Réalisation d'un composant permettant détection et correction d'erreurs sur disques magnétiques haute capacité

Development of an error detection and correction component for high capacity magnetic disc

Michel FROIDEVAUX

Département Étude et Développement Matériel, Division Sous-Systèmes Mémoires de Masse, BULL, 94, avenue Gambetta, 75020 PARIS

Ingénieur études dans un service concernant des contrôleur de périphériques (disques magnétiques), l'auteur appartient à une équipe de développement matériel qui a pour mission de concevoir les schémas logiques et les programmes de tests associés. Dans ce cadre, il a été chargé de la conception d'un nouveau système de protection des enregistrements, de sa description logique et de l'algorithme associé. Actuellement il s'occupe des études sur un nouveau contrôleur qui est en phase de définition.

RÉSUMÉ

Nous avons réalisé un composant pouvant protéger des enregistrements de longueur variable, pouvant atteindre 50 koctets, et contenus sur des disques magnétiques. On utilise pour cela un code Reed-Solomon connu, qui est un code produit. C'est un code correcteur d'erreur simple qui corrige tout paquet d'erreurs dont la taille est inférieure ou égale à 17 bits. L'utilisation de la méthode du reste chinois permet un décodage rapide.

L'architecture, d'implantation originale, est contenue dans un composant de haut niveau d'intégration, et est incluse dans un boîtier standard de 40 broches.

MOTS CLÉS

Codes Reed-Solomon, VLSI, théorème du reste chinois, réalisation matériel.

SUMMARY

We developped an electronic component able to protect variable length magnetic disc records as long as 50 kbytes. This code can correct a single error burst of length 17 bits or less. The Chinese remainder method allows for a high speed decoding.

The architecture of this circuit, which is quite original, is implemented using large scale integration techniques, in a single standard 40 pin package.

KEY WORDS

Reed-Solomon codes, VLSI, *Chinese remainder Theorem, hardware realisation.*

TABLE DES MATIÈRES

Introduction

Dans les systèmes informatiques, l'apparition de disques magnétiques de haute capacité utilisant des codes d'enregistrement propagateurs d'erreurs a amené à repenser le système de protection des données.

A partir des tailles d'erreurs connues sur nos enregistrements, et, en tenant compte de l'accroissement de densité et de la propagation d'erreurs liés aux nouveaux disques, la taille moyenne de l'erreur (15 bits) sera supérieure à la capacité de correction du système actuellement utilisé (11 bits).

D'où la présente étude.

1. Description générale

1.1. Caractéristiques théoriques

Le chip ECC (Error Correction Code) est destiné à détecter et à corriger des erreurs sur des enregistrements de longueur variable pouvant atteindre la longueur d'une piste. Il utilise pour cela un code Reed-Solomon travaillant sur caractère 16 bits composé de trois polynômes dont les coefficients générateurs sont :

$$\alpha = (C081)\,H,$$
$$\beta = (2109)\,H,$$
$$\gamma = (0999)\,H.$$

Il crée, associé à tout champ de données, 6 caractères de contrôle (soit 12 octets) définis comme suit :

$$E1 = \sum_{i=1}^{N} x_i [\alpha^{-1}]^{N-i+1},$$

$$E2 = \sum_{i=1}^{N} x_i [\beta]^{N-i+1},$$

$$E3 = \sum_{i=1}^{N_1} x_{2i-1} [\gamma]^{N_1-i+1},$$

$$N1 = E\left(\frac{N+1}{2}\right),$$

$$E4 = \sum_{i=1}^{N_2} x_{2i} [\gamma]^{N_2-i+1},$$

$$N2 = E\left(\frac{N}{2}\right),$$

N pair :

$$E5 = \sum_{i=1}^{N_1} x_{i-1} \oplus E3 \oplus E1,$$

$$E6 = \sum_{i=1}^{N_2} x_{2i} \oplus E4 \oplus E2,$$

N impair :

$$E5 = \sum_{i=1}^{N_1} x_{2i-1} \oplus E2 \otimes E4,$$

$$E6 = \sum_{i=1}^{N_2} x_{2i} \oplus E1 \oplus E3,$$

$[\alpha^{-1}]$, $[\beta]$, $[\gamma]$ sont des matrices construites à partir des coefficients α, β, γ; x_i est le i-ième caractère du champ de données; N, leur nombre.

Devoir protéger des enregistrements de 50 koctets impose d'utiliser des polynômes générateurs de degré 16.

Si ces polynômes sont irréductibles et primitifs, leur période $2^{16}-1$, conduit à des temps de correction prohibitifs.

