Institut Supérieur des Sciences Appliquées
et de Technologie de Sousse (ISSAT)

Mastère de recherche sûreté et sécurité des systèmes industriels

Option :Électronique et Informatique temps réel

Projet de mastère :

Étude et implémentation d'une unité arithmétique auto-contrôlable

par:

Khedhiri Chiraz

Sous la direction de:

Mr. Hamdi Belgacem

Plan de l'exposé

- **Introduction générale**
- **I - Le test des circuits intégrés numériques**
- **II - Les unités arithmétiques standards (non self-checking)**
- **III - Implémentation des unités arithmétiques self-checking**
- **VI – Simulation de pannes**
- **Conclusion et perspectives**

Plan de l'exposé

- **Introduction générale**
- **I - Le test des circuits intégrés numériques**
- **II - Les unités arithmétiques standards (non self-checking)**
- **III - Implémentation des unités arithmétiques self-checking**
- **VI – Simulation de pannes**
- **Conclusion et perspectives**

Introduction générale

L'évolution rapide de la microélectronique liée aux progrès réalisés dans le domaine de la technologie empêche désormais les techniques de prévention couramment utilisées jusqu'à aujourd'hui (test externe) de couvrir l'ensemble des pannes possibles dans les circuits intégrés logiques. Cela est d'autant plus critique lorsque l'application est dite de sécurité car la défaillance d'un système peut être à l'origine de situations catastrophiques pouvant entraîner la mort.

Introduction générale

Dans ce cas, il est nécessaire de signaler la présence de la première sortie erronée. Cette mission n'est en aucune façon remplie par les techniques classiques de test.

Pour satisfaire un tel besoin, on a vu se développer au fil des années le domaine du test en ligne intégré dont l'objectif est de développer des systèmes à défaillance sûre (**fail safe**), qui passe par l'étude et la réalisation des systèmes auto-contrôlables.

Introduction générale

Notre travail a pour but d'étudier et de réaliser une unité arithmétique auto-contrôlable en technologie CMOS 70nm.

L'outil CAO utilisé est le logiciel Microwind 3.1.

Plan de l'exposé

- **Introduction générale**
- **I - Le test des circuits intégrés numériques**
- **II - Les unités arithmétiques standards (non self-checking)**
- **III - Implémentation des unités arithmétiques self-checking**
- **VI – Simulation de pannes**
- **Conclusion et perspectives**

I - Le test des circuits intégrés numériques

Une faute : est la manifestation d'un défaut physique du circuit.

↓

Des erreurs : qui se caractérisent par des états incorrects.

↓

Des défaillances : qui se caractérisent par une déviation plus ou moins importante du comportement du système par rapport au comportement nominal attendu.

↓

La défaillance devient un risque si elle est succeptible d'entraîner un accident.

I - Le test des circuits intégrés numériques

Les modèles de fautes

**** Stuck-at :*** collage à 0 ou 1 d'une ligne de connexion.

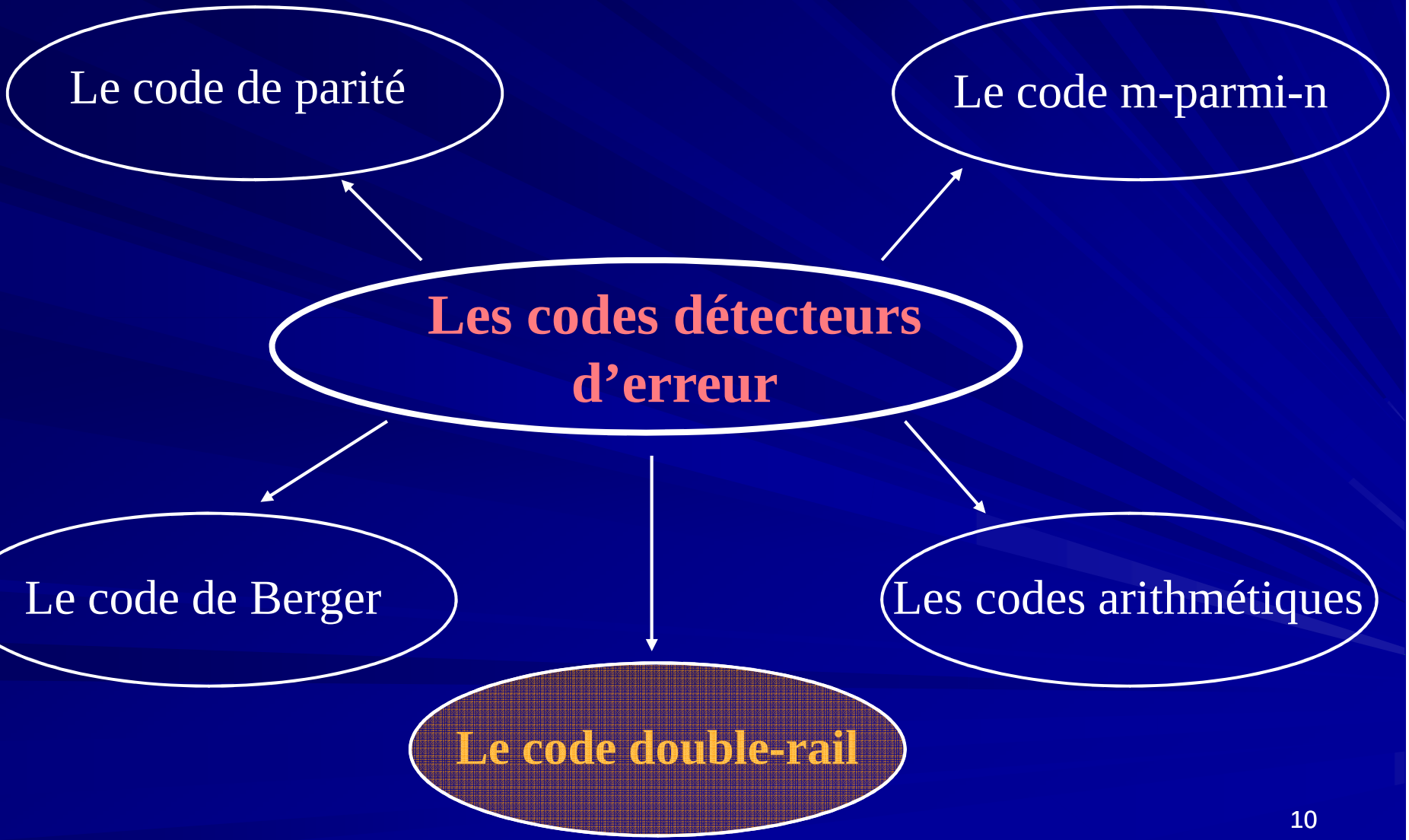
**** Stuck-open :*** collage d'un transistor à l'état bloquant.

**** Stuck-on :*** collage d'un transistor à l'état passant permanent.

**** Delay-fault :*** Ces fautes se traduisent par des erreurs de temps de transfert dans un circuit.

…..

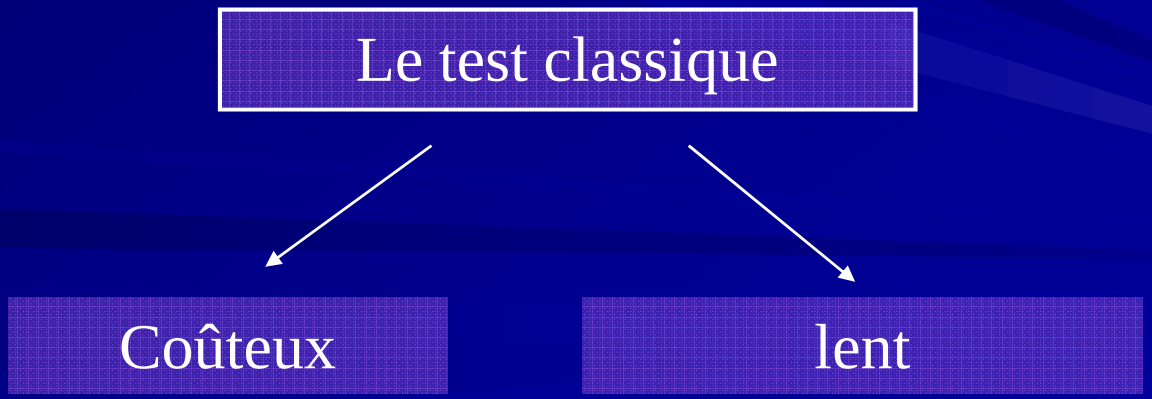
I - Le test des circuits intégrés numériques



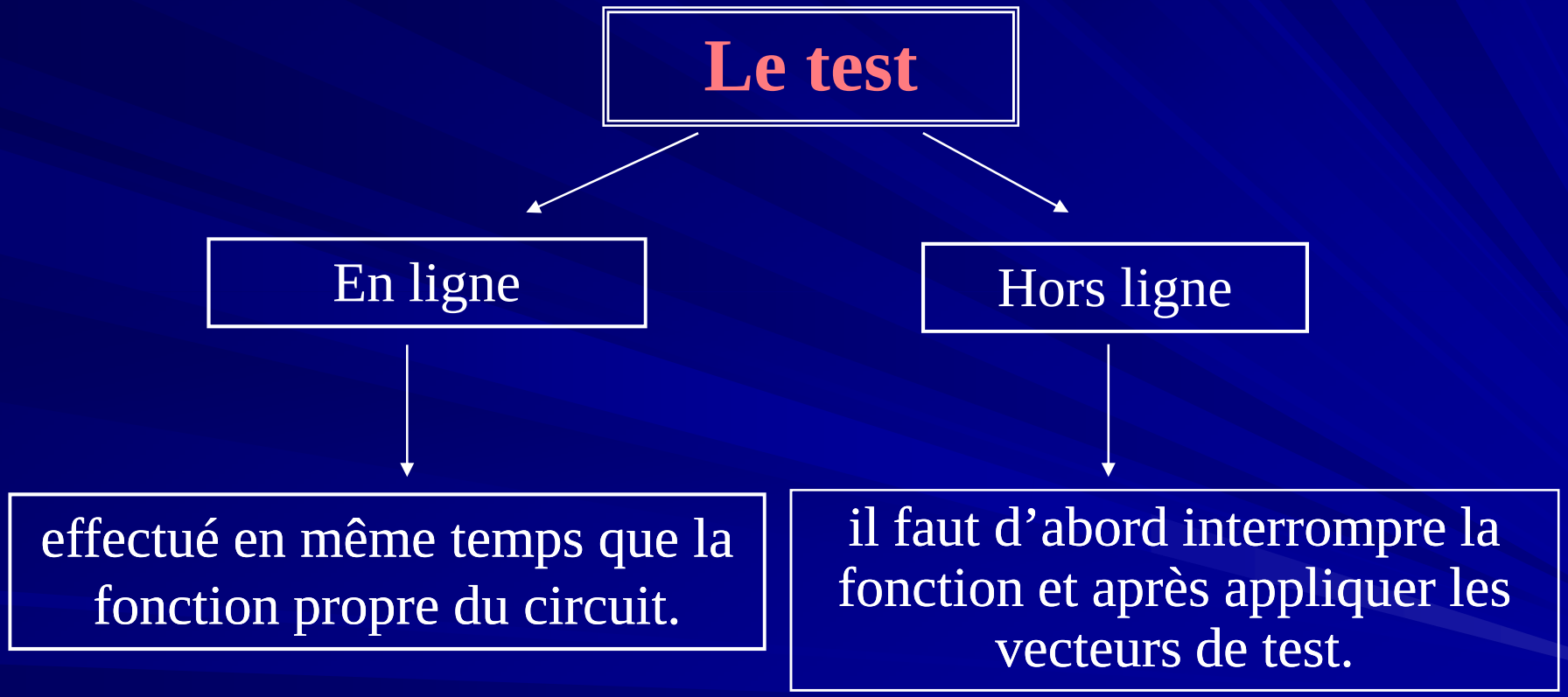
I - Le test des circuits intégrés numériques

Le test classique : La procédure de test consiste à appliquer des vecteurs de test à l'entrée d'un circuit testé. A la sortie on récupère les autres valeurs et on décide si les valeurs obtenues sont correctes ou non. Ça se répète jusqu'à la détection d'une faute ou déclaration que le circuit est bon.

Inconvénient : Pour n entrées nous avons besoin de 2^n combinaisons possibles de vecteurs de test.



