Sureté de Fonctionnement en Environnement Spatial

Première Partie

Luigi DILILLO

LIRMM (Laboratoire d'Informatique, Robotique et Microélectronique de Montpellier)
Université Montpellier

Sommaire

- Sureté de fonctionnement et tolérance aux fautes
  - Concepts de base
  - Entraves à la sureté de fonctionnement
  - Techniques de tolérance aux fautes
- Les structures tolérantes aux fautes
  - La redondance d'information
  - La redondance matérielle
  - La redondance temporelle
  - La redondance logicielle
  - Redondance mixtes

2

Sureté de fonctionnement et tolérance aux fautes

3

Concepts de base

La sureté de fonctionnement est caractérisée par des attributs, des entraves et des moyens

- Attributs
  - Disponibilité
  - Fiabilité
  - Sécurité-innocuité
  - Confidentialité
  - Intégrité
  - Maintenabilité
- Entraves
  - Fautes
  - Erreurs
  - Défaillances
- Moyens
  - Prévention des fautes
  - Tolérance aux fautes
  - Elimination des fautes
  - Prévision des fautes

Arbre de la sureté de fonctionnement

Selon la ou les applications auxquelles le système est destiné, l'accent peut être mis sur différentes facettes de la sureté de fonctionnement

4

Concepts de base

Les attributs:

- le fait d'être prêt à l'utilisation conduit à la disponibilité ;
- la continuité du service conduit à la fiabilité ;
- l'absence de conséquences catastrophiques pour l'environnement conduit à la sécurité-innocuité ;
- L'absence de divulgations non autorisées de l'information conduit à la confidentialité ;
- l'absence d'altérations inappropriées de l'information conduit à l'intégrité ;
- l'aptitude aux réparations et aux évolutions conduit à la « maintenabilité ».

5

Concepts de base

Un service correct est délivré par un système lorsqu'il accomplit sa fonction.

Les Entraves:

- Une défaillance du service est un événement qui survient lorsque le service délivré dévie du service correct.
- Le service délivré étant une séquence d'états externes, une défaillance du service signifie qu'au moins un état externe dévie du service correct. La déviation est une erreur.
- La cause adjugée ou supposée d'une erreur est une faute. Ces dernières peuvent être internes ou externes au système.

6

Concepts de base

Les Moyens: Le développement d'un système sûr de fonctionnement passe par l'utilisation combinée d'un ensemble de méthodes

- prévention des fautes : comment empêcher l'occurrence ou l'introduction de fautes ;
- tolérance aux fautes : comment fournir un service à même de remplir la fonction du système en dépit des fautes ;
- élimination des fautes : comment réduire la présence (nombre, sévérité) des fautes ;
- prévision des fautes : comment estimer la présence, le taux futur et les possibles conséquences des fautes ;

7

Entraves à la sureté de fonctionnement

Les fautes susceptibles d'affecter un système peuvent être classées selon sept points de vue, ce qui permet de définir les classes de fautes élémentaires :

1. *Phase de création ou d'occurrence* : Fautes de développement ou Fautes opérationnelles.
2. *Frontières du système* : Fautes internes ou Fautes externes.
3. *Cause phénoménologique* : Fautes naturelles ou **Fautes dues à l'homme.**
4. *Dimension* : Fautes matérielles ou Fautes logicielles

…

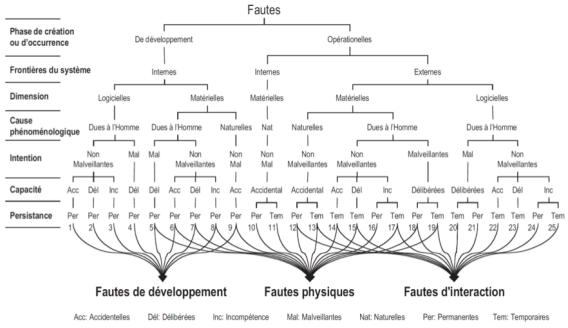
8

Entraves à la sureté de fonctionnement

5. *Intention* : Fautes malveillantes ou fautes non malveillantes.
6. *Capacité* : Fautes accidentelles, Fautes délibérées ou **Fautes d'incompétence.**
7. *Persistance* : Fautes permanentes ou fautes temporaires/transitoires.

Toutes les combinaisons des classes élémentaires étaient possibles, il y aurait 192 classes de fautes combinées.

Entraves à la sureté de fonctionnement



Classes de fautes combinées (réduites à 25 pertinentes)

Entraves à la sureté de fonctionnement

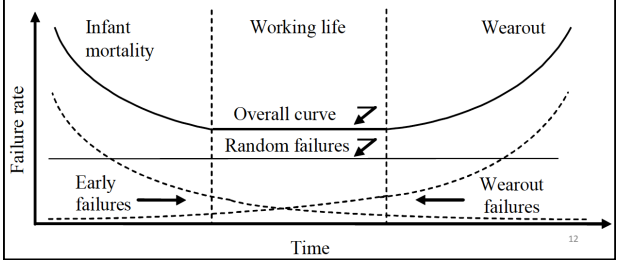
Trois grandes classes (non exclusives) :

1. Fautes de développement, qui rassemblent les fautes pouvant survenir pendant le développement.
2. Fautes physiques, qui rassemblent les fautes affectant le matériel.
3. Fautes d'interaction, qui rassemblent les fautes externes.