Ainsi les polynômes liés coefficients α et β :

$\alpha = (C081)_h$:

$$g(x) = x^{16} + x^{15} + x^8 + x + 1,$$

$\beta=(2109)_h$:

$$g(x)=x^{16}+x^{13}+x^8+x^3+1,$$

sont irréductibles et non primitifs de période 257 [1], le polynôme lié au coefficient :

$\gamma=(0999)_h$:

$$g(x)=x^{16}+x^8+x^7+x^4+x^3+1$$

est le produit de deux polynômes irréductibles et primitifs de degré 8 et de période 255 [1], soit :

$$g(x)=(x^8+x^6+x^5+x^3+1)\times(x^8+x^6+x^5+x^4+1),$$

les caractères E3 et E4 montrent que le polynôme lié à γ, travaille sur deux sous-champs distincts, les caractères pairs et les caractères impairs, ce qui porte la période « utile » liée à γ à $255\times 2=510$ caractères.

La période du code complet est le plus petit commun multiple des périodes « utile » de chaque polynôme. Soit : P = ppcm (257, 510) = 131 070 caractères 16 bits.

Le code utilisé peut donc protéger des enregistrements de 262 koctets.

Les caractères E5 et E6, d'où l'on déduit des clés de parité sur les caractères pairs et impairs, montrent que la capacité de correction est de deux caractères consécutifs soit, puisque 1 bit en erreur crée un caractère en erreur, une capacité de correction de 17-32 bits.

1.2. Organisation physique

Le chip contient 6 registres de 16 bits R1 à R6. Chacun contient 1 des 6 caractères de contrôle E1 à E6 qui sont réalisés en 4 blocs. Le premier bloc, contenant le registre R1 à 16 bits, et le second bloc, contenant le registre R2 à 16 bits, reçoivent tous les caractères du champ de données en les multipliant respectivement par $[\alpha^{-1}]$ et $[\beta]$.

Le troisième bloc, contenant les registres 16 bits R3 et R4, et le quatrième bloc, contenant les registres 16 bits R5 et R6, montrent une structure entrelacée (*voir* bloc diagramme).

L'entrelacement revient à diviser artificiellement le champ de données en deux sous-champs, ce qui permet de multiplier par 2 la portée de l'enregistrement protégé et le nombre de caractères en erreur que l'on peut corriger.

Dans ces structures entrelacées les caractères de rang impair, et les caractères de rang pair sont respectivement combinés entre eux. Le troisième bloc multiplie les caractères par $[\gamma]$. Le quatrième bloc réalise une clé de parité et contiendra l'erreur éventuelle en fin de lecture.

Le bus interne est à 16 bits. Les données en entrée et en sortie sont à 8 bits. Ainsi à l'entrée on mémorise 1 octet et on attend le suivant pour prendre en compte un caractère de 16 bits.

En sortie on envoie les caractères sur un multiplexeur dont la sélection est commandée par un compteur. Ce compteur commande aussi les sélections d'un multiplexeur 8×16 bits.

Ce multiplexeur a en entrée :

- R1 et R2;
- R3 et R4;
- R5 et R6;
- zéro, CPTR, 2B, STAT et BI (0 : 8), I (0 : 8). (On utilise ici une notation classique, I (0 : 8) est un groupe de 8 bits dont le premier est de poids 0.)

BI sont les I (0 : 8) qui sont mémorisés pour passer du bus externe 8 bits au bus interne 16 bits.

Zéro représente quatre zéros logiques.

CPTR représente la valeur du compteur 3 bits.

2B représente la valeur de la bascule créant les horloges caractères.

STAT représente des détecteurs de zéro sur chacun des six registres [Z1, Z2, Z4, Z3, Z6, Z5, 0L, 0L].

Les 16 sorties appelées DSL (00 : 16) (DSL = Data selected) sont envoyées sur les 4 blocs de registres et sur le multiplexeur qui donne les octets en sortie.

2. Description fonctionnelle

Le VLSI ECC réalise les fonctions suivantes commandées par les pins C(0 : 3) et A(4 : 4).

2.1. Écriture des données, écriture des octets de contrôle

On a $C(0:3)=0_H$, $A(4:4)=0$, pendant l'écriture des données. Les caractères sont envoyés sur les quatre blocs décrits au paragraphe 1. Chaque registre contient à tout instant le résultat du champ de données qu'il a reçu.