I - Le test des circuits intégrés numériques



I - Le test des circuits intégrés numériques

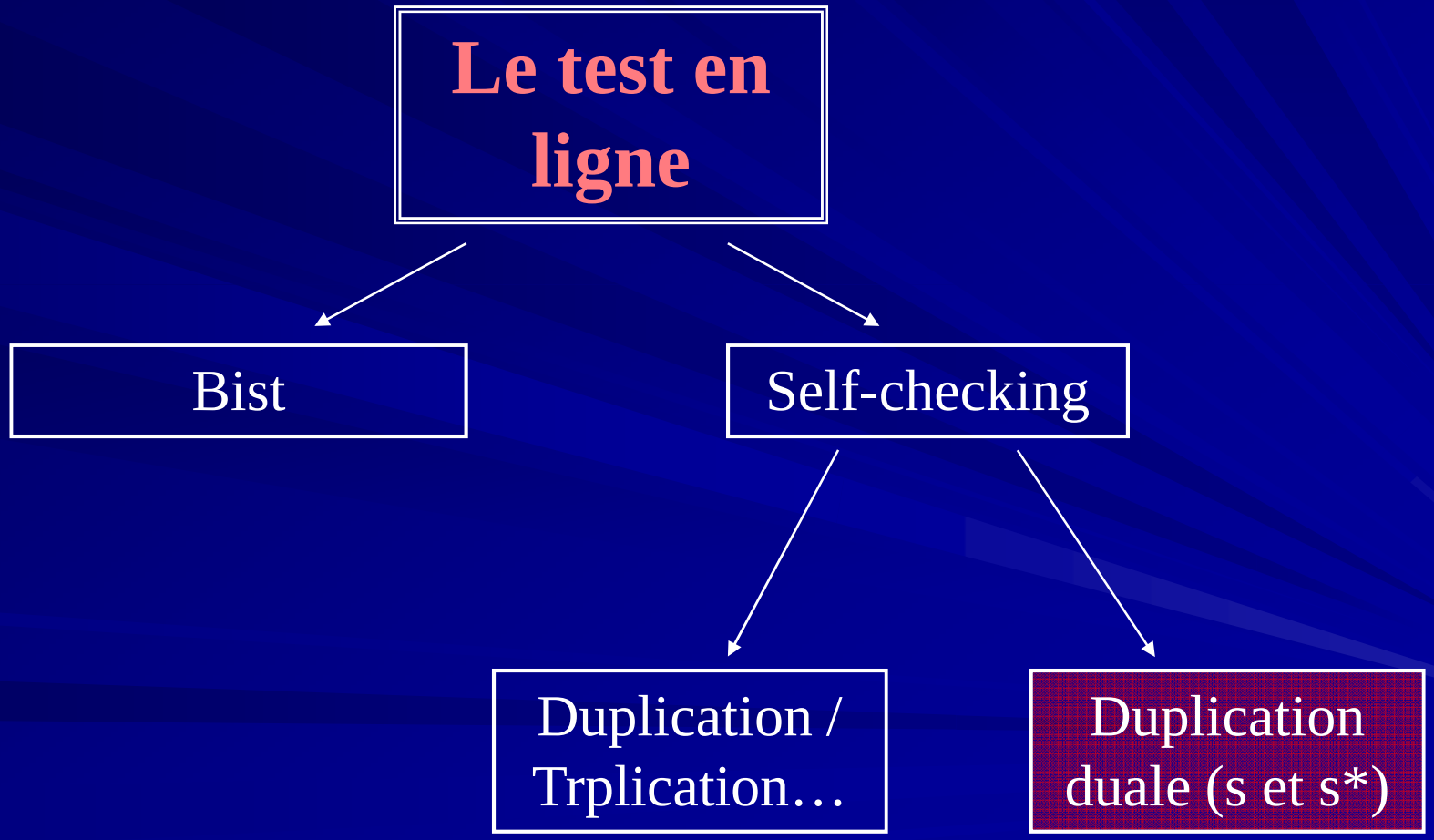
✓ Le test hors ligne ne permet pas de détecter les erreurs aussitôt qu'elles apparaissent.

⇨ Donc une erreur non détectée peut engendrer d'autres erreurs et par suite le circuit peut devenir irréparable.

✓ Le test en ligne permet de détecter les erreurs dès qu'elles apparaissent d'où la possibilité de réparation immédiate.

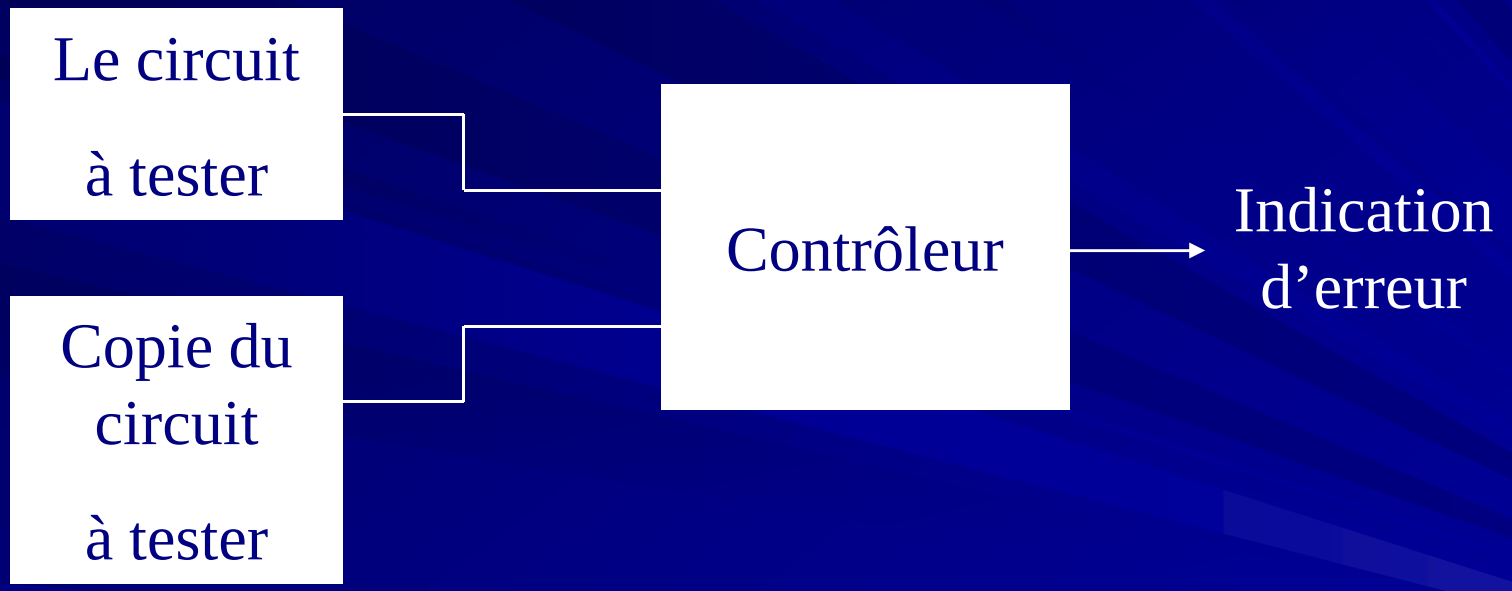
⇨ Le test en ligne est plus efficace que le test hors ligne

I - Le test des circuits intégrés numériques



I - Le test des circuits intégrés numériques

* Le test par duplication



Cette implémentation implique 100 % de surcoût sans compter les contrôleurs. De même, les défauts de conception ne sont pas détectés.

I - Le test des circuits intégrés numériques

* Le test par duplication duale

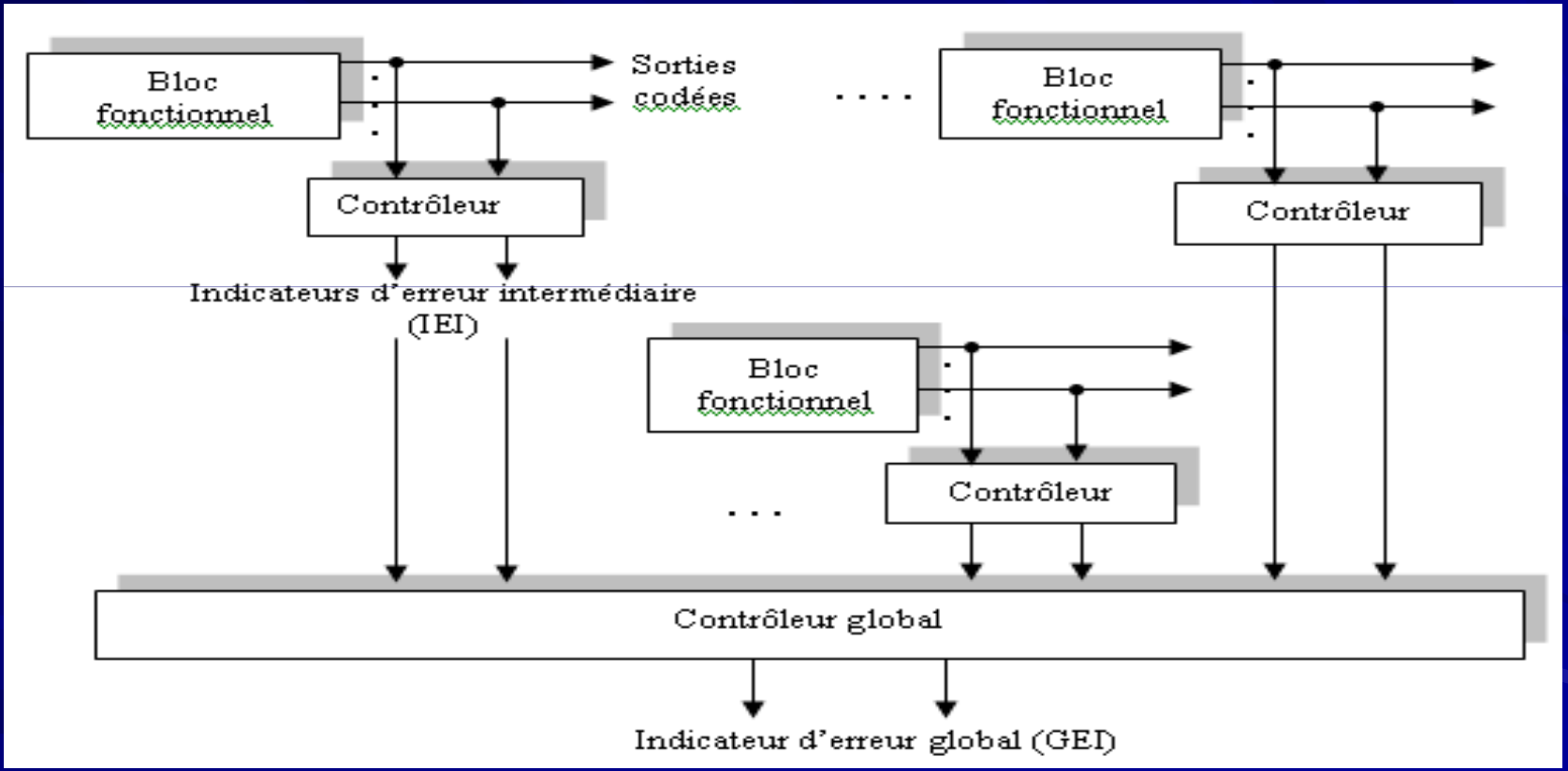


Figure : Structure générale des circuits « self-checking »

I - Le test des circuits intégrés numériques

Le but à atteindre de manière à rendre le circuit « sûr en fonctionnement » est celui du ***« totalement auto contrôlable »*** (TSC : Totally Self Checking) qui s'exprime ainsi : « La première sortie erronée du bloc fonctionnel doit provoquer une indication d'erreur en sortie du contrôleur ».

Les avantages du test par duplication duale

✓ Le test par duplication duale est moins coûteux et plus rapide que le test classique.

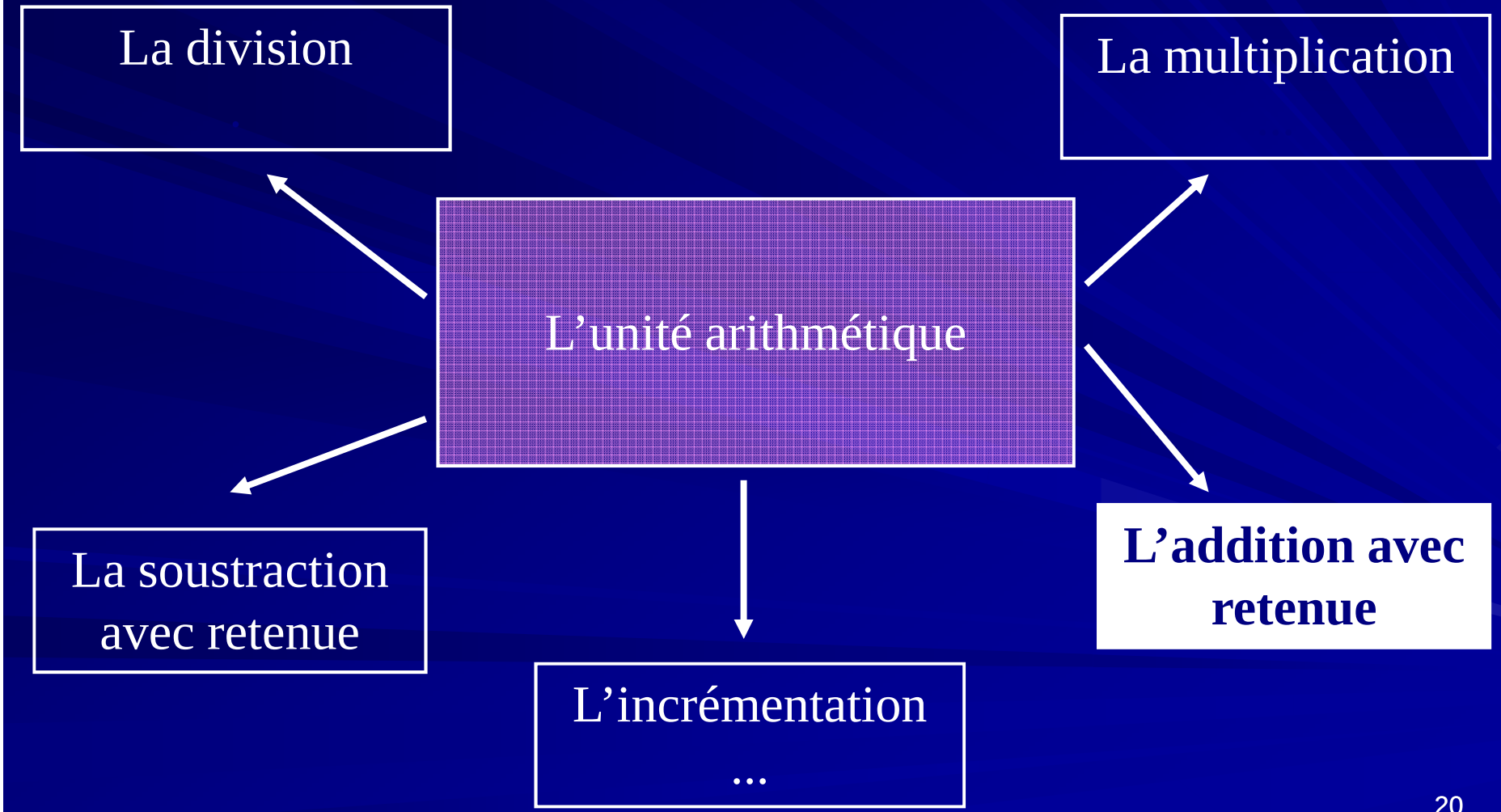
✓ Il permet de détecter les fautes transitoires et permanentes donc il est plus efficace que le test hors ligne.