Failure Rate

λ(t) = Failure rate at device level

It is measured in FITS (Failures In Time for 10^9 hours)



Failure Rate at system level

A system is composed of multiple devices

Without the employ of fault tolerance techniques, whatever component fails the whole system may be considered failing.

$$\lambda_{sys} = \sum_{i=1}^{k} \lambda_{c,|i}$$

13

Reliability

If a component works properly at time t=0,
- R(t) is the probability that it will keep working at time t

When assuming **constant the failure** rate, the **exponential failure law** can be considered as a good approximation after **infant mortality**:

$$R(t) = e^{-\lambda t}$$

14

Computing exponential failure law

Considering N as the number of components of the system, we can define the **reliability R(t)** as:

$$R(t) = \frac{S(t)}{N}$$

With S(t) the number of component working at time t. The probability of failure is:

$$Q(t) = \frac{F(t)}{N}$$

With F(t) the number of faulty component at time t. Thus:

F(t) + S(t) = N and R(t) + Q(t) = 1

15

Computing exponential failure law

The failure rate is defined as the number of faults per unit of time and normalized at the number of remaining components:

$$Z(t) = \frac{1}{S(t)} \frac{dF(t)}{dt}$$

Supposing Z(t) as constant, and considering the previous equations, we obtain:

$$R(t) = \frac{S(t)}{N} = \frac{N - F(t)}{N} = 1 - \frac{F(t)}{N}$$

Deriving:

$$\frac{dR(t)}{dt} = -\frac{1}{N}\frac{dF(t)}{dt} \quad \Rightarrow \quad \frac{dF(t)}{dt} = -N\frac{dR(t)}{dt}$$

16

Computing exponential failure law

Let's change the elements Z(t) in the definition:

$$Z(t) = \lambda = \frac{1}{S(t)}\frac{dF(t)}{dt} = -\frac{N}{S(t)}\frac{d\hat{R}(t)}{dt} = -\frac{1}{R(t)}\frac{dR(t)}{dt}$$

Being R(t)= S(t)/N, we obtain:

$$\lambda \cdot dt = -\frac{dR(t)}{R(t)}$$

Integrating:

$$\int_0^t \lambda \cdot dt = -\int_1^{R(t)} \frac{dR(t)}{R(t)} \quad \Rightarrow \quad R(t) = e^{-\lambda t}$$

17

Reliability Block Diagram

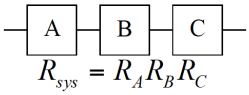
- The RBD can be used to represent the reliability of simple systems:
  - It is based on diagrams that show how the system reliability is related to the reliability of the single components
  - In RDB, the blocks are connected in **serial** or **parallel** configurations. Each block is a component of the system and it has it own reliability.
  - Parallel blocks are redundant, thus all parallel blocks must fail to induce system failure
  - A single failing block generate a system failure in a serial configuration

18

Reliability of a Serial System

- In a serial system all components must work (properly) to have the system work (properly)

$$R_{sys} = \prod_{i=1}^{N} R_i$$

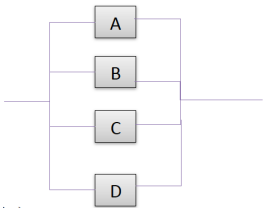


$$R_{sys} = R_A R_B R_C$$

19

Reliability of a Parallel System

- In a parallel system all components must fail to have the system failure



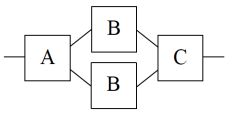
$$R_{sys} = 1 - \prod_{i=1}^{N} (1 - R_i)$$

$$R_{sys} = 1 - (1 - R_A)(1 - R_B)(1 - R_C)(1 - R_D)$$

20

Reliability of a System with redundancies

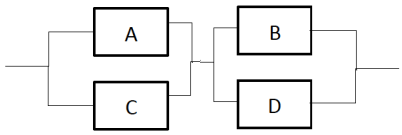
- System Reliability with the component B in parallel configuration can tolerate a failure in B



$$R_{sys} = R_A\left[1-(1-R_B)^2\right]R_C = R_A(2R_B - R_B^2)R_C$$

21

Example: comparison between parallel-serial and serial-parallel systems

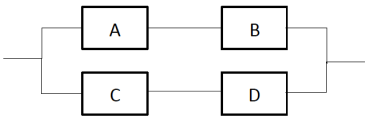


Parallel-serial system

$$R_{PS} = \left[1-(1-R_A)(1-R_C)\right]\cdot\left[1-(1-R_B)(1-R_D)\right]$$

22

Example: comparison between parallel-serial and serial-parallel systems



Serial-parallel system

$$R_{SP} = \left[1-(1-R_AR_B)(1-R_CR_D)\right]$$

23

Example: comparison between parallel-serial and serial-parallel systems

Considering the simplification

$R_A=R_B=R_C=R_D$

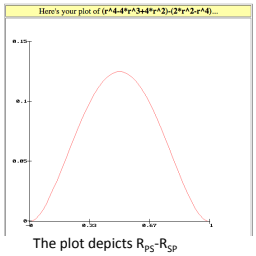
The previous equations become

$$R_{PS} = R^4 - 4R^3 + 4R^2$$

$$R_{SP} = 2R^2 - R^4$$

For values of R in [0,1]

$$R_{PS} \geq R_{SP}$$



The plot depicts R_{PS}-R_{SP}

24

Mean Time To Failure (MTTF)

- MTTF is the mean time before the system fails
- MMTF is the area below the reliability curve

$$MTTF = \int_0^\infty R(t)dt$$

- In case of an exponential failure

$$MTTF = \int_0^\infty e^{-\lambda t} dt = \frac{1}{\lambda}$$

25

Maintainability

- If a system is failing at t=0
  - M(t) is the probability that the system is repaired at time t
- The repair time is composed of
  - **Passive repair time**: time needed to reach the site
  - **Active repair time:** time necessary to isolate the failing component or replace it and check that the system works properly. It can be reduced with proper design that eases diagnosis and replacement/repair