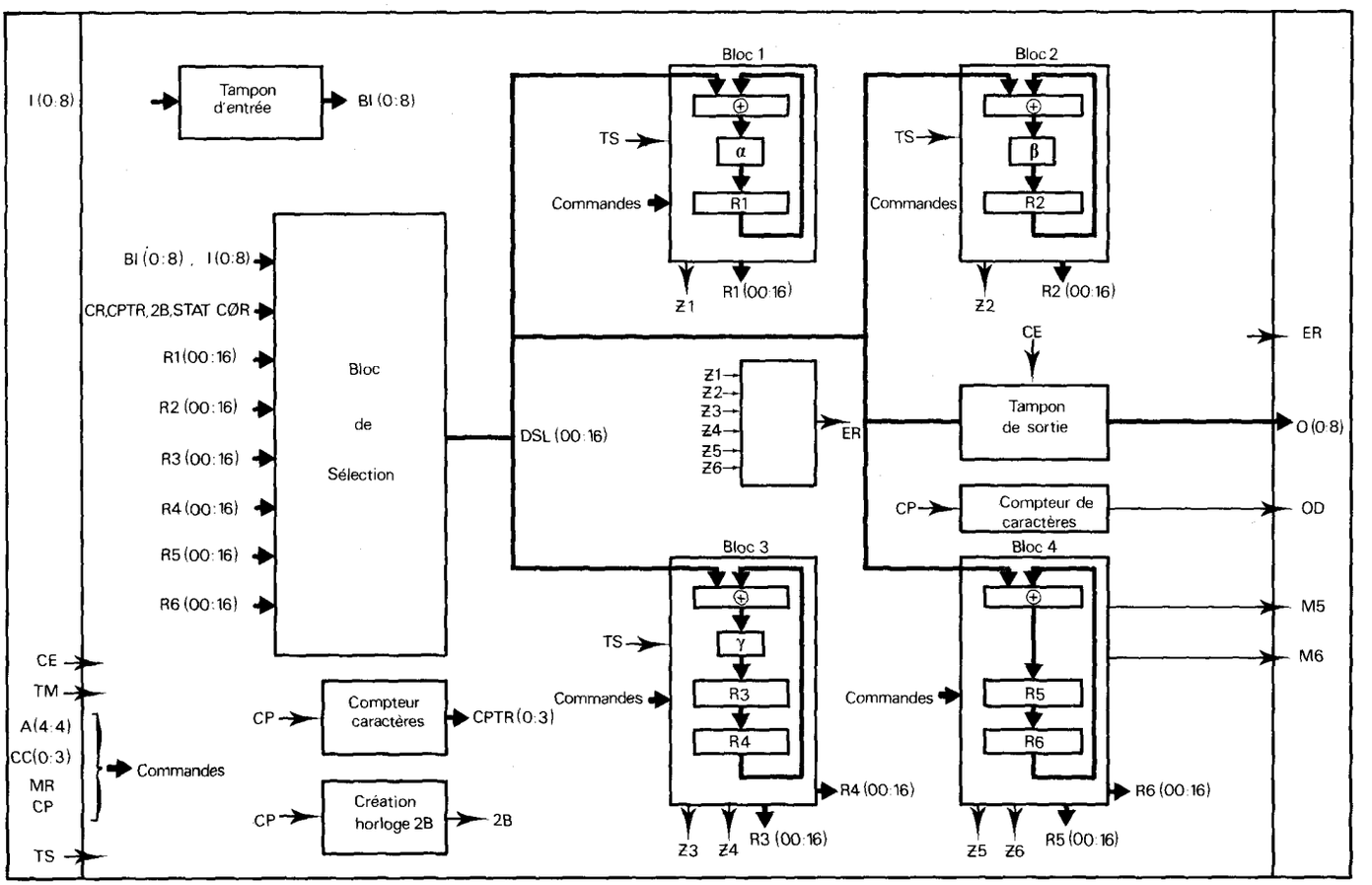
Ainsi à la fin de l'écriture du champ de données on peut immédiatement écrire les caractères de contrôle ($C(0:3)=4_H$, $A(4:4)=0$). Les caractères sortent octet par octet en ordre croissant de E1 à E6.

2.2. Lecture des données, lecture des octets de contrôle

Pendant la lecture des données ($C(0:3)=1_H$, $A(4:4)=0$), chaque bloc reçoit tous les caractères comme pendant l'écriture des données.

Pendant la lecture des octets de contrôle ($C(0:3)=5_H$, $A(4:4)=0$) chacun des trois premiers blocs ne reçoit que le caractère qu'il a généré lors de l'écriture : R1 reçoit le premier caractère de contrôle (appelé aussi E1), R2 reçoit E2, R3/R4 reçoit E3 et E4.

