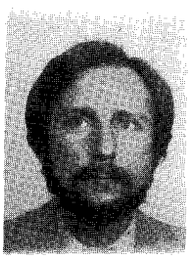
Codeur-décodeur intégré de codes en blocs

Integrated block codes encoder-decoder



Pierre LAURENT

THOMSON-CSF, Division Télécommunications, BP n° 156, 92231 GENNEVILLIERS CEDEX

Après une année passée à concevoir et mettre au point des simulations de trafic routier, l'auteur s'est orienté vers les systèmes de télécommunications protégées : aspects système, simulations, études de faisabilité dans un premier temps.

Il participe actuellement à la conception de systèmes de télécommunication numériques avancés dans les domaines civil et militaire, en s'occupant plus particulièrement des problèmes de codes détecteurs et correcteurs, de modulation, de codage de la parole et de traitement du signal en général.

RÉSUMÉ

La généralisation des transmissions de type numérique sur des canaux de qualité médiocre conduit à protéger l'information par adjonction de codes détecteurs et/ou correcteurs d'erreurs et d'effacements. Ces erratums peuvent être isolés ou au contraire regroupés en paquets, propriété que le code utilisé se doit d'exploiter.

Dans cette optique, on présente un codeur-décodeur de codes en blocs capable de traiter indifféremment des codes de type BCH ou Reed-Solomon de tailles variées. Pour les codes BCH, il admet de plus la répétition des bits et le vote majoritaire pondéré avant décodage.

Ce dispositif se présente sous la forme d'un circuit intégré, périphérique d'un microprocesseur, et pouvant fonctionner de manière autonome après paramétrisation. Son architecture est optimisée pour les opérations spécifiques du codage et du décodage.

MOTS CLÉS

Circuits intégrés monolitiques MOS, coprocesseur, codeur-décodeur, code détecteur et correcteur d'erreurs en bloc, décodage par transformée, code Reed-Solomon et BCH.

SUMMARY

Increasing needs in digital communications through poor quality channels lead to protect information by the use of errors erasures detection/correction codes. These errata may be randomly scattered or packed into short blocks: this fact can be exploited by the coding scheme itself.

The encoder/decoder that we describe here can process as well BCH *as Reed-Solomon codes; their sizes and error-correcting capabilities are user-defined parameters.*

Moreover, for BCH *codes, it is possible to repeat R times each of the transmitted bits and make a weighted majority decoding before* BCH *decoding.*

The device is to be a monolithic integrated circuit usable as an intelligent co-processor by a microprocessor. It can work independently after reception of the codes parameters. Its architecture is optimized for the repetitive and specific operations involved in the coding and decoding processes.

KEY WORDS

LSI MOS; *co-processor, encoder-decoder, error detecting and correcting block codes, transform-decoder, Reed-Solomon and* BCH *codes.*

TABLE DES MATIÈRES

Introduction

Il existe, particulièrement dans le domaine du radiotéléphone et dans celui des transmissions tactiques, un besoin croissant de protection de l'information contre les erreurs compte tenu de la faible qualité du canal de transmission.

Cette protection, pour des canaux classiques, se fait usuellement par adjonction à l'information de caractères/bits de contrôle, dits « caractères de parité », ou « CRC » qui permettent de minimiser l'effet des erreurs en les détectant dans la majorité des cas. Actuellement, l'augmentation du trafic dans des bandes de fréquence déjà pratiquement saturées, ainsi que le désir d'acheminer des données rapidement et avec sécurité, même en présence de brouillage, conduisent à améliorer ces procédés en leur ajoutant la possibilité de corriger directement tout ou partie des erreurs : ceci évite de demander systématiquement la répétition des messages où des erreurs ont été détectées et concourt ainsi à limiter le taux de charge du canal de transmission.

Les codes correcteurs le plus souvent utilisés sont des codes en blocs, choisis dans deux grandes familles :

– les codes construits sur un alphabet binaire, généralement dérivés des codes BCH, et principalement utilisés en transmission binaire avec des erreurs isolées : les symboles transmis sont des bits;

– les codes construits sur un alphabet à M éléments (M-aire), où M est une puissance de 2, de type Reed-Solomon, adaptés à la correction de paquets d'erreurs : les symboles sont des mots de m bits.