26

Main Time To Repair (MTTR)

- μ is the frequency of repair of the system
- μ is similar to the failure rate λ
- The Maintainability is often modeled as

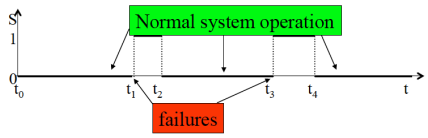
$$M(t) = 1 - e^{-\mu t}$$

- The Main Time Too Repair (MTTR) is given by

$$(MTTR) = 1/\mu$$

27

Availability



- The **Availability** of the system is the fraction of time in which the system is operational

$$system\ availability = \frac{MTTF}{MTTF + MTTR}$$

28

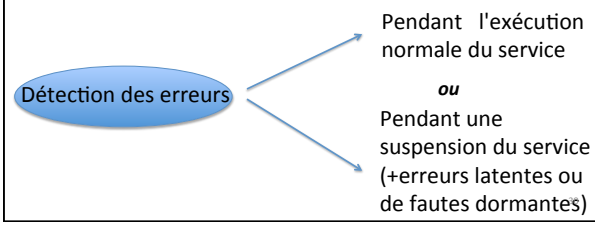
Safety: S(t)

- S(t) is the probability that in the frame of time [0,t] the system has not failure the way to induce inacceptable damages or catastrophic effects
- The Safety is a measure of how much the system is **fail-safe**

29

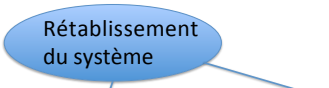
Techniques de tolérance aux fautes

La tolérance aux fautes est une des méthodes permettant le développement de systèmes sûr de fonctionnement. Elle vise à éviter les défaillances et est mise en œuvre par la **détection des erreurs** et le **rétablissement du système**.



30

Techniques de tolérance aux fautes



Manifestation

Traitement d'erreur (élimination des erreurs)

et

Cause

Traitement de fautes (prévention d'une nouvelle activation de fautes)

- **Reprise** : le système est ramené dans un état sauvegardé (point de reprise) survenu avant l'occurrence d'erreur
- **Compensation**: l'état erroné comporte suffisamment de redondance pour permettre de masquer l'erreur

- Diagnostic
- Passivation
- Reconfiguration ou réinitialisation.

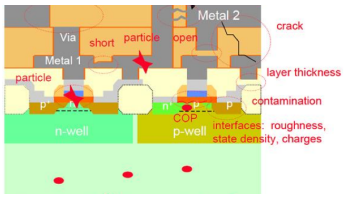
31

Classification of main failure mechanisms

32

Manufacturing defects

In IC manufacturing, different processes can be responsible for defects on fabricated products: implantation, etching, deposition, planarization, cleaning, lithography…



Different types of manufacturing defect

Contaminations and other mechanisms are defect causes in ICs:

a) Airborne Molecular Contamination (AMC);
b) Process induced defects, e.g. scratches, cracks, particles, overlay faults and stresses;
c) Process variations in doping profiles or layer thicknesses;
d) Deviation from design, due to pattern transfer from mask to wafer;
e) Diffusion of atoms through layers and in the semiconductor bulk material.

Interference phenomena

Ics continuously interact with the operating environment. They may be subject to:
- radiation strikes,
- electrical noises from crosstalk
- electromagnetic interferences with other running circuits in proximity.

Among these interactions, radiation strikes are the most important sources of error in CMOS ICs. They mostly create Soft Errors which affect memories, registers and combinational logics of digital systems.

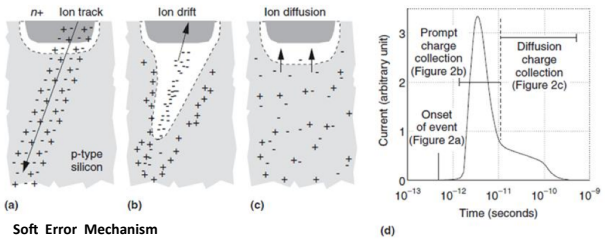
When a particle hits a silicon atom, it can induce :
- fission;
- shattering the atom into charged fragments that continue traveling through substrate. A cylindrical track of electron-hole pairs is formed.

When the ionization track comes close to the depletion region, the electric field rapidly collects carriers and creates a current/voltage glitch at that node.

34

Interference phenomena

After a particle strike a tunnel shape extending high field depletion region deeper into substrate is formed.



Soft Error Mechanism

Soft errors create voltage glitches at struck nodes. In combinational logic parts of digital systems, these glitches are called Single Event Transient (SET).

To completely flip state of a node, a minimum quantity of charge $Q_{critical}$ must be collected. This value depends on the capacitance and the voltage of the node.

35

Aging phenomena

Hot carriers injection, temperature variations, oxide wear-out and electromigration are responsible of aging of ICs, which may lead to defects in digital systems.

During ICs' lifetime, gate oxides are subject to stress and gradually wear out, with three main mechanisms:

1. Hot Carriers Injection (HCI): As transistors switch, high-energy (hot) carriers are occasionally injected into the gate oxide. These carriers are trapped in the oxide, and therefore change the current-voltage characteristics of the device.
2. Negative Bias Temperature Instability (NBTI): This problem concerns mostly PMOS transistors because they almost always operate at elevated temperature with strong negative bias (current gate voltage is at 0 while drain and source voltages are at Vdd). Dangling bonds called **TRAPS** develop at the Si-SiO2 interface and the threshold voltage Vth increases, reducing the drive current, and making transistors slower.
3. Time-Dependent Dielectric Breakout (TDDB): With an electric field applied across the gate oxide, the gate current gradually increases. After sufficient stresses, this can result in catastrophic dielectric breakdown that short-circuits the gate and cause system failures.