Le quatrième bloc R5/R6 reçoit les six caractères du champ E1 et E6.



Bloc diagramme

La lecture des octets de contrôle doit se terminer avec le signal ER = 0 : sinon il y a erreur dans le champ de données, d'où correction. Le signal ER est un détecteur de 0 sur l'ensemble des six registres.

2.3. Correction et visualisation/comparaison

La correction ($C(0:3) = 3_H$) débute si ER = 1 en fin de lecture.

Elle permet d'envoyer des coups d'horloge sur l'un quelconque des trois blocs, R1 ($A(4:4) = 8_H$), R2 ($A(4:4) = 4_H$) R3 et R4 ($A(4:4) = 2_H$).

Le quatrième bloc, R5 et R6, qui contient l'erreur, reste inchangé pendant toute la phase de correction.

2.4. Visualisation/comparaison

(*a*) La comparaison ($C(0:3) = 2_H$) permet de comparer le contenu de n'importe quel registre avec les registres R5 et R6 (*voir* commandes pour les valeurs appropriées de A (4 : 4)). Le résultat est donné par les signaux M5 et M6. Ainsi en alternant les fonctions de comparaison et de correction, on peut déterminer la position exacte de l'erreur dans le champ de données.

Ces fonctions seront reprises en détail au chapitre 3.

(*b*) La visualisation ($C(0:3) = 2_H$) permet de connaître le contenu de :

- l'un quelconque des octets de contrôle;
- des status;
- des données d'entrées : I (0 : 8);
- des données d'entrées qui ont été stockées (pour passer du bus externe 8 bits au bus interne 16 bits) : BI (0 : 8);
- du compteur interne (qui permet d'extraire les octets de contrôle pendant l'écriture, et de les distribuer pendant la lecture) et de la bascule « 2 B » (qui permet de passer de l'horloge externe octet aux horloges internes caractères).

On a ainsi la possibilité d'observer sur les bornes de sortie l'état de tous les flip flop internes et des détecteurs de zéro ce qui en association avec la borne TS facilite la testabilité du chip.

3. Organigramme de correction

Pour comprendre l'organigramme de correction, il est nécessaire de se rappeler la structure physique du chip.

On a quatre blocs indépendants contenant respectivement les E1, E2, E3 et E4, E5 et E6. Ces deux derniers blocs sont entrelacés et à tout instant les R3 et R5, les R4 et R6 ont reçu les mêmes caractères. On pourra donc directement comparer le contenu de ces registres.

Par contre, les deux premiers registres reçoivent tous les caractères, alors que les registres R5 et R6 reçoivent un caractère sur deux. Il sera donc nécessaire de calculer une référence pour le traitement des deux premiers registres.

L'organigramme se déroule en quatre phases successives :

1. A partir de l'analyse des status on différencie une erreur sur un caractère, sur deux caractères ou dans les ECC.

On distingue les status suivants :

ST = 28, erreur dans R5 (et pas dans R6).

ST = 14, erreur dans R6 (et pas dans R5).

ST = 00, erreur dans R5 et R6.

Ces trois cas renvoient sur trois routines de traitement distinctes.

ST = 78, B4, E4, D4, F8, F4, 30, C0, F0, A0, D0, 20.

L'erreur est dans les ECC ou est incorrigible ou à cheval data-ECC. On considère qu'une erreur est incorrigible si elle donne lieu, soit à un status non référencé, soit à un cas particulier de status (*voir* annexe).

2. *Traitement* E3 *et* E4

(*a*) *Erreur sur un caractère* ST = 14 *ou* ST = 28

On sait que l'erreur (ou les erreurs) n'affecte(nt) que le registre R5 (ou R6). On compare donc la valeur de ce registre à son équivalent R3 (ou R4).

Pour cela on sélectionne R3 (ou R4) (avec C0 : $3 = 2_H$ et A4 : $4 = 4_H$) et on teste le résultat de la comparaison en examinant M5 (ou M6).

Si M5 (ou M6) = 0 (inégalité) on incrémente le compteur d'horloge N3 (ou N4) et on se place sur le caractère suivant en donnant deux coups d'horloges sur les registres R3 et R4 (C0 : $3 = 3_H$ et A4 : $4 = 2_H$).

Si au bout de 255 comparaisons, c'est-à-dire au-delà de la période du polynôme généré à partir du coefficient γ, l'égalité n'est pas atteinte, c'est qu'il y avait plus d'une erreur dans le champ de données, on déclare l'erreur incorrigible.