✓ Il permet un gain en surcoût puisqu'il ne s'agit pas de dupliquer les blocs fonctionnels comme pour le test par duplication.

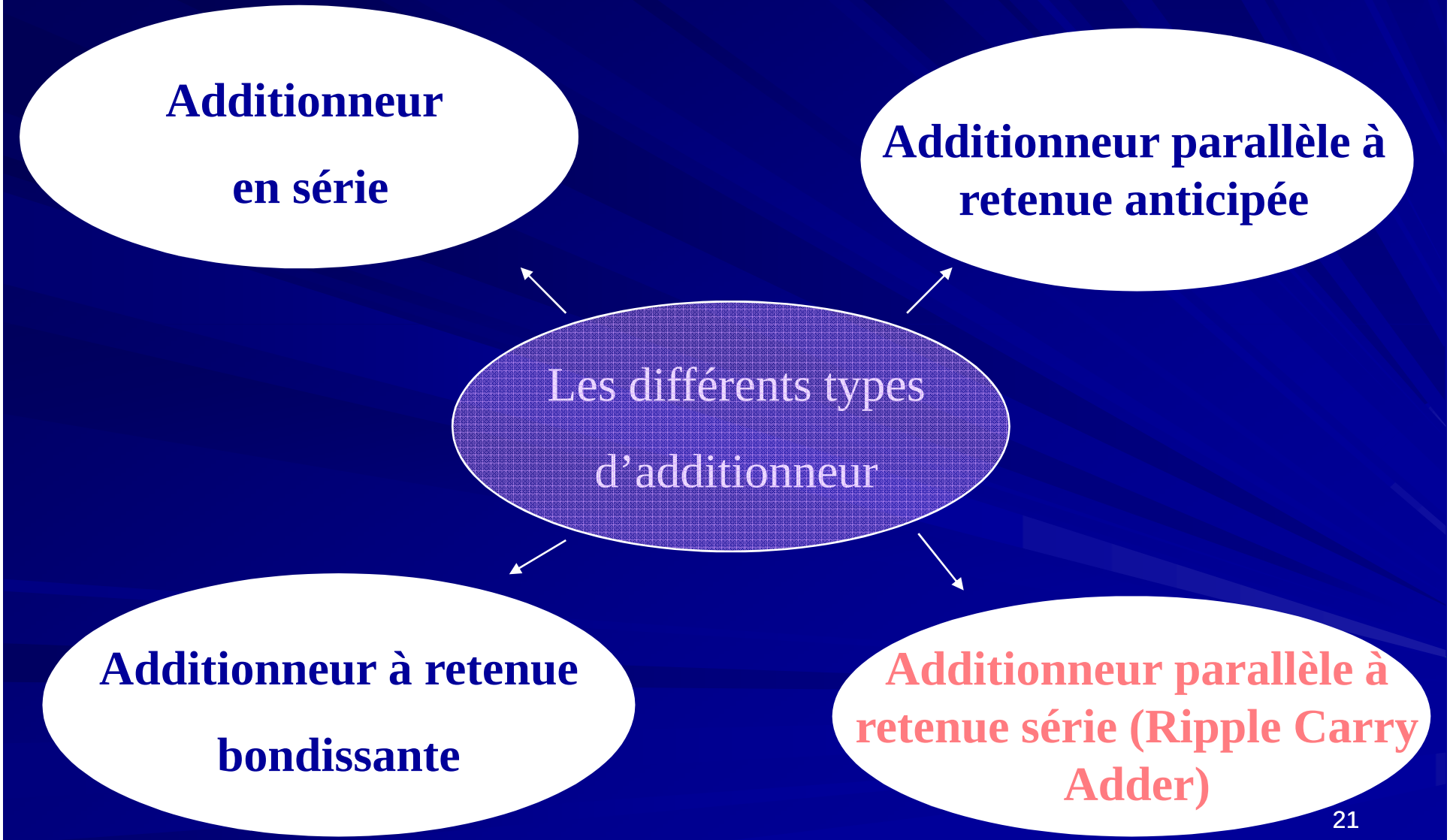
Plan de l'exposé

- **Introduction générale**
- **I - Le test des circuits intégrés numériques**
- **II - Les unités arithmétiques standards (non self-checking)**
- **III - Implémentation des unités arithmétiques self-checking**
- **VI – Simulation de pannes**
- **Conclusion et perspectives**

II - Les unités arithmétiques standards (non self-checking)



II - Les unités arithmétiques standards (non self-checking)



II - Les unités arithmétiques standards (non self-checking)

Additionneur parallèle à retenue série (Ripple Carry Adder)

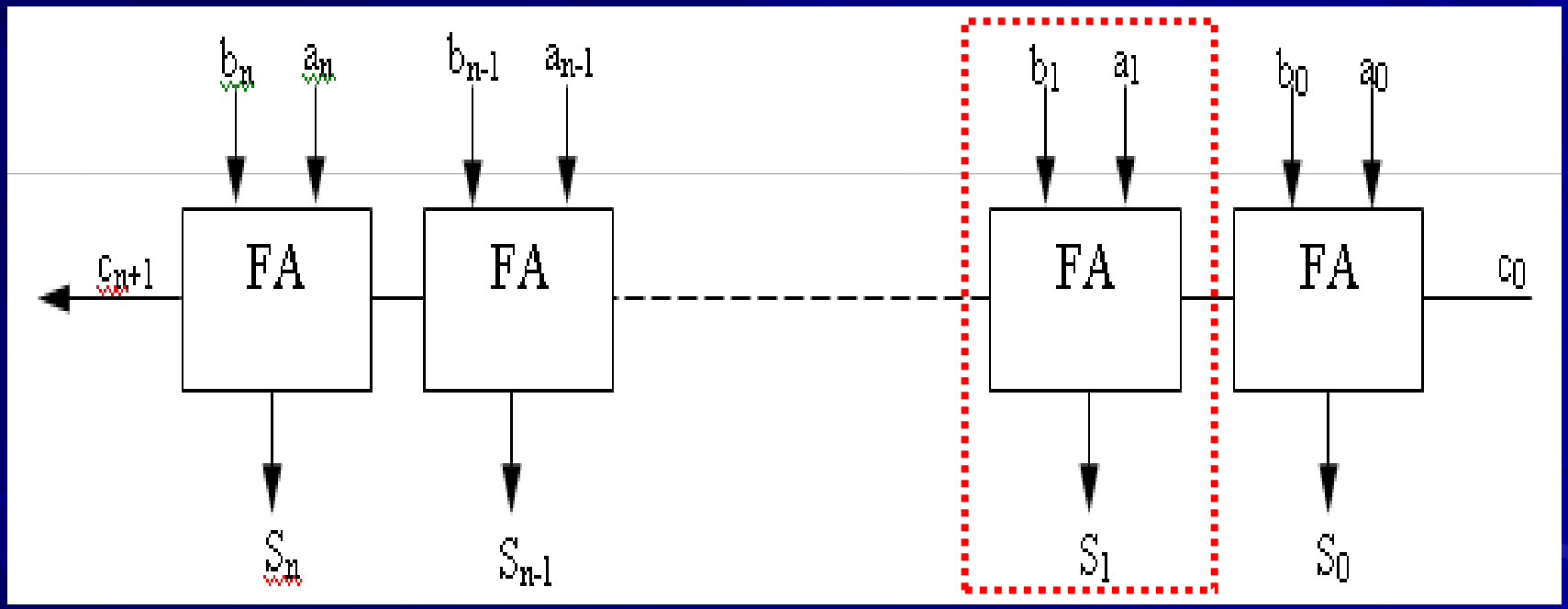


Figure : Additionneur parallèle à retenue série (Ripple Carry Adder)

II - Les unités arithmétiques standards (non self-checking)

* L'additionneur 1 bit

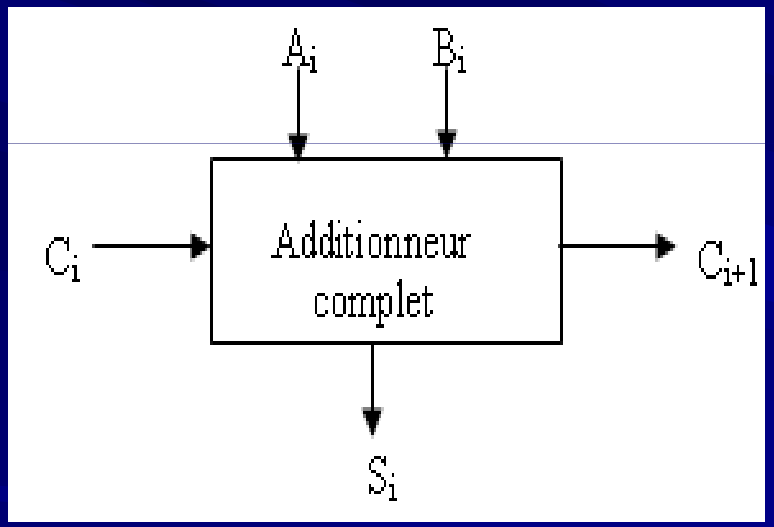


Figure : représentation d'un bit slice de l'additionneur

A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tableau : Table de vérité de l'addition

Ai ⊕ Bi = Pi

II - Les unités arithmétiques standards (non self-checking)

L'expression de la somme et de la retenue peut être écrite sous la forme :

$$Ci+1 = (Ai.Bi) + (Ai \oplus Bi).Ci$$

$$Si = Ai \oplus Bi \oplus Ci$$

Avec :

- ✓ Ai . Bi = Gi : représente la génération de la retenue sortante indépendamment de la retenue rentrante
- ✓ Ai ⊕ Bi = Pi : représente la propagation de la retenue rentrante

Les expressions de la somme et de la retenue deviennent alors :

$Si = Pi \oplus Ci$ et $Ci+1 = Gi + Pi.Ci$

II - Les unités arithmétiques standards (non self-checking)

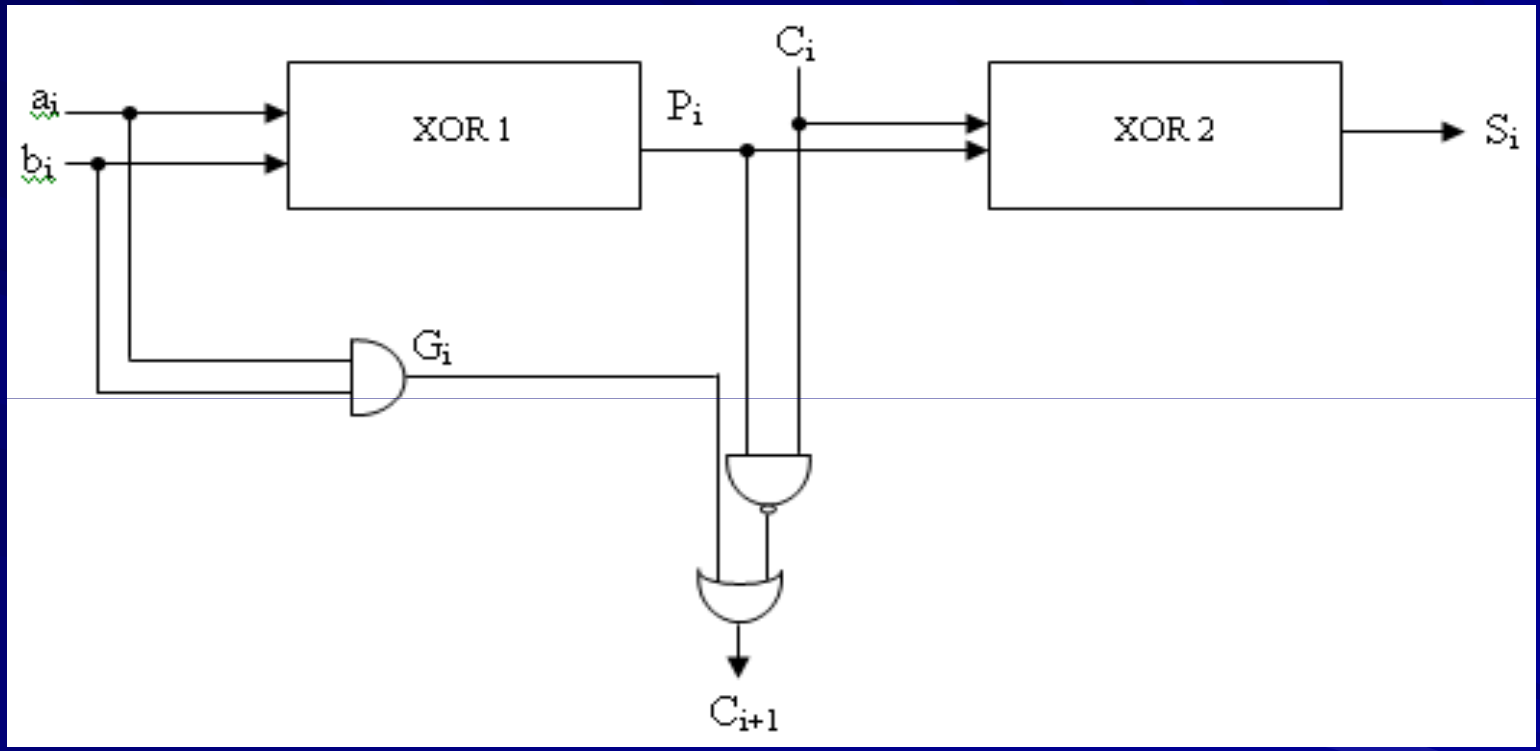


Figure : Le bit slice

II - Les unités arithmétiques standards (non self-checking)