36

Fault classification in ICs

Manufacturing defects, variability, interference and aging phenomena may create different fault types in ICs. Despite different causes, these faults have some common impacts on circuits.

Based on **duration**, faults can be classified into three groups:

1. **Permanent faults**: They are irreversible. The most common sources of permanent faults are manufacturing defects. These faults can also be caused by aging phenomena at the end of lifetime when wear-out approaches.
2. **Transient faults**: These are faults that cause IC components to malfunction during a short period of time. Principle sources of transient faults are variability and interference phenomena.
3. **Intermittent faults**: They appear now and then, but they never disappear completely like transient faults, and they do not occur continuously like permanent fault either. However, intermittent faults often precede the occurrence of permanent faults. Aging phenomenon is the main cause of this reliability issue.

37

Fault classification in ICs

Faults can also be classified by their **resulting errors** in ICs. There are three types of error:

1. **Hard errors**: These are permanent errors which change permanently the logic function of ICs. A typical example is an error caused by a stuck-at-fault.
2. **Soft errors**: They are caused by transient faults. Depending on the place of error occurrence (in memories, combinational logics or registers), they may lead to bit-flipping (in memories and registers) or voltage glitches (in combinational logic). SET in combinational logic part and SEU in registers are the most common soft errors observed in digital circuits.
3. **Timing errors**: Components that suffer from timing error still provide correct logic outputs. However, they have higher delays between input and output signal establishments. Transient faults induced by PVT variability, manufacturing defects and aging phenomenon are responsible for this type of error.

38

Fault classification in ICs

	Permanent	Transient
Manufacturing defect	Hard error	Timing error
Variability		Timing error
Interference		Soft Error
Aging phenomenon	Hard error	Timing error

Types of fault and error, and the four main reasons for reliability issues.

39

Les structures tolérantes aux fautes

40

Généralité

Les structures numériques tolérantes aux fautes ont essentiellement été conçues pour tolérer des fautes apparaissant **pendant le fonctionnement des circuits**.

Le fin est d'empêcher que l'activation d'une faute sous la forme d'une erreur ne conduise à la défaillance du système par propagation.

Les structures tolérantes sont classe en fonction de leur type de redondance :

1. Redondance d'information
2. Redondance matérielle
3. Redondance temporelle
4. Redondance logicielle

41

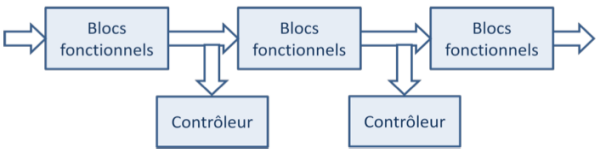
Les structures tolérantes aux fautes

Redondance d'information

42

La redondance d'information

La redondance d'information utilise des **codes détecteurs et correcteurs d'erreur**.



Architecture générale d'un système utilisant la redondance d'information

- Blocs fonctionnels : traitent les données en tenant compte naturellement des propriétés associées aux données ;
- Contrôleurs qui vérifient l'absence ou la présence de ces propriétés.

43

La redondance d'information

Les données sont manipulées et codées sur **n bits**

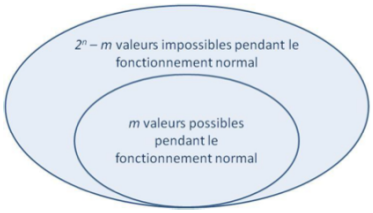
2^n possibilités de codage

m ($m < 2^n$) sont possibles en fonctionnement normal

(2^n - m) possibilités de mauvais fonctionnement

44

La redondance d'information



Valeurs possibles et impossibles d'un code

- L'ensemble des combinaisons possibles pendant le fonctionnement **correct** sont dites valides et forment un **dictionnaire.**
- Toute information qui présente des combinaisons **interdites** en fonctionnement normal sera qualifiée de **hors code.**

45

La redondance d'information

Le codage de l'information permet de **détecter** des erreurs mais il permet aussi de **corriger** des erreurs

Exemples:

- 1972, la sonde Mariner transmettait des images de Mars. Différents niveaux de gris étaient alors codés en mots de 32 bits. Pour cette transmission, les codes de **Reed et de Muller** étaient utilisés.
- 1979, la sonde Voyager transmettait des images en couleur de Jupiter. Transmission : 4096 niveaux de couleurs, 12 bits d'information, longueur des mots de 24 bits. Le code utilisé était le code de **Golay**. Il permet de corriger 3 erreurs et d'en détecter 7.

46

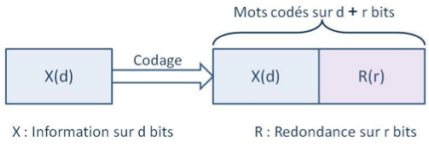
La redondance d'information

Exemples:

- CD audio numériques,. L'information est stockée sur un film d'aluminium et des trous microscopiques représentent des "0" et des "1". Un faisceau laser permet de lire l'information. Les lecteurs CD de Philips et Sony utilisent des codes de Reed-Solomon entrelacés.

47

La redondance d'information

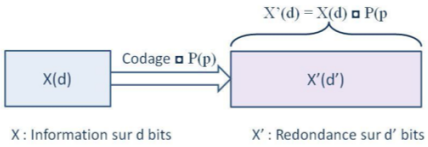


Représentation d'un code séparable

Le contrôle par codes détecteurs d'erreur est basé sur une redondance dans la représentation de l'information, soit en ajoutant des bits de contrôle aux mots

48

La redondance d'information



Représentation d'un code non séparable

Une opération mathématique ¤ est réalisée entre le mot à coder et un polynôme P(p).