(*b*) *Erreur sur deux caractères*

Le raisonnement est strictement identique, sauf que l'on mémorise le nombre de coups d'horloge N3 et N4 et que l'on interroge successivement M5 et M6.

A la fin du calcul on doit trouver :

- soit N3 = N4;
- soit N3 = N4 − 1.

Si cette condition n'est pas réalisée l'erreur est incorrigible.

(*c*) *Calcul de la référence de* E1 *et de la référence de* E2

Une fois traités E3 et E4 par rapport à E5 et E6, on traite E1 et E2. Il faut pour cela calculer les références réf. E1 et réf. E2. En effet E1 et E2 recouvrent tous les caractères du champ de données alors que E5 et E6 reçoivent un caractère sur deux.

Si l'erreur est sur un caractère, il n'y a pas de problème puisqu'alors E5 ou E6 porte seul l'erreur. On a donc :

ST = 28 : réf. E1 = R5, réf. E2 = R5.

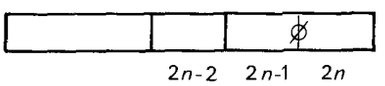
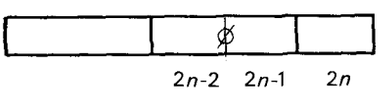
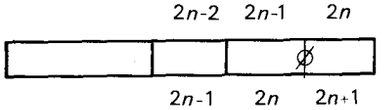
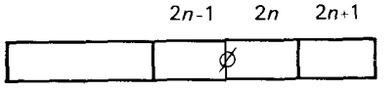
ST = 14 : réf. E1 = R6, réf. E2 = R6.

Si l'erreur est sur deux caractères, on doit alors tenir compte de la longueur du champ de données, pour établir la correspondance entre les registres R5 et R6 et la réalité E5 et E6.

Soit (R3), (R4), (R5), (R6), les valeurs figurant en fin de lecture dans les registres physiques R3, R4, R5, R6.

Soit E3, E4, E5, E6, les valeurs logiques définies par le code $\left(\mathrm{E3} = \sum_{i=1}^{N} x_{2i-1}[\gamma]^i \text{ par exemple}\right)$.

Envisageons les différents cas de figures. On a :

	ØD	Dernier caractère	Référence
	0	(R3) = E4 (R5) = E6 (R6) = E5	Réf. E1 = E5 [α^{-1}] ⊕ E6 = (R6) [α^{-1}] ⊕ (R5) EP = (R6), (R5)
	0	(R3) = E4 (R5) = E6 (R6) = E5	Réf. E1 = E6 [α^{-1}] ⊕ E5 = (R5) [α^{-1}] ⊕ (R6) EP = (R5), (R6)
	1	(R3) = E3 (R5) = E5 (R6) = E6	Réf. E1 = E6 [α^{-1}] ⊕ E5 = (R6) [α^{-1}] ⊕ (R5) EP = (R6), (R5)
	1	(R3) = E3 (R5) = E5 (R6) = E6	Réf. E1 = E5 [α^{-1}] ⊕ E6 = (R5) [α^{-1}] ⊕ (R6) EP = (R5), (R6)

Quand une erreur est du type $2n-1$, $2n$ (c'est-à-dire arrivant sur un caractère de rang impair, puis sur un caractère de rang pair) elle correspond à une réalité logique $E5[\alpha^{-1}] \oplus E6$, puisque E1 a reçu d'abord la partie de l'erreur contenue dans un caractère de rang impair, puis la partie de l'erreur contenue dans un caractère de rang pair. En fonction de la longueur du champ de données (parité ou imparité du nombre de caractères), on sait dans quel registre physique se trouve l'ECC logique. Ainsi, puisque les registres physiques R3 et R5 reçoivent le dernier caractère, on a R5 = E6 et R3 = E4 si le dernier caractère est de rang pair. Si le dernier caractère est de rang impair on a le symétrique, soit (R5) = E5, ...

Le calcul des références réf. E1 et réf. E2 doit être fait par logiciel.

3. *Traitement* E1 *et* E2

Le processus est identique à ceux décrits précédemment. La période des polynômes créant E1 et E2 est de 257, on s'interroge jusqu'à cette valeur en comparant réf. E1 et E1, et en donnant un coup d'horloge sur R1 si la comparaison n'est pas bonne.
Idem pour E2.

On doit avoir N1 = N2 (compteurs d'horloge sur E1 et E2).
On calcule D1.