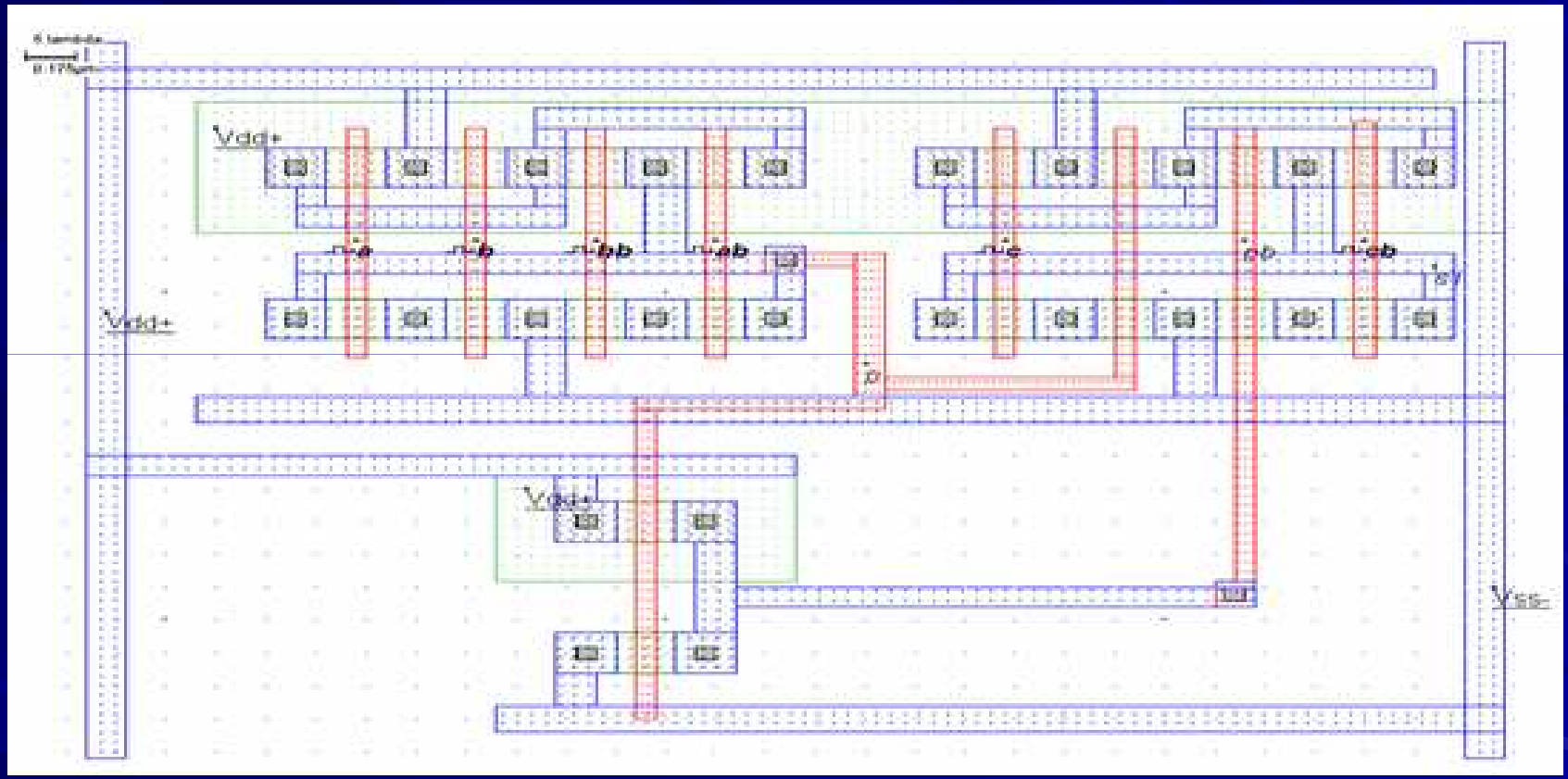


Figure : Le dessin de masque de l'additionneur standard 1 bit en CMOS (70nm)

II - Les unités arithmétiques standards (non self-checking)

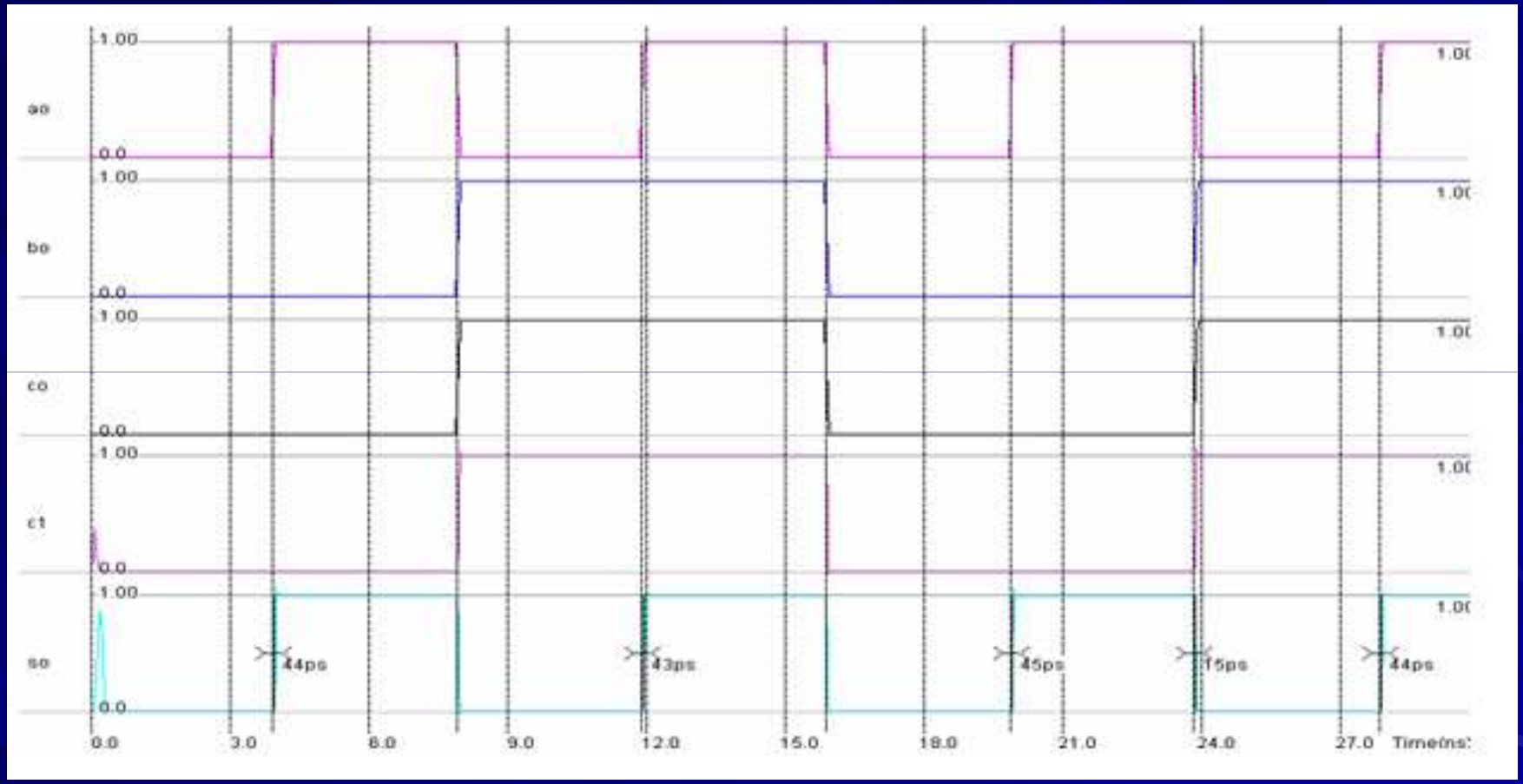


Figure : Simulation SPICE de l'additionneur standard de la figure précédente

II - Les unités arithmétiques standards (non self-checking)

* L'additionneur 8 bits

L'additionneur 8 bits est obtenu par la mise en cascade de 8 tranches 1 bit : c'est l'addition parallèle à retenue série.

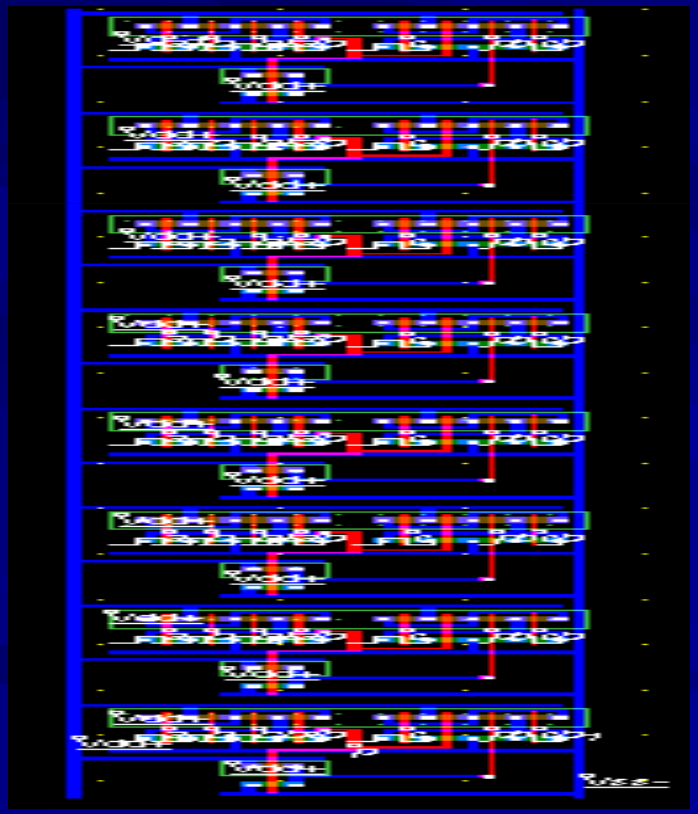


Figure : Le dessin de masques (Layout) d'un additionneur 8 bits

Plan de l'exposé

- **Introduction générale**
- **I - Le test des circuits intégrés numériques**
- **II - Les unités arithmétiques standards (non self-checking)**
- **III - Implémentation des unités arithmétiques self-checking**
- **VI – Simulation de pannes**
- **Conclusion et perspectives**

III - Implémentation des unités arithmétiques self-checking

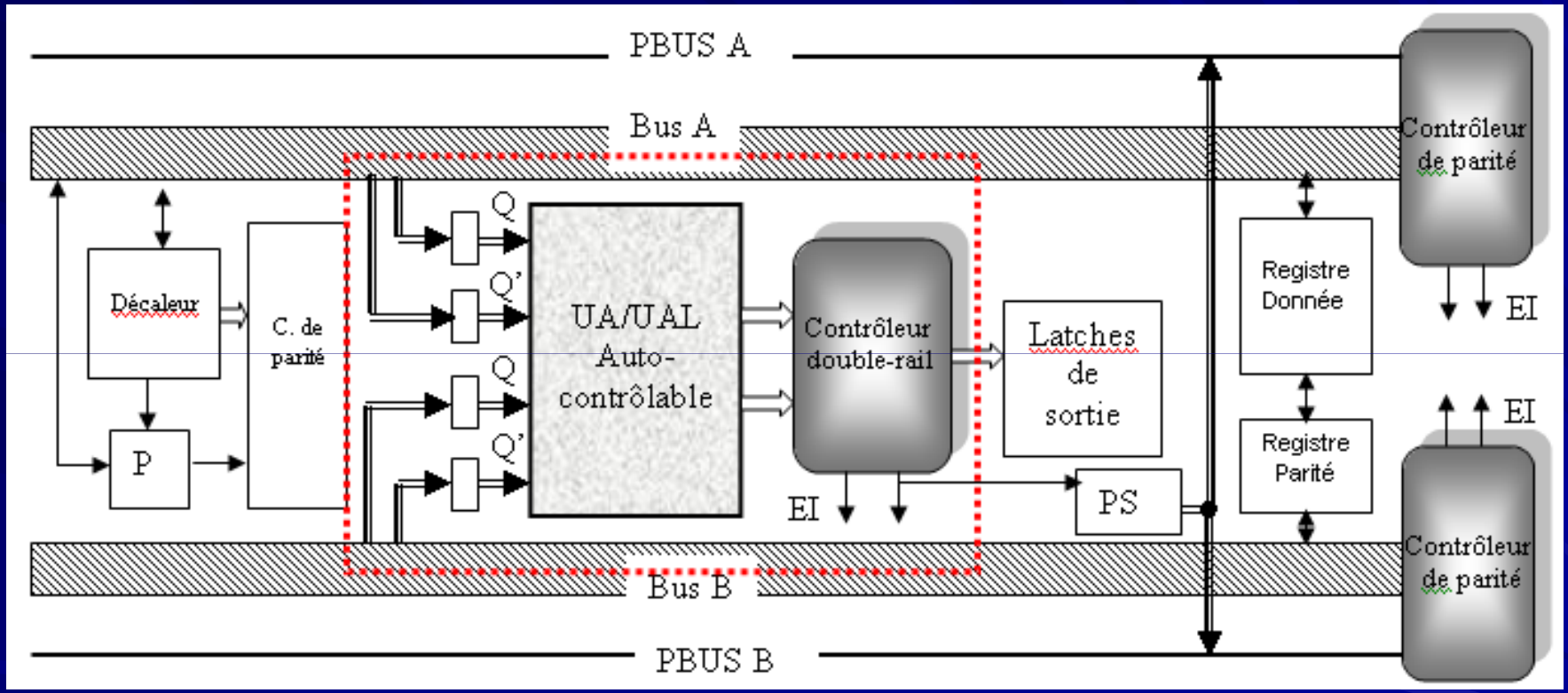


Figure : Structure générale des parties opératives auto-contrôlables : Contrôle de la Sortie/Génération de Parité (CS/GP)