L'extraction de l'information nécessite de faire appel à des procédures de décodage complexes.

49

Le code de parité

Implémentation consiste à ajouter un bit supplémentaire aux bits de données.
Le code de parité détecte toutes les erreurs simples ou plus généralement toutes les erreurs sur un nombre impair de bits.

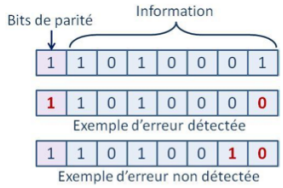
Le bit de vérification (ou de parité) p, examine le nombre de '1' et de '0' dans le mot de données :

- Si le nombre de '1' est impair, alors la valeur logique du bit de vérification de parité sera p = 1;
- Si le nombre de '1' est pair, alors la valeur logique du bit de vérification de parité sera p = 0

Exemple : le mot de donnée "10010100" deviendra, une fois codé, "100101001"

50

Le code de parité



Code de parité simple

Les bits de vérification, en général, sont formés par des opérations logiques OU EXCLUSIF entre les bits d'information

51

Le code de parité

Applications typiques sont la détection de fautes sur :

- les bus de données
- les registres
- dans les mémoires

Exemples :

- Dans les premiers PC les mémoires RAM utilisaient également ce mécanisme pour contrôler les données (1 bit de parité pour 8 bits de données). Une erreur était reliée à une interruption non masquable qui redémarrait la machine. (Remplacée par ECC et AECC)

- Le Minitel utilisait aussi le code de parité pour ses transmissions.

52

Le code de parité

Plusieurs bits de parité peuvent être utilisés et permettre ainsi non seulement la **détection** mais également la **localisation** de l'erreur à corriger.

r bits de parité -> $\mathbf{2^r}$ combinaisons différentes

Le nombre minimum de bits de parité pour corriger une erreur est le plus petit r qui vérifie l'inéquation suivante :

$$2^r \geq d+r+1$$

d représente le nombre de bits des données (information)

53

Le code de parité

Exemple :

d = 4 bits de données a3 a2 a1 a0

$2^r \geq d+r+1$ ⇒ r = 3 p2 p1 p0

Le mot complet sera donc :

a3 a2 a1 a0 p2 p1 p0

54

Le code de parité

Le mot complet est : a3 a2 a1 a0 p2 p1 p0
b6 b5 b4 b3 b2 b1 b0

Erreur	Bits de parité erronés	Syndrome
Pas d'erreur	Aucun	000
Bit 0 (p_0)	p_0	001
Bit 1 (p_1)	p_1	010
Bit 2 (p_2)	p_2	100
Bit 3 (a0)	p_0, p_1	011
Bit 4 (a1)	p_0, p_2	101
Bit 5 (a2)	p_1, p_2	110
Bit 6 (a3)	p_0, p_1, p_2	111

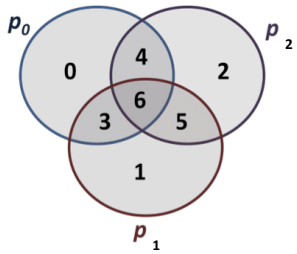
Interprétation des bits de parité

Ex. : si les 3 bits de poids faibles correspondant aux bits de parité valent "010", alors nous pouvons en déduire que c'est le bit 1 p1 qui est erroné

55

Le code de parité

Le mot complet est : a3 a2 a1 a0 p2 p1 p0
b6 b5 b4 b3 b2 b1 b0



Interprétation graphique des bits de parité

56

Le code de parité

Le mot complet est : a3 a2 a1 a0 p2 p1 p0
b6 b5 b4 b3 b2 b1 b0

p0 regroupe les bits de position 0, 3, 4 et 6 ce qui correspond aux bits p0, a0, a1 et a3 :

$p0 = a0 \oplus a1 \oplus a3$

de la même façon

$p1 = a0 \oplus a2 \oplus a3$

$p2 = a1 \oplus a2 \oplus a3$

Pour d = 1100 ⟹ p2 p1 p0 = 0 0 1

57

Le code de parité

D	R	A_D (r/ d)
2	3	1,5
4	3	0.75
8	4	0.50
16	5	0.31
…	…	…
256	9	0.035
512	10	0.020
1024	11	0.011

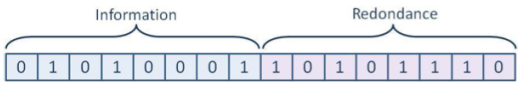
Coût en surface minimum pour détecter et corriger une erreur

Plus le mot est long et moins le coût en surface est important.
Pour un mot de 1024 bits, le coût en surface ne sera que de 1,1 % mais ce codage ne sera pas utile s'il y a plus d'une erreur simple.

58

Le code double rail

Il consiste à dupliquer et à inverser les bits de données.



Il permet de détecter des erreurs multiples

Redondance matérielle introduite similaire à duplication (élevée)

L'utilisation de cette technique est réduite à des applications critiques de petites surfaces.

59

Les codes k parmi N

Les codes k-parmi-n sont des codes **non séparables**.

Chaque mot a exactement :

k bits à la valeur '1'
Et
n − k bits à la valeur '0'

Si une erreur affecte le code, le mot résultant n'appartient plus au code non ordonné et ainsi l'erreur est détectable.

Exemple : le code 2 parmi 5
On peut coder 10 mots différents (chiffre d'un nombre décimal)

Chaque mot codé contient alors deux '1' et trois '0'

60

Les codes k parmi N

Les codes k-parmi-n sont des codes **non séparables**.