4. *Calcul de la distance* D *ou se situe l'erreur à partir de* D1 *et* D34

On utilise la formule de Bezout soit :

$$\forall (u, v), \quad \exists (a, b)$$

tel que :

$$ua + vb = \text{pgcd}(a, b),$$

soit pour $a = 510$ et $b = 257$:

$$u\ 510 + v\ 257 = 1 = 64 \times 510 - 127 \times 257.$$

On sait que :

$$D = D_1 \bmod (257) = D_{34} \bmod (510),$$

soit :

$$64 \times 510 \times D = 64 \times 510\ D_1 \bmod (64 \times 510 \times 257),$$
$$64 \times 510 \times D = 64 \times 510\ D_1 \bmod (510 \times 257)$$

et

$$127 \times 257 \times D = 127 \times 257\ D_{34} \bmod (127 \times 257 \times 510),$$
$$127 \times 257 \times D = 127 \times 257\ D_{34} \bmod (510 \times 257),$$

soit :

$$D = (32640\ D1 - 32639\ D34) \bmod (131070).$$

D est calculée en caractère (2 octets), et est comptée à partir du dernier caractère de donnée.

4. Méthodes de construction des matrices directes et inverses

(*a*) *Construction d'une matrice directe*

On doit à partir du coefficient générateur créer une matrice.

Il faut tout d'abord connaître le polynôme générateur de cette matrice.

On sait que, pour une matrice directe (c'est-à-dire de la forme $[\beta]$ par opposition à $[\alpha^{-1}]$ matrice inverse) le polynôme s'obtient en ajoutant un 1 en poids fort du coefficient.

Exemple :

$$\gamma = (0999)_H = 0000\ 1001\ 1001\ 1001,$$

On retrouve la propriété des matrices directes à savoir que la matrice $[\alpha^{-1}]$ commence par l'élément α_{-1}, le coefficient α_{16} est particulièrement intéressant car il donne le polynôme générateur du corps.

On le calcule avec :

$$\alpha^{16}[\alpha^{-1}] = \alpha^{15} \quad \text{soit} \quad \alpha^{16} = (8103)_H,$$

d'où le polynôme générateur :

$$g(x) = x^{16} + x^{15} + x^8 + x + 1,$$

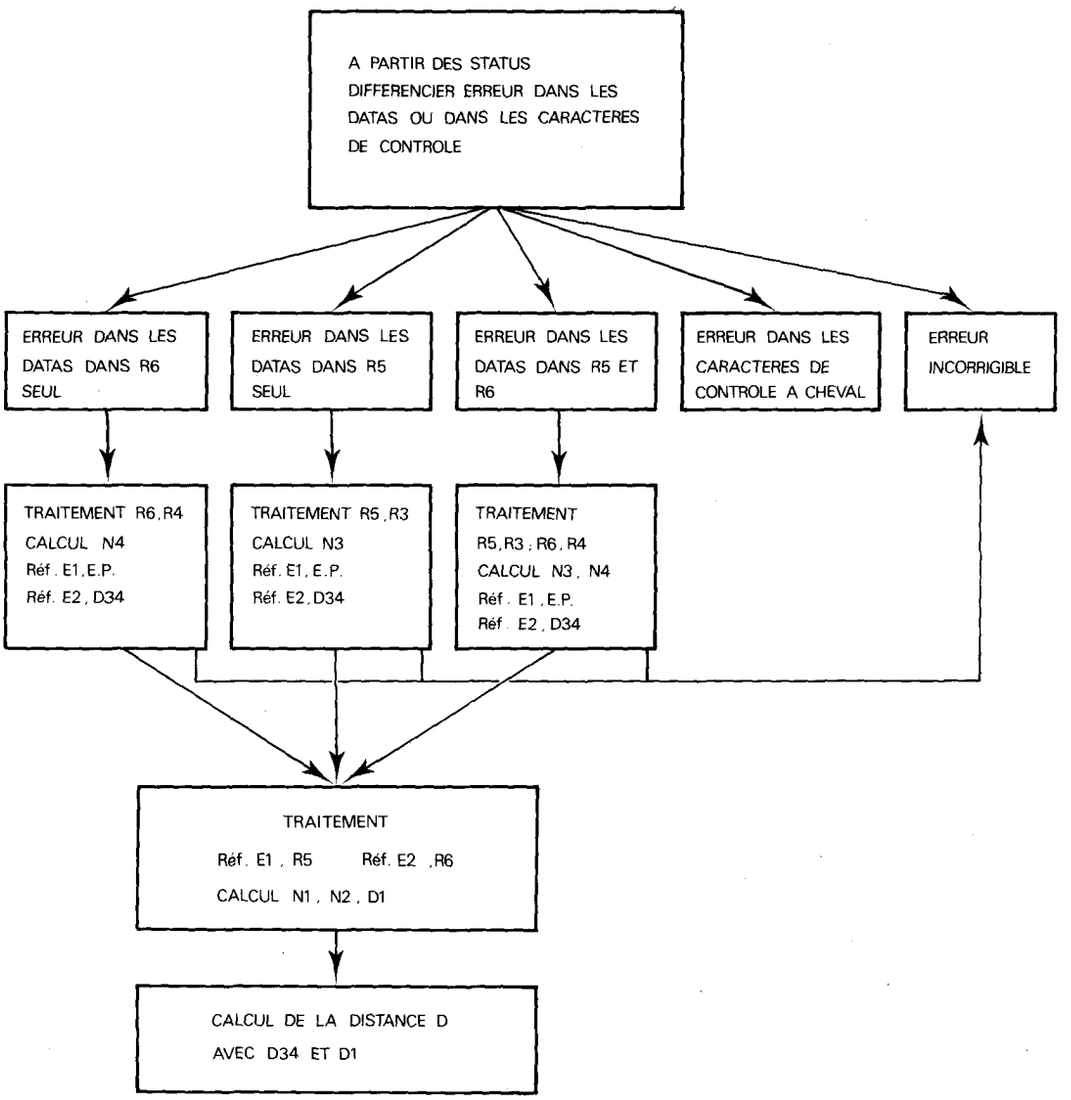
qui s'obtient à partir du coefficient α en ajoutant un 1 en poids faible.