III - Implémentation des unités arithmétiques self-checking

Les cellules de base des UAs

1- Le bit slice

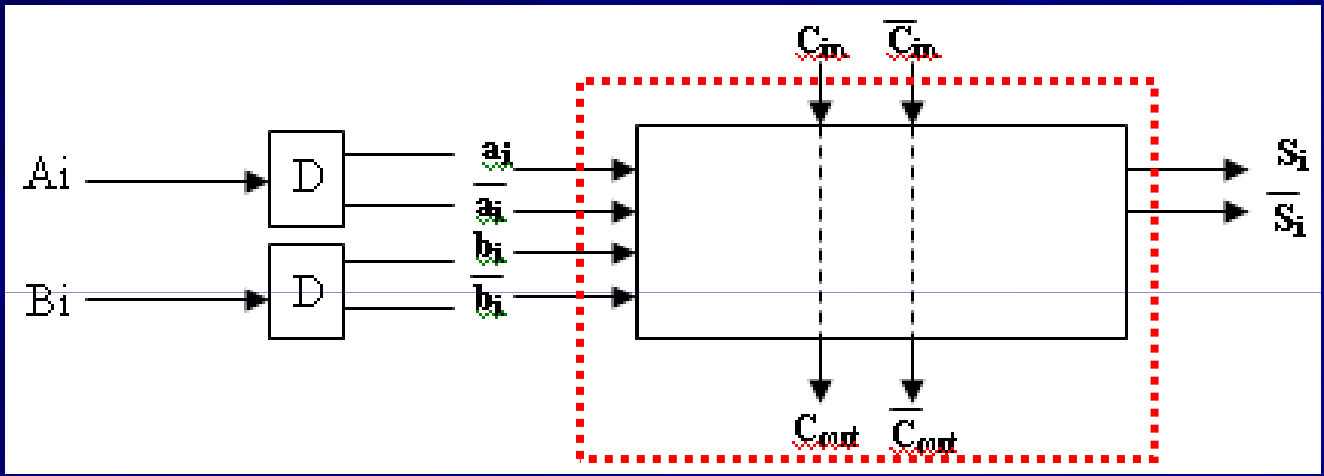


Figure : Tranche d'un bit d'additionneur à propagation de la retenue

✓ Le bit slice reçoit les entrées double rail et génère des sorties double rail. Les données qui arrivent du bus passent par un registre d'entrée (formé par de D latch) qui les présentera à l'additionneur sous forme duale (Qi et Qi*).

III - Implémentation des unités arithmétiques self-checking

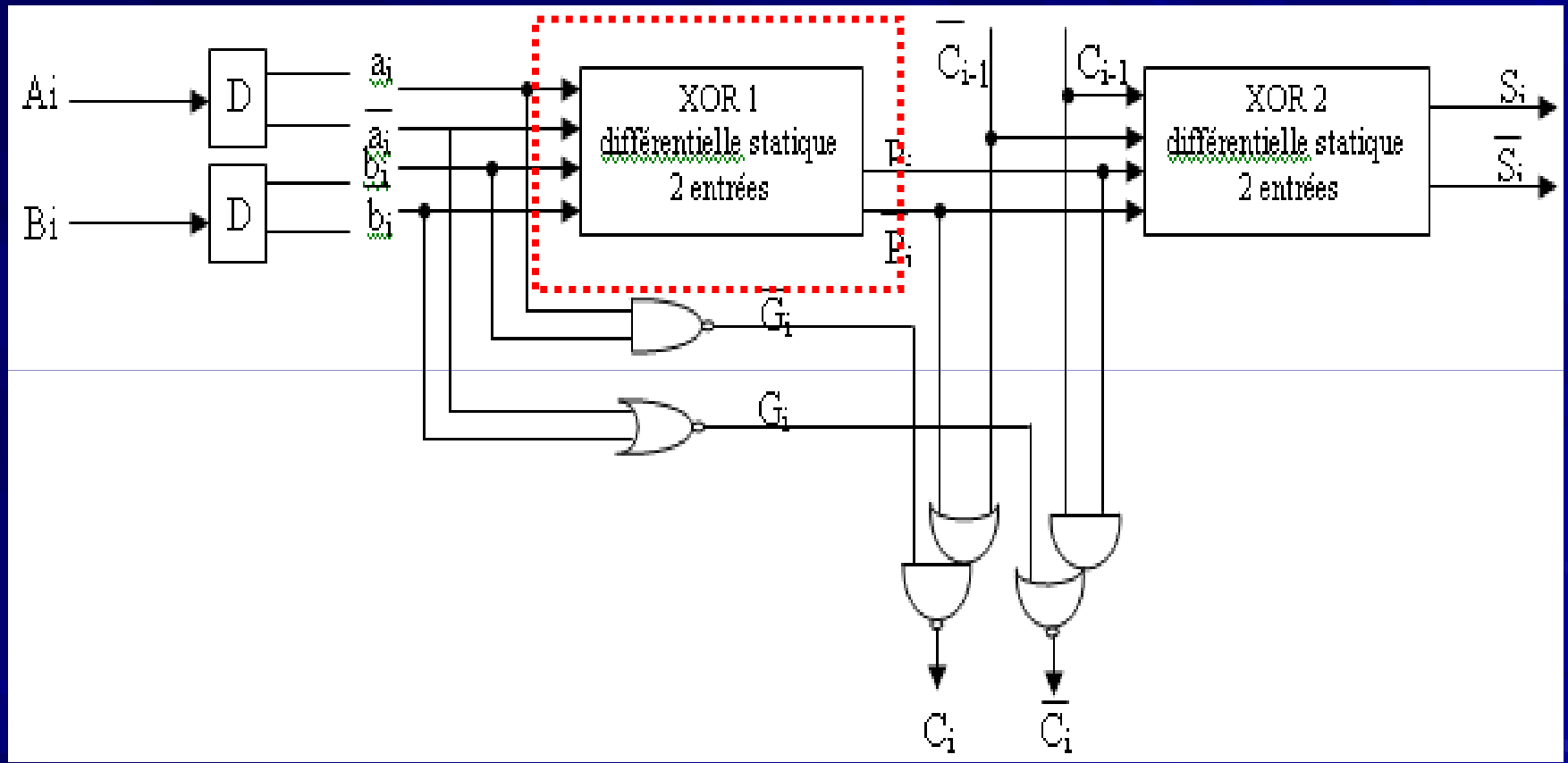


Figure : Additionneur à retenue propagée (40 Transistors)

III - Implémentation des unités arithmétiques self-checking

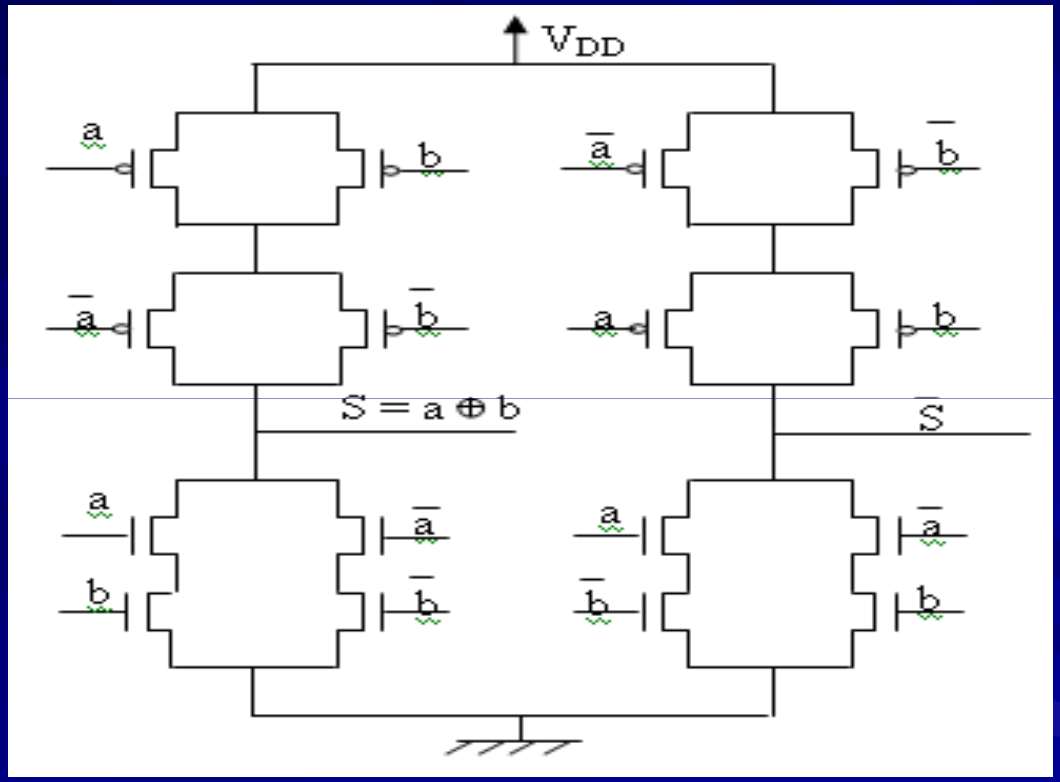


Figure : La XOR différentielle

Cette structure comporte 16 transistors.

III - Implémentation des unités arithmétiques self-checking

Une structure plus simple et qui comporte moins de transistor (12 transistors) a été introduite.

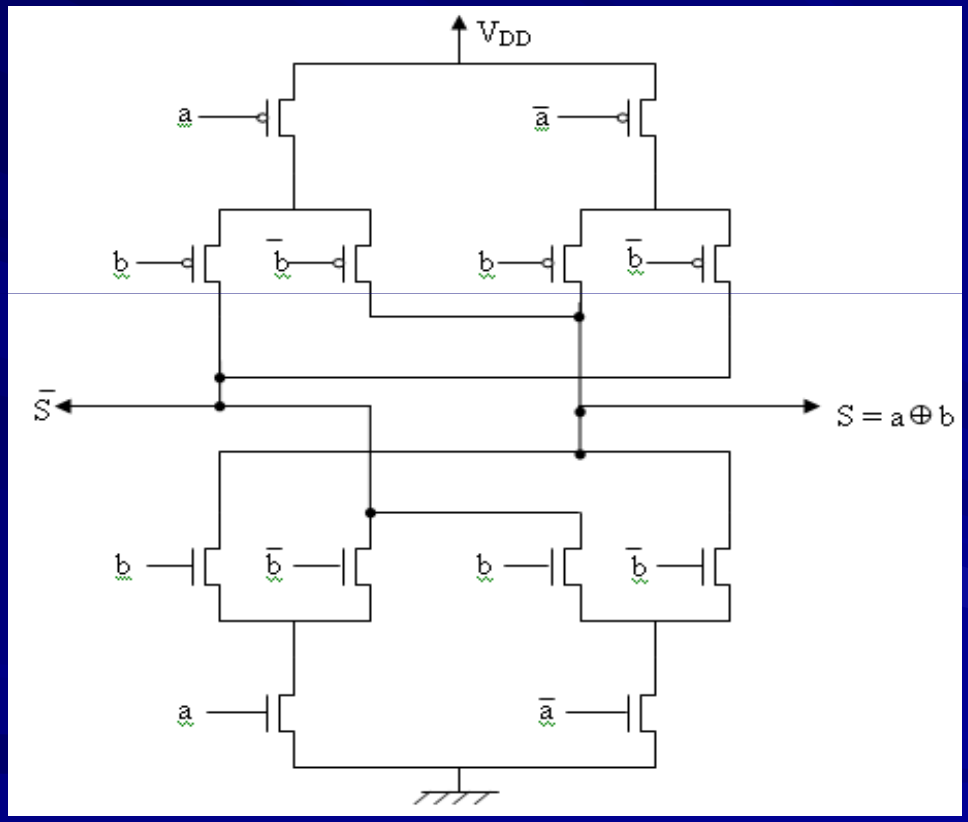


Figure : Schéma de la XOR différentielle statique à deux entrées

III - Implémentation des unités arithmétiques self-checking

Dans ce qui suit nous allons utiliser la XOR différentielle modifiée puisqu'elle comporte moins de transistor donc elle est moins coûteuse.