Exemple : le code 2 parmi 5

On peut coder 10 mots différents (chiffre d'un nombre décimal)

Chaque mot codé contient alors deux '1' et trois '0'

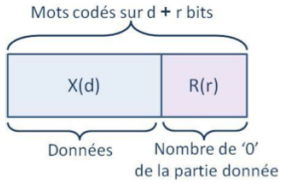
Si une erreur affecte le code, le mot résultant n'appartient plus au code non ordonné et ainsi l'erreur est détectable.

Chiffre	Mot codé
0	00011
1	00101
2	00110
3	01001
4	01010
5	01100
6	10001
7	10010
8	10100
9	11000

61

Les codes de Berger

Il est un code séparable où la partie de contrôle représente le nombre de '0' de la partie donnée.



Le nombre de bits de la partie contrôle, pour d bits de la partie donnée, est égal à

$\log_2(d + 1)$

Le coût en surface A_O (Area Overhead) du code de Berger est:

$A_O = [\log_2 (d + 1)] / d$

62

Les codes de Berger

Il est un code séparable où la partie de contrôle représente le nombre de '0' de la partie donnée.

Application: détecter les erreurs unidirectionnelles affectant surtout les

- PLA
- les mémoires ROM
- les circuits logiques sans inverseur avec de la logique partagée

Exemple : partie donnée = '00010',

la partie redondante de l'information est "100" (car il y a quatre '0' dans la donnée)

le mot codé sera "00010100"

63

Les codes cycliques

Le codage d'une donnée consiste à multiplier cette donnée par un **polynôme**.

Un code cyclique (k, n) est un code tel que toute permutation circulaire d'un mot du code est encore un mot du code.

Exemples :

1. Un code cyclique (1, 2) possède les mots de code suivants : {01, 10} ou {00, 11}, mais pas {01, 11}

2. Un code cyclique (1, 3) possède les mots de code suivants : {000, 111}.

64

Les codes cycliques

Codes cycliques performants sont

- codes BCH (Bose-Chaudhuri-Hocquenghem)
  - ils ont la plus grande capacité de correction d'erreurs indépendantes pour une redondance et une longueur donnée.
  - ils sont construit non sur un alphabet binaire mais sur un alphabet composé d'un grand ensemble de symboles

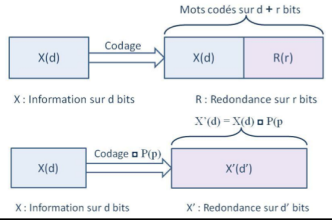
- RS (Reed-Solomon)
  cas particulier de codes BCH

65

Les codes arithmétiques

Les codes arithmétiques sont intéressants pour les **circuits arithmétiques** parce qu'ils sont préservés par les opérations arithmétiques comme l'addition, la soustraction ou la multiplication.

Il existe des codes arithmétiques **séparables** et des codes arithmétiques **non séparables**.



Séparables

Non séparables

66

Les codes arithmétiques

Dans les **codes non séparables**, pour une opération arithmétique ¤ entre deux opérandes A et B codés A' et B', cela signifie que :

$$A' \text{ ¤ } B' = (A \text{ ¤ } B)'$$

Exemple : le code consiste à multiplier par 5 chaque opérande

$A=3_{10}=00011_2$ et $B=6_{10}=00110_2$.

On en déduit

$A'=15_{10}=01111_2$ et $B'=30_{10}=11110_2$

Le résultat

$A' \times B' = 450_{10}=111000010_2$

Si une **erreur** s'est produite sur un des opérandes, ou pendant l'opération, ou sur la sortie (1 bit erroné par exemple), alors cette erreur peut-être détectée si le résultat n'est pas un multiple de 25.

67

Les codes arithmétiques

Dans les **codes séparables**, les mots codés sont obtenus en associant à la partie donnée, une partie de contrôle (code de **résidu**)

Définition: Le résidu d'un entier naturel est le nombre obtenu en additionnant tous ses chiffres, puis en additionnant les chiffres du résultat, et ainsi de suite jusqu'à l'obtention d'un nombre à un seul chiffre.

Exemple : le résidu du nombre 32452 est 7

car

3 + 2 + 4 + 5 + 2 = 16 puis 1 + 6 = 7

68

Les codes arithmétiques

Deux ***propriétés des résidus*** sont :

- Le résidu d'une addition de 2 entiers est le résidu de la somme des résidus de chaque terme
- Le résidu d'une multiplication est le résidu du produit des résidus de chaque facteur

Exemple : Prenons deux nombres 2048 et 3135

2048 + 3135 = 5183 2048 × 3135 = 6420480

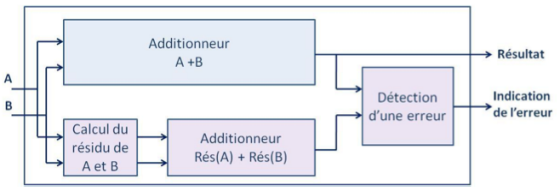
Le résidu de 2048 est 5 Le résidu de 3135 est 3

Pour l'addition : le résidu de 5183 est 8, ce qui correspond à la somme des résidus de 2048 et 3135 (5 + 3 = 8)

Pour la multiplication : le résidu de 6420480 est 6, ce qui correspond à la multiplication des résidus de 2048 et 3135 (5 × 3 = 15 ; 1 +5 = 6)

Les codes arithmétiques

Dans les **codes séparables**, les mots codés sont obtenus en associant à la partie donnée, une partie de contrôle (code de **résidu**)



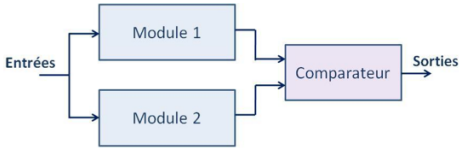
Architecture d'un code de résidu

Les structures tolérantes aux fautes

La redondance matérielle

Duplication

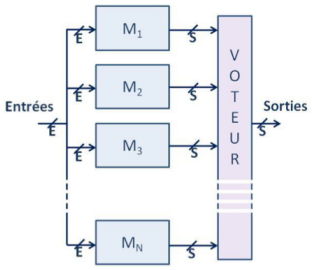
Un système dupliqué est la forme la plus simple d'architecture redondante.