Soit la formule :

$$\begin{cases} \alpha_{t+1}(i) = \alpha_t(i-1) \oplus [\alpha_{-1}(i) : \alpha_t(15)], \\ i \in [1, 15], \\ \alpha_{t+1}(0) = \alpha_t(15). \end{cases}$$

Exemples de patterns : Soit D_i les caractères du champ de données et E_i les caractères de contrôle.

D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	E1	E2	E3	E4	E5	E6
FFFF	0000	0000	0000	0000	0000	8000	0000	0000	0000	0800	2941	1332	1677	9B32	C0C9
FFFF	FFFE	0000	0000	0000	0000	0000	0000	0000	0000	7FAA	F5F6	0000	0020	7FAA	F5D7
0000	0000	0000	0000	0000	0000	0000	0000	0000	0001	1FCA	CEE7	1677	1675	F642	276C



Algorithme de correction ECC

a pour polynôme générateur :

$$g(x) = 1\ 0000\ 1001\ 1001\ 1001$$
$$= x^{16} + x^{11} + x^8 + x^7 + x^4 + x^3 + 1.$$

Associons aux puissances croissantes de x, les restes γ_i de la division de ces puissances par $g(x)$.
On aura :

Poids	15 14 . . . . . 1 0	
γ_0	1 0 0 . . . . . 0 0	
γ_1	0 1 0 . . . . . 0 0	$= [\gamma]$
γ_{15}	0 0 . . . . . 0 0	
γ_{16}	1 0 0 1 1 0 0 1 1 0 0 1 0 0 0 0	
	9 9 9 0	

On sait que les 16 éléments constitués par $\gamma_i \ldots \gamma_{16+i}$ représentent une matrice de la forme $[\gamma]^i$.

Pour construire la matrice $[\gamma]$ il suffit donc de prendre les éléments $\gamma_1 \ldots \gamma_{16}$.
Ce que l'on peut vérifier sur un exemple :

Poids	15 14 . . . . . 1 0
γ_{36}	0 0 0 1 1 1 0 1 1 1 0 1 1 0 1 0
γ_{37}	0 0 0 0 1 1 1 0 1 1 1 0 1 1 0 1

et

$$\gamma_{36} \times [\gamma] = \gamma_{37}.$$

On constate donc que la multiplication par une matrice directe correspond à un décalage vers les poids forts avec addition d'un masque (qui est le coefficient générateur) si le bit de poids fort est présent.

D'où la formule :

$$\gamma_{t+1}(i) = \gamma_t(i+1) + [\gamma_{16}(i) \cdot \gamma_t(0)],$$
$$i \in [0, 14],$$
$$\gamma_{t+1}(15) = \gamma_t(0),$$

$\gamma_t(i)$ représente le poids i de l'élément γ_t.

(*b*) *Construction d'une matrice inverse*

Nous avons construit la matrice $[\alpha^{-1}]$ à partir de l'examen de patterns.