III - Implémentation des unités arithmétiques self-checking

2- Le contrôleur double-rail

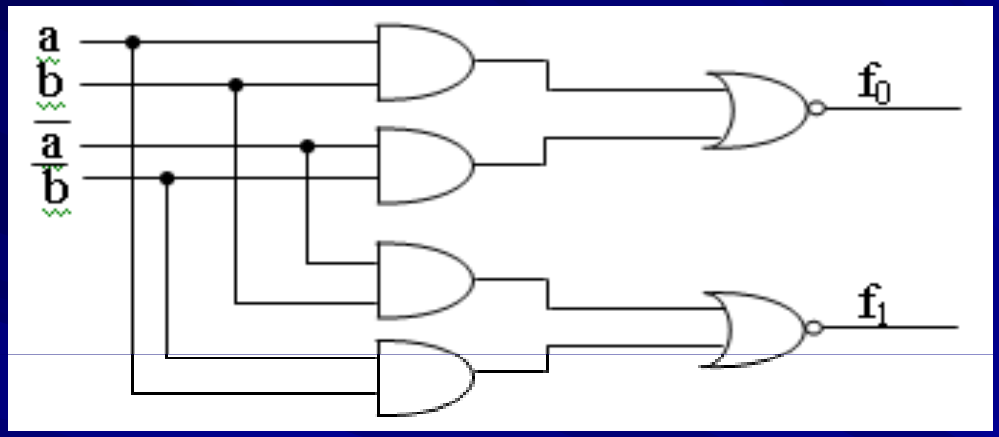


Figure : Contrôleur double-rail à deux entrées

On peut remarquer que : $f_0 = a \oplus b$ et $f_1 = \overline{a \oplus b}$

Le codage des sorties du contrôleur est également de type « double-rail », c'est-à-dire que les sorties (f0, f1) ϵ { (1,0), (0,1)} représentent le bon fonctionnement et que les sorties (f0, f1) ϵ { (0,0), (1,1)} sont données en cas de mauvais fonctionnement.

III - Implémentation des unités arithmétiques self-checking

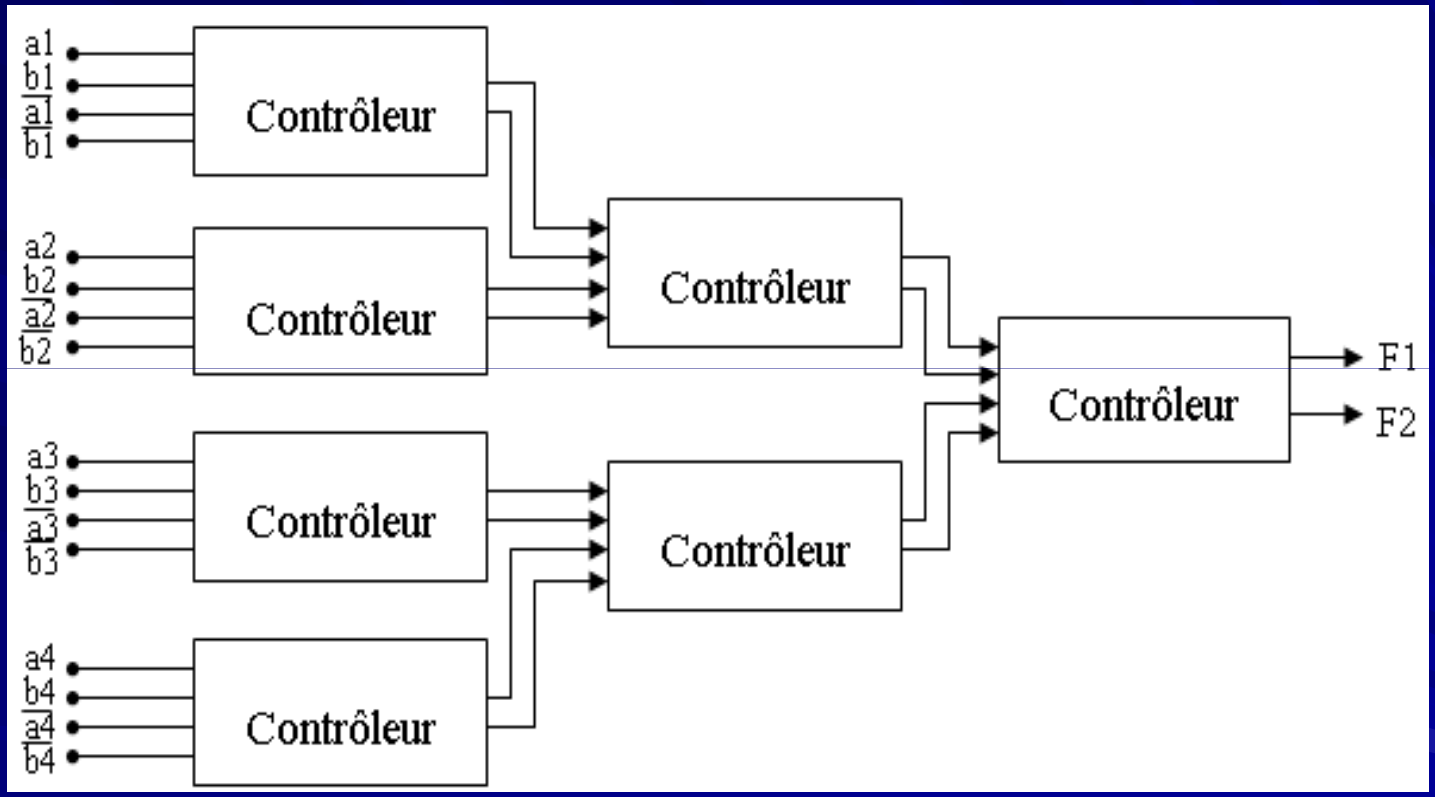


Figure : Contrôleur double-rail à huit entrées

III - Implémentation des unités arithmétiques self-checking

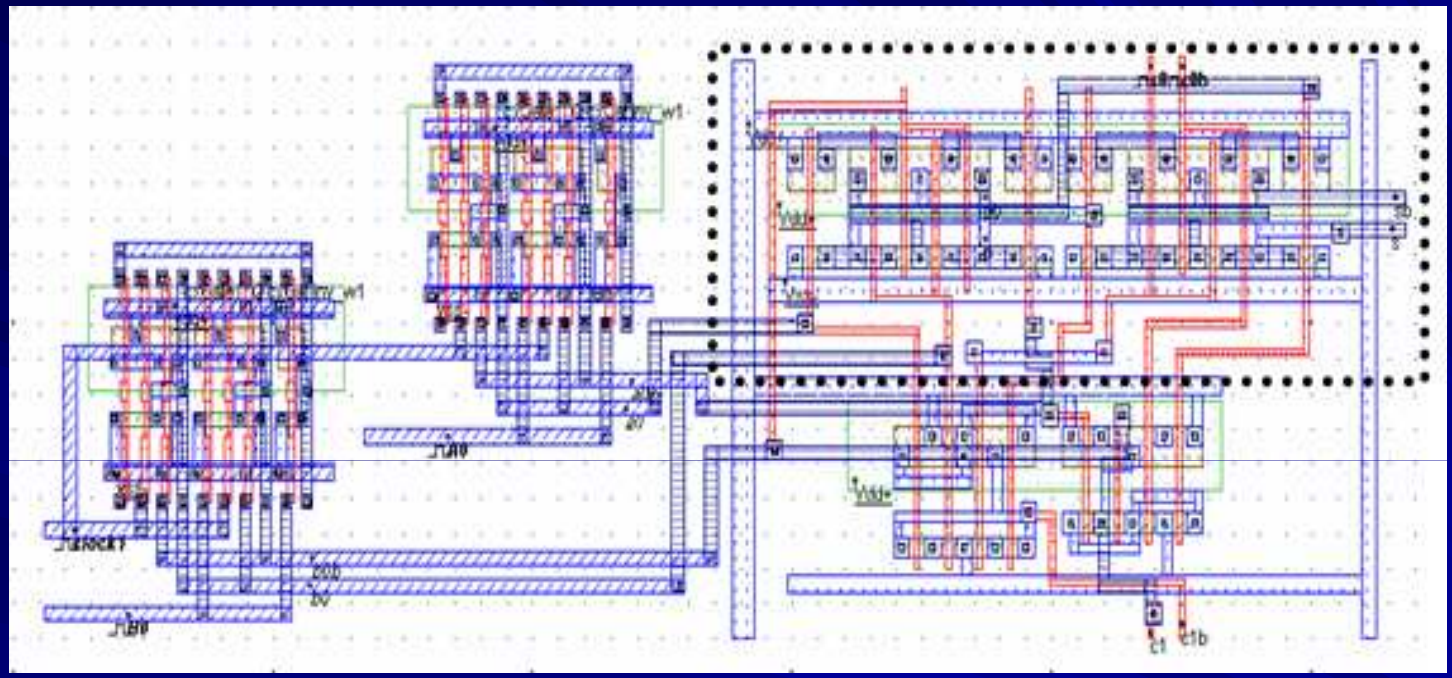


Figure : Le dessin de masques du bit slice

✓ La partie encadrée en pointillé représente les deux cellules de XOR différentielle et la partie en bas constitue la cellule de génération de la retenue.

✓ Pour valider ce dessin électriquement, des simulations SPICE ont été effectuées avec succès.

III - Implémentation des unités arithmétiques self-checking

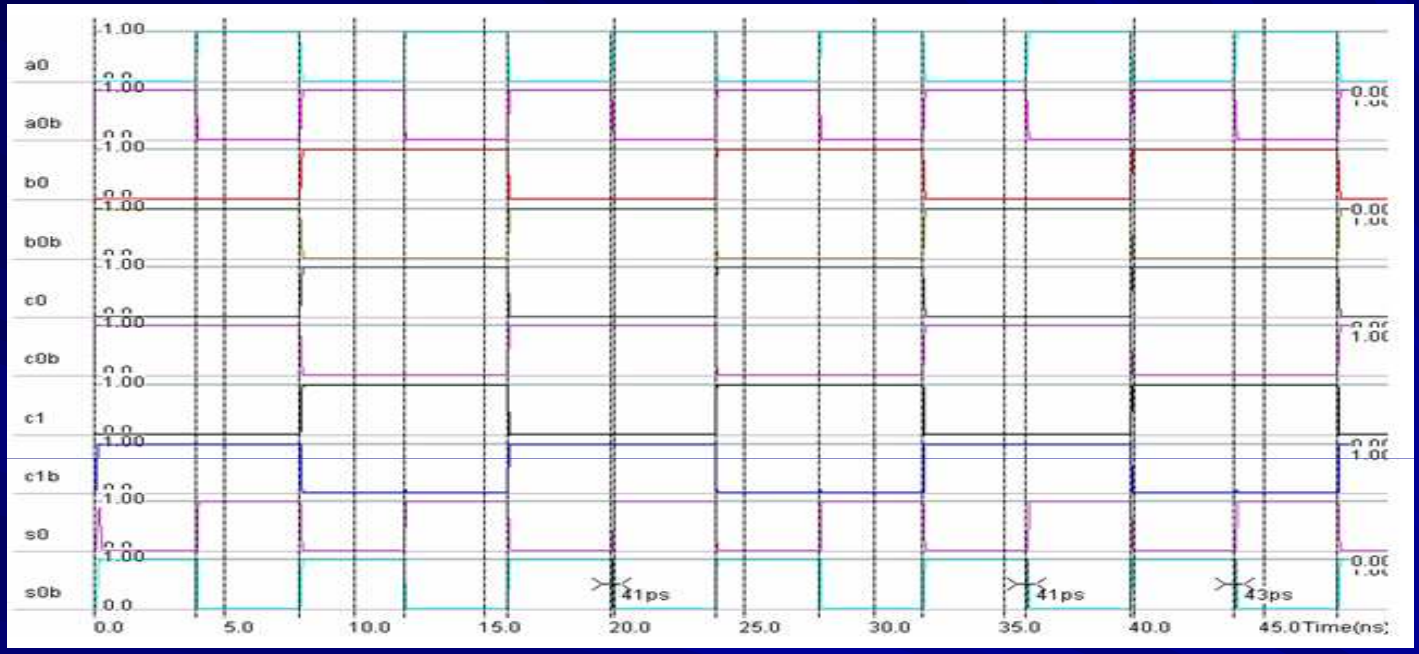


Figure : Une simulation SPICE de l'additionneur de la figure précédente

Commentaire : On remarque bien que le résultat de simulation nous a donné des sorties complémentaires puisque le circuit est simulé sans défaut. (a et ab sont complémentaires et pas de défaut dans le dessin). Aussi, on peut vérifier que :

s0 = a0 + b0 + c0 et **c1= majorité (a0, b0, c0)**

III - Implémentation des unités arithmétiques self-checking

L'additionneur huit bits

✓ Maintenant on va prendre huit cellules du bit slice et des cellules de contrôleur double-rail pour construire un additionneur auto-contrôlable 8 bits.