Les deux modules sont identiques
Les entrées sont reliées -> les modules effectuent les mêmes opérations
Si un des deux modules a une défaillance le comparateur le détecte
Aucune possibilité de savoir lequel des deux modules est défaillant.
Pour pouvoir connaître cela il faut utiliser d'autres méthode de tolérance en plus de la duplication…

Systèmes NMR

Un système NMR (N-Modular Redundancy) est un système composé de N modules identiques qui fonctionnent en parallèle et qui traitent les mêmes données.



Les sorties sont des modules reliées à un voteur de majorité.

Le système fonctionne parfaitement si plus de la moitié des modules fonctionnent.

Pour que le système NMR tolère le dysfonctionnement d'au moins un de ses modules, il faut que N ≥ 3. Pour éviter les risques d'égalité, les systèmes NMR sont composés d'un **nombre impair** de modules

73

Systèmes NMR

Le NMR plus simple et la plus utilisée est le 3MR ou **TMR** ("Triple Modular Redundancy")

La TMR a trois modules identiques M1, M2 et M3
et
un élément de vote de majorité (Voteur)

S_{1i}, S_{2i}, S_{3i} les i-eme sorties de M1, M2, M3

SV_i la i-eme sortie du Voteur

$$SV_i = S_{1i}*S_{2i} + S_{1i}*S_{3i} + S_{2i}*S_{3i}$$

Si un seul module est fautif, alors les deux autres seront dominants et la faute sera masquée. Autrement, le résultat peut-être erroné.

74

Les structures tolérantes aux fautes

La redondance temporelle

75

La redondance temporelle

La technique de redondance du temps est basée sur la répétition des opérations.

Chaque opération est répétée plusieurs fois avant que le circuit ne délivre le résultat final.

Les sorties, à chaque fois, sont stockées pour la comparaison.

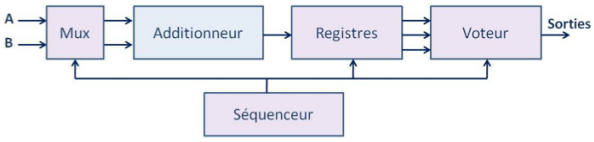
Exemple : Additionneur

1er op. S = A + B
2eme op. S = B + A
3ème op. S = A + B
4ème op. les trois résultats obtenus sont ensuite comparés et la sortie S est déterminée.

76

La redondance temporelle

La technique de redondance du temps est basée sur la répétition des opérations.



Le multiplexeur en entrée permet d'intervertir A et B.
Les registres mémorisent les résultats de l'additionneur.
Le voteur permett de sélectionner les résultats majoritaires stockés dans les registres
Le séquenceur gére les différentes étapes.

77

La redondance temporelle

La technique de redondance du temps est basée sur la répétition des opérations.

Avantage : l'implantation du matériel supplémentaire n'est pas très importante.

Défaut : puisque les opérations sont répétées, est que la fréquence de fonctionnement du circuit est beaucoup plus faible par rapport à une structure à redondance matérielle (au moins 1/3).

78

Les structures tolérantes aux fautes

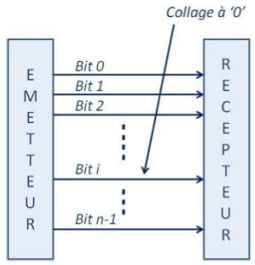
La redondance logicielle

79

La redondance logicielle

La redondance logicielle est une thématique à part entière.
N'est pas traite en détail.

Exemple : Bus composé de n bits



Bus de n bit avec une faute permanente de collage

80

La redondance logicielle

Exemple : Bus composé de n bits avec une faute permanente de collage

Le collage à '0' ne sera pas détecté si les bits i et i+1 du mot à transmettre sont égaux à '0'.

Si l'on considère que les 2^n mots différents que le bus peut transmettre sont équiprobables, alors 25% des fautes de collage ne seront pas détectées.

Pour réduire ce pourcentage, on peut retransmettre les données en effectuant une nouvelle opération de décalage.

81

La redondance logicielle

Exemple : Bus composé de n bits avec une faute permanente de collage

```
i = 0;
x = 3;
y = 1;
while (i < 5)
    {
    y = y × (x + i);
    i = i + 2;
    }
z = y;
```

a) Programme original

```
i = 0;
x = 6;
y = 2;
while (i < 10)
    {
    y = y × (x + i) / 2;
    i = i + 4;
    }
z = y;
```

b) Programme modifié

Deux programmes différents avec la même fonctionnalité

Si une erreur est présente, elle n'affectera pas les mêmes données ce qui entrainera deux résultats différents et permettra la détection de l'erreur.
La détection d'une telle faute (permanente) aurait été impossible si le même programme avait été exécuté deux fois.