Nous avons constaté :

– que le décalage se faisait vers les poids faibles;

– que le masque d'addition (le coefficient générateur) ne s'appliquait que si le bit de poids faible était présent.

Ceci conduit à une matrice comprenant les éléments $\alpha_0, ..., \alpha_{14}$ définis comme pour les matrices directes, auxquelles on ajoute un élément α_{-1} correspondant au coefficient générateur.

Soit :

Poids	15	14											2	1	0
		1				8				0				c	
α_{-1}	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1
α_0	1	0	0	.	.	.	.	.	.	.	.	.	.	.	0
α_1	0	1	0	.	.	.	.	.	.	.	.	.	.	.	0
α_{14}	0	0		.	.	.	.	.	.	.	.	.	.	1	0
α_{15}	0	0		.	.	.	.	.	.	.	.	.	.	0	1

$= [\alpha^{-1}]$

Ceci est possible grâce à une architecture originale permettant de réduire au maximum le volume logique.

L'intégration a permis de réduire :

– l'encombrement en transformant une plaque de $11 \times 9{,}5$ pouces contenant 120 chips de 16 broches en un chip de $2 \times 0{,}6$ pouces de 40 broches;

– la consommation électrique par un facteur 50 en passant de 10 W à 200 mW, soit 40 mA sous 5 V et par voie de conséquence les besoins de ventilation;

– les risques, la fiabilité d'une plaque étant globalement proportionnelle aux nombres de broches et à la température.

Le composant ECC est utilisé avec une routine de correction simple utilisant le principe du reste chinois ce qui permet de répondre en un nombre de coups d'horloge inférieur ou égal à 767. On peut ainsi protéger des enregistrements de grande longueur avec une routine de correction rapide, de l'ordre de 3 ms dans notre application et peu encombrante, environ 500 lignes de code utilisant un langage assembleur pouvant tester le bit.

Le composant ECC est utilisé dans un contrôleur de disques magnétiques et peut supporter un débit de 5 Moctets/s.

Dans notre application la logique interne au contrôleur et spécifique au composant est de 10 boîtiers TTL 16 broches, une ligne de code occupant 4 octets.

Conclusion

Le composant ECC réalise dans un boîtier standard de 40 broches, la détection d'erreur sur des enregistrements de 260 koctets et la correction d'erreur simple dont la longueur est inférieure ou égale à 17 bits.

Annexe

Cas des erreurs ECC et erreur à cheval

ST = 78	et	$\begin{cases} R1 = R5[\alpha^{-1}] \\ R1 \neq R5[\alpha^{-1}] \end{cases}$	→ Erreur dans R1 → Erreur incorrigible
St = B4	et	$\begin{cases} R2 = R6[\beta] \\ R2 \neq R6[\beta] \end{cases}$	→ Erreur dans R2 → Erreur incorrigible
ST = E8	et	$\begin{cases} R3 = R5[\gamma] \\ R3 \neq R5[\gamma] \end{cases}$	→ Erreur dans R3 → Erreur incorrigible
ST = D4	et	$\begin{cases} R4 = R6[\gamma] \\ R4 \neq R6[\gamma] \end{cases}$	→ Erreur dans R4 → Erreur incorrigible
ST = F8			→ Erreur dans R5
ST = F4			→ Erreur dans R6
ST = 30	$\begin{cases} \text{et} \\ \text{autres} \end{cases}$	$\begin{cases} R1 = R5[\alpha^{-1}] \\ R2 = R6[\beta] \end{cases}$	→ Erreur dans R1 et R2 → Erreur incorrigible

BIBLIOGRAPHIE

[1] PETERSON et WELSON, *Erreur correction codes*, second edition, p. 489-476, Massachussets Institute of Technology.

$ST = C0$ { et { $R3 = R5[\gamma]$ / $R4 = R6[\gamma]$ → Erreur dans R3 et R4
autres → Erreur incorrigible

$ST = F0$ → Erreur dans R5 et R6

$ST = A0$ { et { $R3 = R5[\gamma]$ / $R2 = R6[\beta]$ → Erreur dans R2 et R3
autres → Erreur incorrigible

$ST = D0$ { et $R4 = R6[\gamma]$ → Erreur dans R5 et R4
autres → Erreur incorrigible

$ST = 20$ et { $R3 = R5[\gamma]^2$ / $R1 = R5[\alpha^{-1}]^2 \oplus R6[\alpha^{-1}]$ / $R2 = R5[\beta]^2$ → Erreur à cheval data ECC
$EP = R5$
$D34 = D1 = D = 1$