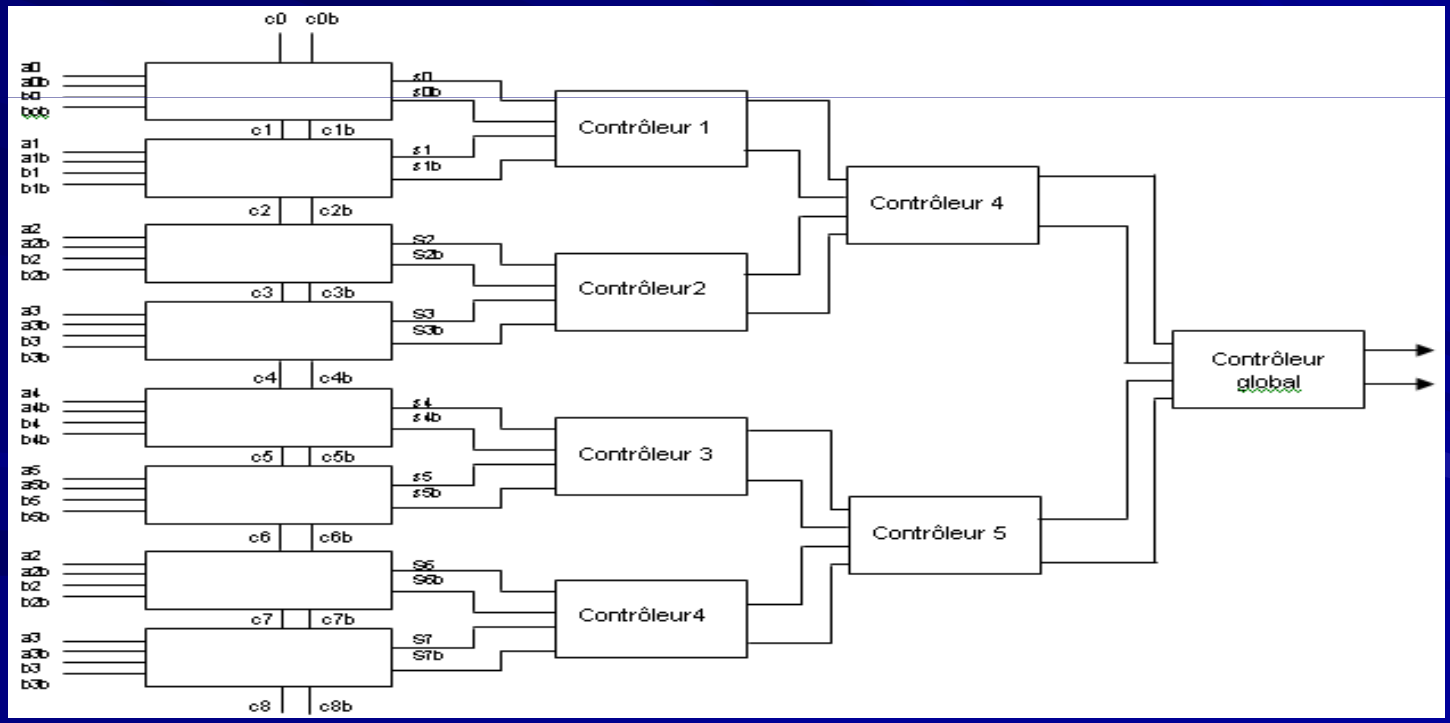


Figure : l'additionneur auto-contrôlable 8 bits.

III - Implémentation des unités arithmétiques self-checking

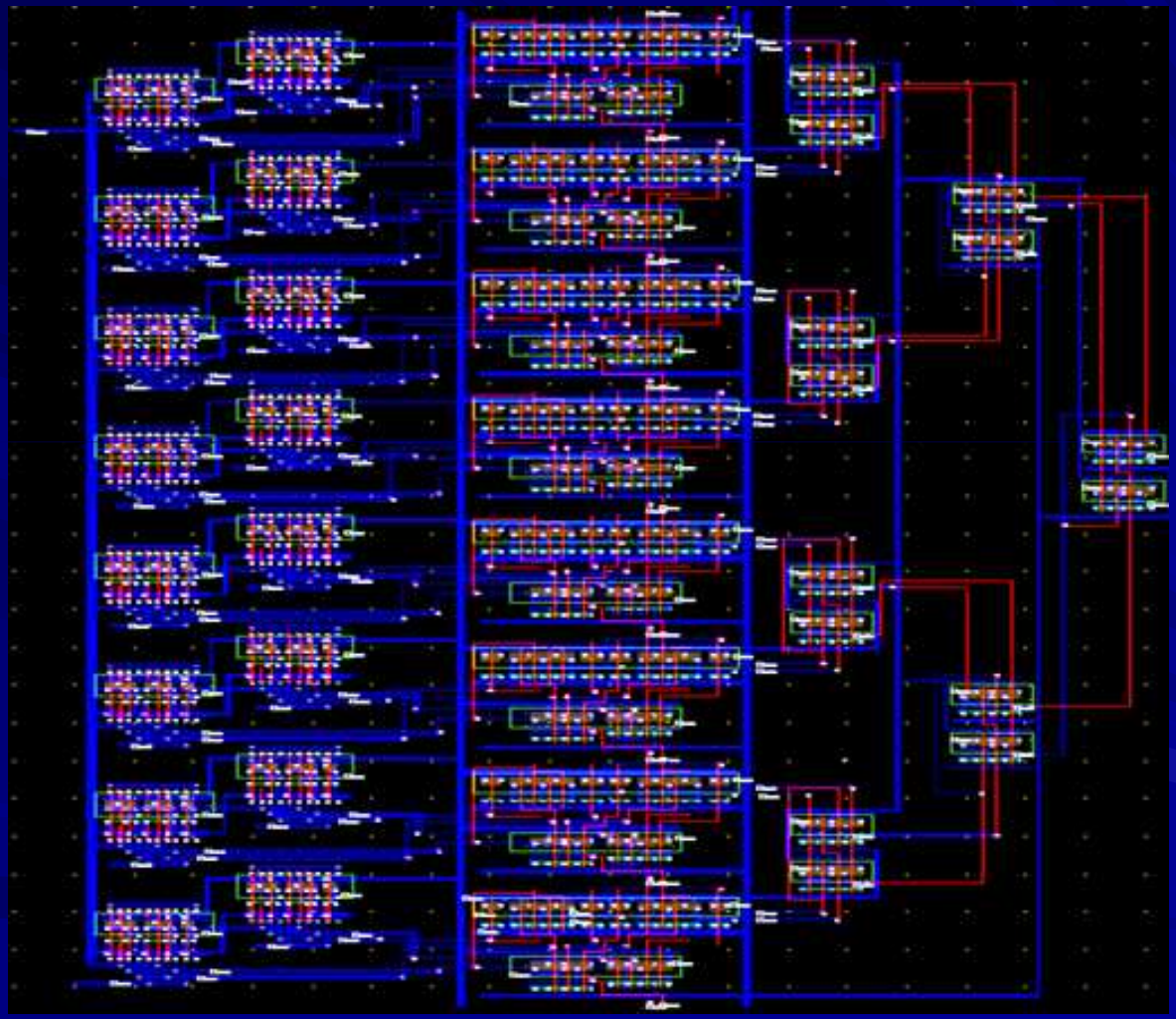


Figure : Le dessin de masque de l'additionneur 8 bits auto-contrôlable en CMOS 70nm

III - Implémentation des unités arithmétiques self-checking

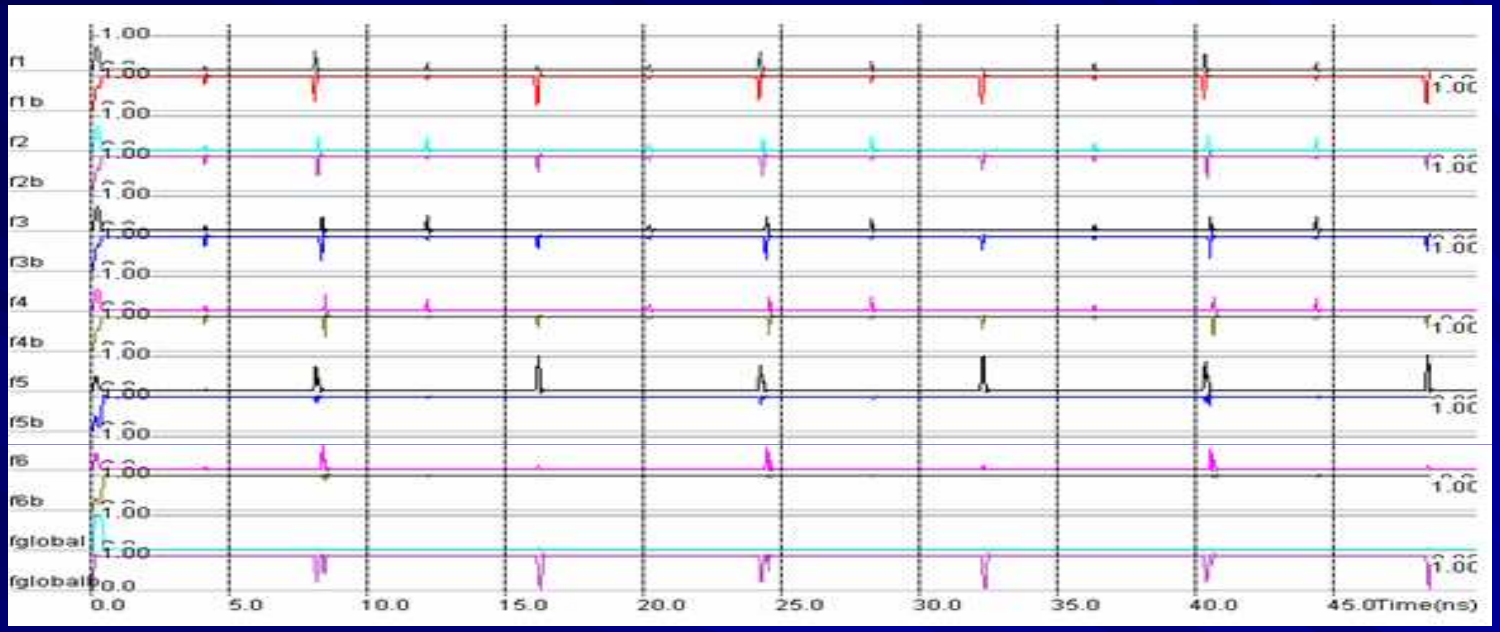


Figure : Une simulation SPICE de l'additionneur 8 bits sans défauts

Commentaire : Les sorties des différents contrôleurs appartiennent au code correct ((fi,fib) Є {01, 10}). Dans ce cas, nous n'avons pas simulé de pannes donc les contrôleurs fonctionnent correctement et ne signalent aucune erreur.

III - Implémentation des unités arithmétiques self-checking

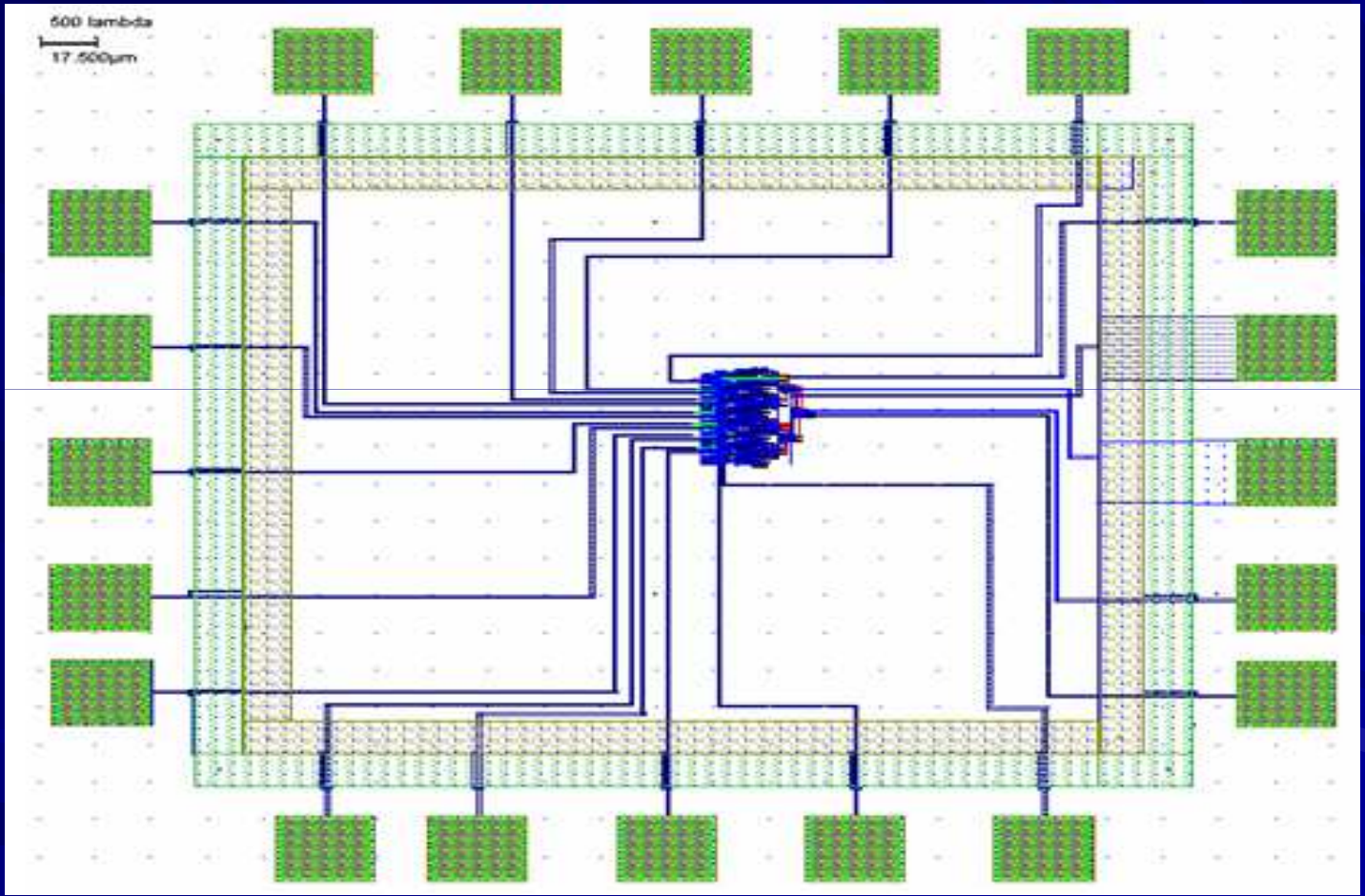


Figure : Dessin physique de l'additionneur self-checking en technologie 70nm

III - Implémentation des unités arithmétiques self-checking

Comparaison entre l'UA standard et l'UA self-checking

	Nombre de transistors pour l'unité arithmétique standard	Nombre de transistors pour l'unité arithmétique self-checking	Surcoût
Additionneur 2 bits	52	152	65.789 %
Additionneur 8 bits	208	656	68.292 %

Tableau : Comparaison du nombre de portes entre l'unité arithmétique standard et l'unité arithmétique self-checking

III - Implémentation des unités arithmétiques self-checking

	Surface pour l'unité arithmétique standard	Surface pour l'unité arithmétique self-checking	Surcoût
Additionneur 2 bits	S = 8.12 * 4.655 = 37.798 μm^2	S = 11.865 * 23.24 = 275.742 μm^2	86.292 %
Additionneur 8 bits	S = 32.48 * 4.655 = 151.194 μm^2	S = 46.09 * 31.955 = 1472.805 μm^2	89.734 %

Tableau : Comparaison de la surface entre l'unité arithmétique standard et l'unité arithmétique self-checking

Commentaire : Bien que l'auto-contrôle présente une grande importance dans la détection des fautes, nous pouvons remarquer d'après ce tableau qu'il cause une importante augmentation de surface.