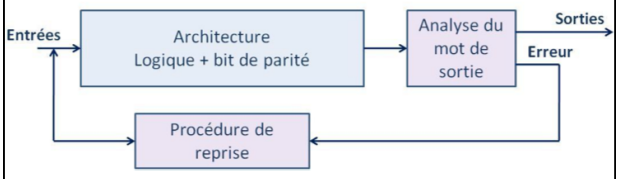
82

Les structures tolérantes aux fautes

Les redondances mixtes

83

Redondance d'information et redondance temporelle

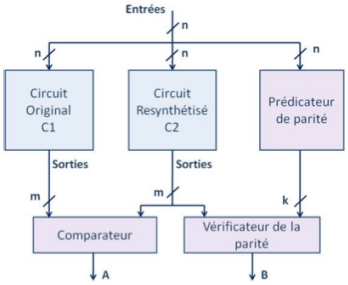


Cette architecture détecte une erreur sur une de ses sorties grâce à un code de parité.
Une procédure de reprise est alors effectuée et consiste à ré-exécuter la même opération mais à une **fréquence moins élevée**.
Correction des pannes temporelles (soft errors) ainsi que certaines pannes dynamiques (pannes n'apparaissant qu'en hautes fréquences).

84

Redondance d'information et redondance matérielle

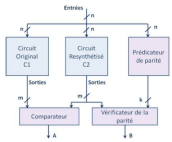
Il s'agit d'une architecture utilisant un code de parité et la duplication.



C2 fonctionnellement identique au premier mais le plus différent possible structurellement

Redondance d'information et redondance matérielle

Il s'agit d'une architecture utilisant un code de parité et la duplication.



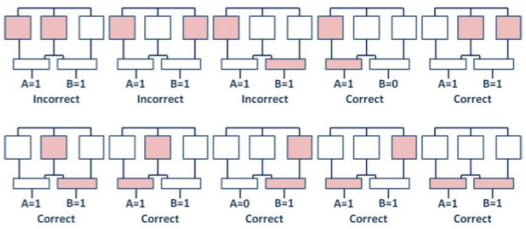
Inconvénient :la réalisation du bloc qui prédit la parité

La taille de ce bloc peut être supérieure à la taille des blocs logiques C1 et C2

Avantage : Même si 2 blocs sont défectueux, les sorties de l'architecture sont correctes dans 7 cas sur 10

Redondance d'information et redondance matérielle

Avantage : Même si 2 blocs sont défectueux, les sorties de l'architecture sont correctes dans 7 cas sur 10



Les structures tolérantes aux fautes

Bilan

Structures de tolérance aux fautes	Erreurs		Conditions / Particularités
	Détection	Correction	
Code de parité	×		Détecte un nombre impair d'erreurs
Code de parité multiple	×	×	Corrige un nombre d'erreur qui dépend de la longueur du code
Code double-rail	×		Détecte les erreurs multiples
Code k parmi n	×		Détecte les erreurs unidirectionnelles
Code de Berger	×		Détecte les erreurs unidirectionnelles
Code cyclique	×	×	Corrige un nombre d'erreur qui dépend de la longueur du code
Code arithmétique	×		Détecte les erreurs multiples
Duplication	×		Détecte les erreurs multiples
NMR	×	×	Corrige (N-1)/2 erreurs au minimum
N redondance temporelle	×	×	Ne corrige pas les fautes permanentes. Corrige seulement si la faute transitoire n'apparaît que pendant maximum (N-1)/2 opérations
Redondance mixte (information + temporelle)	×	×	Corrige les fautes permanentes dynamiques (n'apparaissant qu'en haute fréquence)
Redondance mixte (information + matérielle)	×	×	Corrige toutes les erreurs simples et certaines erreurs multiples

Récapitulatif des erreurs détectées et corrigées de différentes structures de tolérances aux fautes

89

TP/TD

90

Ex 1

Considering the table in slide #55 'le code de parité', change the syndrome coding and prove that the result is valid for a four bit data message 1001, for the existing coding and the new one.

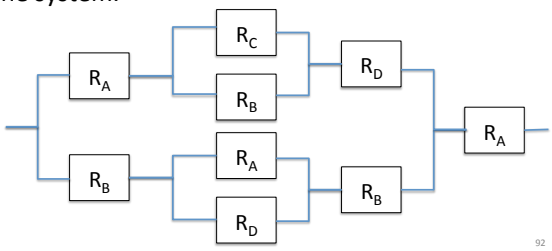
Ex 2

Considering the table in slide #55 'le code de parité', create a new table for a 8 bit message. Please check the function of detection / correction for the following message 10011100

91

Ex 3

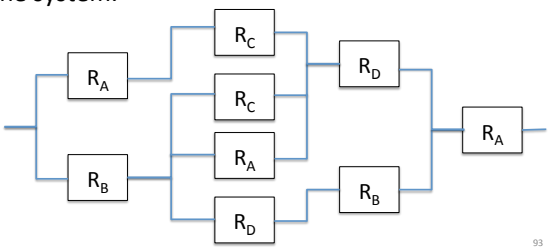
Calculate the expression of reliability of the system in the scheme below. Considering that $R_A=R_C=0.7$ $R_B=R_D=0.9$, calculate the reliability of the system.



92

Ex 4

Calculate the expression of reliability of the system in the scheme below. Considering that $R_A=R_C=0.7$ $R_B=R_C=0.9$, calculate the reliability of the system.



93

Ex 6

Considering the scheme in the exercise #3, find the value(s) of $R=R_A=R_B=R_C=R_D$ that induces the maximum reliability of the system.

Ex 7

Considering the scheme in the exercise #4, find the value(s) of $R=R_A=R_B=R_C=R_D$ that induces the maximum reliability of the system.

Ex 8

Considering the scheme in the exercise #5, find the values of $R_1=R_A=R_B$, $R_2=R_C=R_D$ that induce the maximum reliability of the system.

94