM en 1966
  - Il a été prouvé que l'algorithme D génère toujours un test qui détecte la faute s'il en existe un
- Utilisation de la notation D
  - Pour un signal S, on a :
    - $S = D \Leftrightarrow S = 1$ dans le bon circuit et $S = 0$ dans le circuit fautif
    - $S = \overline{D} \Leftrightarrow S = 0$ dans le bon circuit et $S = 1$ dans le circuit fautif

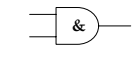
Algorithme D (suite)

- Notation D

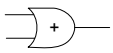
Valeur du signal pour circuit correct C.C. (V)	Valeur du signal pour circuit fautif C. F. (V_f)	Notation D
0	0	0
0	1	D'
1	0	D
1	1	1

$D' \Leftrightarrow \overline{D}$

- Tables de vérités en notation D



AND	0	1	D	D'
0	0	0	0	0
1	0	1	D	D'
D	0	D	D	0
D'	0	D'	0	D'



OR	0	1	D	D'
0	0	1	D	D'
1	1	1	1	1
D	D	1	D	1
D'	D'	1	1	D'

NOT	0	1	D
	1	0	D'

Algorithme D (suite)

- L'algorithme D fonctionne en trois phases :
  - 1 - Phase de « setup » (activation locale)
    - On assigne les entrées de la porte pour ramener le signal fautif à la valeur opposée du collage
  - 2 - Phase de propagation
    - On propage la faute jusqu'aux sorties primaires
  - 3 - Phase de justification
    - On remonte des valeurs imposées par les deux premières phase sur les signaux internes jusqu'aux entrées primaires

Algorithme D (suite)

- Chacune des trois phases se décompose en deux étapes :
  - 1 - Étape d'affectation
    - C'est l'étape pendant laquelle on fixe des valeurs aux entrées des portes pour contrôler ou propager un autre signal
  - 2- Étape d'implication
    - C'est l'étape pendant la quelle on simule l'effet des signaux imposées dans l'étape d'affectation sur les autres signaux du circuit (on met à jour les signaux du circuit)
- Problèmes de conflits sur les signaux

Problèmes de conflits

- En cas de conflit sur les signaux entre les valeurs à imposer et les valeurs déjà assignées, on revient au dernier choix effectué (pour choisir une autre combinaison des entrées de la porte en question) dans l'ordre suivant :
  - 1 - Phase de justification
  - 2 - Phase de propagation
  - 3 - Phase de « setup »
- S 'il ne reste plus de choix possible
  - => La faute est indétectable

Phase de setup

- Définition :

  C'est la première phase de génération, elle représente le processus pendant lequel on crée la faute (signal D)

- Exemple : F=S@0

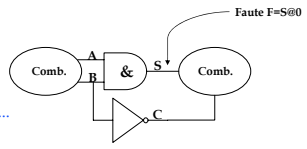
  Setup :

  Affectation :

  => A=1 ; B=1

  Implication :

  => S=D ; C=0 ; …...

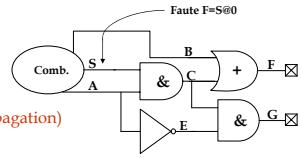


Phase de propagation

- Définition :

  C'est la deuxième phase de génération, elle représente le processus par lequel on propage la faute (signal D) jusqu'aux sorties primaires

- Exemple : F=S@0
  - Propagation : (S=D)
    - Affec. 1 : A=1
    - Impl. 1 : C=D ; E=0 ; G=0
    - Affec. 2 : B=0
    - Impl. 2 : F=D (fin de propagation)



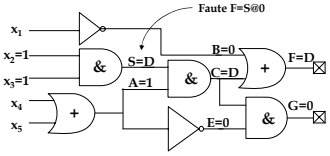
Phase de propagation (suite)

- Algorithme de propagation :
  - procédure propager (S : signal ; V= D ou D')
    - begin
    - if (S est une sortie primaire)
    - return ;
    - else
    - trouver une porte P qui a le signal S comme entrée et U en sortie
    - trouver une combinaison des entrée de P propageant V
    - simuler le circuit //phase d'implication
    - propager (P , V=D ou D')
    - pour chaque signal d'entrée Si de la porte P
    - justifier (Si , Vi)
    - end if
    - end

Phase de justification

- Définition :
  - C'est la troisième phase de génération, elle représente le processus par lequel on remonte jusqu'aux entrées primaires à partir des signaux imposés pendant les deux premières phases

- Exemple : F=S@0
  - Justification :
    - A=1 => x_4=1 et x_5=x
    - ou x_5=1 et x_4=x
    - B=0 => x_1=1

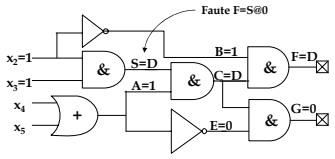


Phase de justification (suite)

- Algorithme de justification :
  - procédure justifier (S : signal ; V= 0 ou 1)
    - **begin**
    - **if** (S est une entrée primaire)
    - **return** ;
    - **else**
    - trouver la porte P qui a le signal S comme sortie
    - trouver une combinaison des entrée de P permettant d 'avoir la valeur V en sortie de la porte
    - simuler le circuit //phase d'implication
    - pour chaque signal d'entrée S_i de la porte P
    - justifier (S_i , V_i)
    - **end if**
    - **end**

Problème de reconvergence

- Définition :
  - La reconvergence des signaux de sortie des portes ayant un fanout > 1 crée des conflits dans les phases de propagation et de justification
- Exemple :
  - Setup :
    - S=1 => x_2=1 et x_3=1
  - Propagation :
    - A=1 ; B=1
  - Justification :
    - A=1 => x_4=1 ou x_5=1
    - B=1 => x_2=0 (conflit)



Algorithme D

- Inconvénients :
  - Nombre très élevé de justifications créant des conflits
    - Choix aléatoire des combinaisons des entrées permettant de justifier la valeur souhaitée
    - Choix aléatoire de la porte qui va propager la faute jusqu'au sorties primaires
  - Complexité exponentielle
- Amélioration :
  - Algorithme PODEM
  - Algorithme FAN
  - Autres algorithmes