III - Implémentation des unités arithmétiques self-checking

✓ Cependant, cette implémentation est plus avantageuse qu'une duplication duale du bloc fonctionnel. En effet, cette dernière implique 100 % de surcoût sans compter les contrôleurs.

✓ En plus, l'outil utilisé ne permet pas d'optimiser la conception comparé à des outils plus performants tel que CADENCE.

Plan de l'exposé

- **Introduction générale**
- **I - Le test des circuits intégrés numériques**
- **II - Les unités arithmétiques standards (non self-checking)**
- **III - Implémentation des unités arithmétiques self-checking**
- **VI – Simulation de pannes**
- **Conclusion et perspectives**

VI- Simulation de pannes

1- Panne extérieure

➢ Nous allons choisir comme exemple de panne le cas où a0 = a0*, c'est-à-dire que ces deux entrées ne sont pas complémentaires.

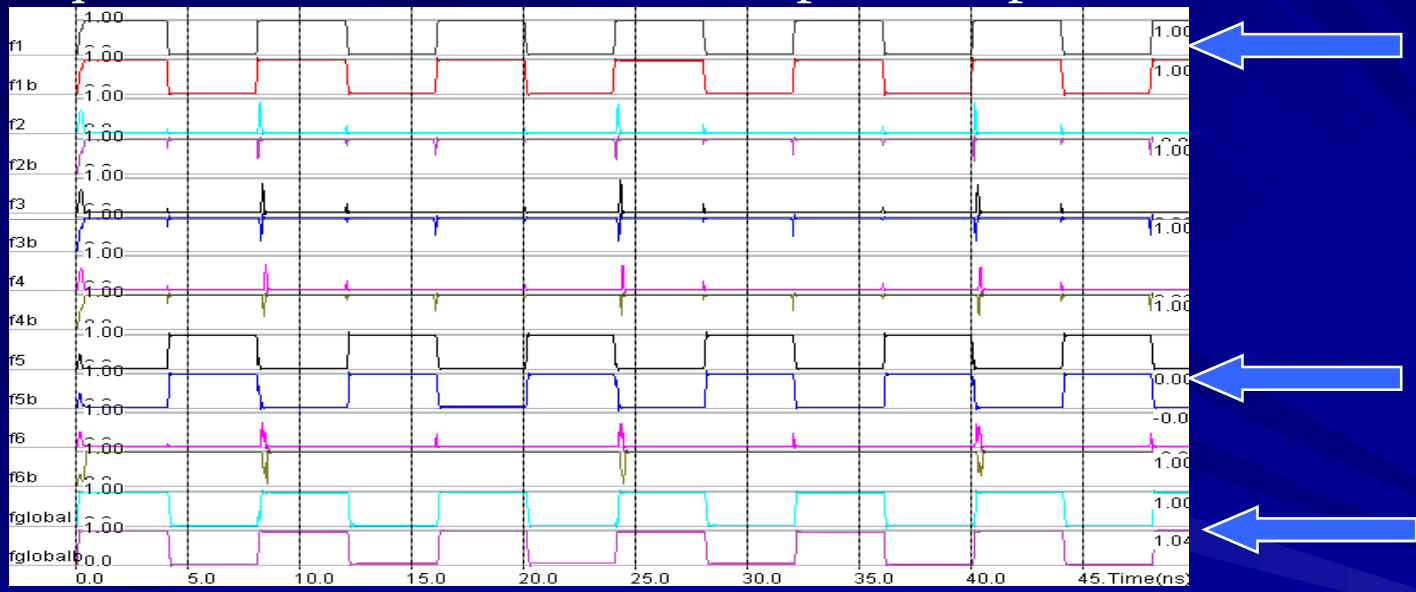


Figure : Une simulation SPICE avec provocation d'un défaut sur les entrées primaires

Commentaire : Dans ce cas de panne extérieure, la faute est détectée en premier lieu par le contrôleur 1 puis elle s'est propagée à travers le contrôleur 5 pour arriver au contrôleur de sortie (contrôleur global).

VI- Simulation de pannes

L'additionneur réalisé est donc auto-contrôlable puisqu'il est capable de détecter la présence d'une erreur au cours du fonctionnement. En effet, l'erreur s'est propagée vers la sortie pour être détectée : le contrôleur global signale un vecteur de sortie hors code puisque (fglobal, fglobalb) Є {01, 10}.

VI- Simulation de pannes

2- Panne de type Stuck-At « 0 »

Nous avons choisi un des transistors et nous avons provoqué volontairement une panne de type collage à zéro.

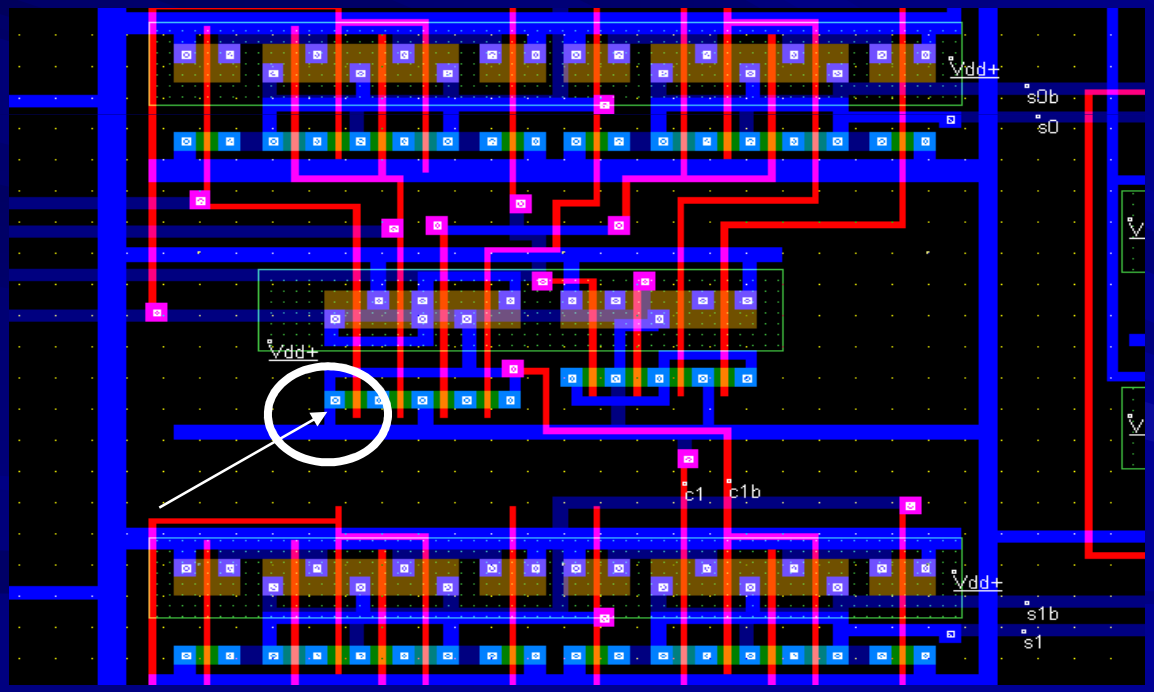


Figure : Circuit présentant une panne de type Stuck-At

VI- Simulation de pannes

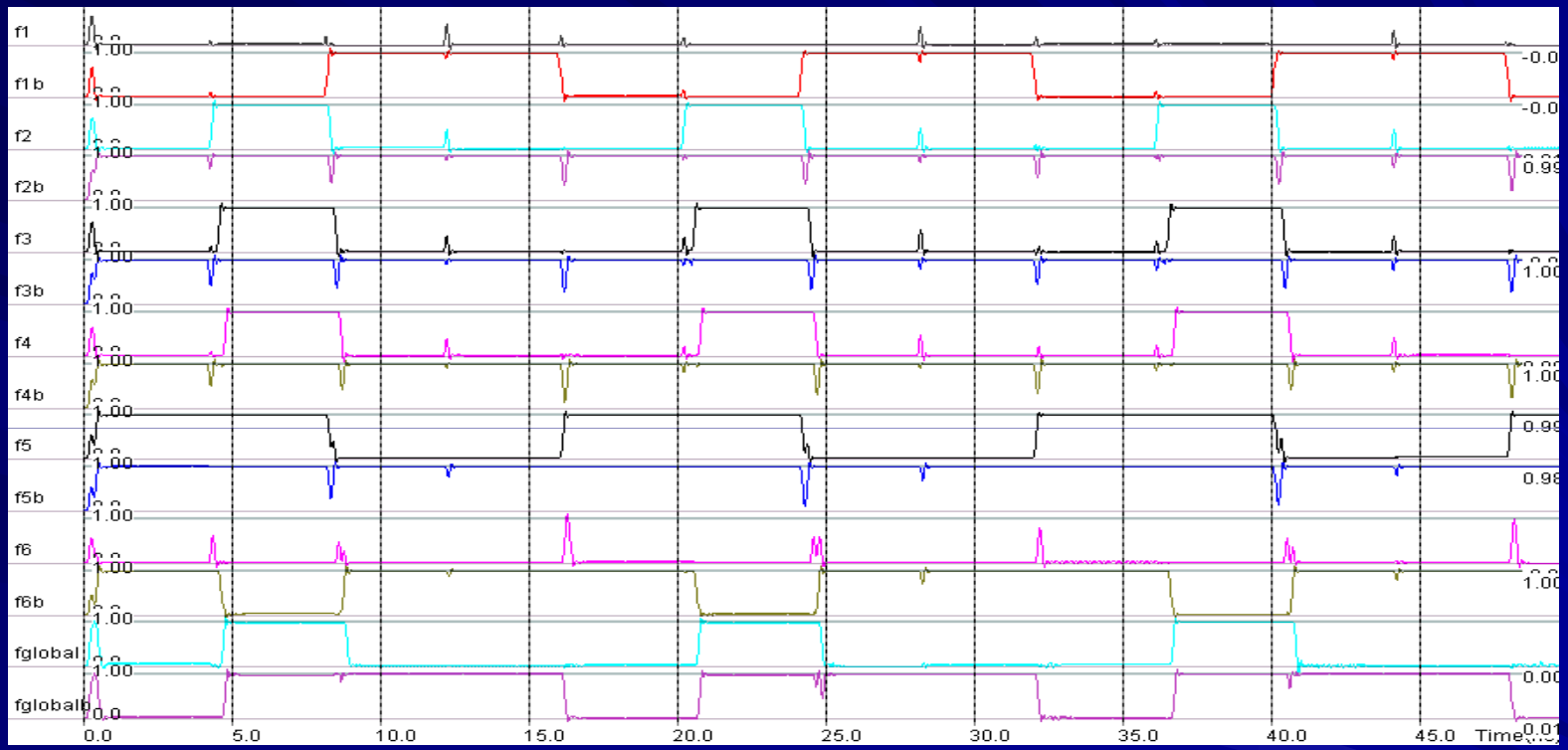


Figure : Une simulation SPICE avec provocation d'une panne de type collage à zéro

Commentaire : Nous remarquons un disfonctionnement des différents contrôleurs puisque fi et fib ne sont pas complémentaires donc la panne est détectée.

Plan de l'exposé

- **Introduction générale**
- **I - Le test des circuits intégrés numériques**
- **II - Les unités arithmétiques standards (non self-checking)**
- **III - Implémentation des unités arithmétiques self-checking**
- **VI – Simulation de pannes**
- **Conclusion et perspectives**

Conclusion et perspectives

✓ Dans ce travail, nous avons étudié et implémenté une unité arithmétique auto-contrôlable.

✓ Cette implémentation a été faite directement au niveau dessin de masques (full-custom).

✓ Le schéma a été ensuite validé par des simulations SPICE en utilisant l'outil CAO « Microwind 3.1 » qui permet de simuler directement le « Layout » du circuit.

✓ Pour démontrer la propriété d'autotestabilité de l'UA, des simulations SPICE ont été faites en présence de défauts, volontairement introduites dans le circuit et les résultats sont satisfaisants.

Conclusion et perspectives

✓ Nous avons montré que le code double-rail parait le mieux adapté vu le risque de propagation de l'erreur et la contamination entre les blocs dans ce type de circuit. Il est cependant à redondance maximale du fait qu'il nécessite la duplication duale du bloc fonctionnel et l'ajout de contrôleurs.

Conclusion et perspectives

✓ Dans notre travail, nous avons choisi comme application de l'unité arithmétique l'additionneur, mais il existe évidemment d'autres circuits auto-contrôlables tel que le multiplieur, le soustracteur, le diviseur…

✓ De même, ce projet s'est limité à une détection de la présence de fautes et nous pouvons penser comme perspective à la réalisation de circuits auto-contrôlables capables de localiser la panne et même de la corriger.

✓ On a des difficultés de conception des circuits self-checking en l'absence d'outils CAO adapté à ce type de circuits.

Merci pour votre attention