Suivant l'utilisation, les blocs codés (mots de code) peuvent avoir différentes longueurs, une capacité de correction plus ou moins élevée, et/ou être volontairement restreints dans leurs capacités de correction pour améliorer la sécurité de transmission : si le code est capable de corriger t erreurs, on se borne à n'en corriger en fait qu'un maximum de t_{eff} (t_{eff} inférieur ou égal à t), et on demande la répétition dans les autres cas.

Le lecteur intéressé par les problèmes de codes correcteurs pourra se reporter aux références [1], [2], et [3] pour plus de détails sur la théorie et les méthodes de codage et de décodage.

Lorsque le débit utile à transmettre ne dépasse pas quelques kilobits par seconde, les opérations de codage et de décodage sont le plus souvent faites par un microprocesseur ou une machine entièrement programmée. On trouvera dans [4] un exemple d'implémentation de décodage par transformée, qui est d'ailleurs celui que l'on propose ici.

Cependant, un microprocesseur n'a pas une structure adaptée à ce genre de traitements : il est plutôt optimisé pour des opérations arithmétiques et logiques simples et non pas pour les opérations arithmétiques et polynomiales sur des corps de Galois, qui font l'essentiel des traitements nécessaires. Qui plus est, il doit la plupart du temps se charger par ailleurs d'autres opérations, telles que la gestion du terminal, ce qui en rend la programmation parfois délicate.

A l'heure actuelle, la tendance est à l'augmentation du débit, nécessaire par exemple pour la transmission de phonie numérisée de bonne qualité, et ce, dans un environnement perturbé : on a donc besoin de codes performants dont l'implémentation sur microprocesseur devient dès lors problématique.

Un autre type de solution est alors envisagé, qui consiste à accoler au microprocesseur un ou plusieurs circuit(s) spécialisé(s) destiné à en accroître l'efficacité [5]; cependant, le microprocesseur est encore chargé d'un grand nombre d'opérations de configuration du circuit et d'échange de données avec lui.

Pour les débits très élevés, l'approche est différente et consiste soit à concevoir un codeur-décodeur entièrement câblé [6] soit à créer un ensemble de circuits (V) LSI spécialisés, éventuellement cascadables, directement interposés dans le flux de données [7, 8, 9].

Pour les débits que l'on pourrait qualifier de « moyens » ($\ll$1 Mbit/s), la voie d'avenir peut être la réalisation d'une machine unique standard, capable de traiter la plupart des codes usuels des familles citées précédemment, dans leurs différentes versions, sans autre intervention du microprocesseur hôte que la définition des codes utilisés, du mode de fonctionne-

ment, et éventuellement les opérations d'entrée-sortie de l'information à coder ou à décoder.

Cette machine pourrait être considérée comme un coprocesseur de traitement de codes, tout comme il existe des coprocesseurs arithmétiques ou de gestion de mémoire, par exemple.

On décrit ci-après une telle machine, qui se présente sous forme d'un circuit monolithique standard, périphérique intelligent d'un microprocesseur, grâce auquel on peut atteindre des débits de plusieurs dizaines de kilobits par seconde, avec des taux d'erreurs en ligne élevés.

1. Spécifications du codeur-décodeur

REMARQUE PRÉLIMINAIRE

Les spécifications données ci-après doivent être considérées comme les caractéristiques préliminaires du codeur-décodeur.

Elles pourront en effet être légèrement modifiées par la suite, notamment au niveau des modes d'exploitation. Elles donnent cependant un bon aperçu des possibilités du circuit, et ne seront pas fondamentalement remises en cause.

1.1. CODES TRAITÉS

Le dispositif peut traiter tous les codes BCH et Reed-Solomon (RS) de la forme (N, K) où N est le nombre total de symboles par mot de code et K le nombre de symboles d'information.

N peut prendre les valeurs maximales $N_{max} = 15$, 31, 63, ou 127.

Ces codes sont construits sur les corps de Galois CG(16), CG(32), CG(64), et CG(128) respectivement.

Les polynômes minimaux définissant l'élément générateur α des quatre corps de Galois concernés sont les suivants :

Pour $N_{max} = 15$:

$$g(X) = X^4 + X + 1 \quad \text{pour} \quad CG(16).$$

Pour $N_{max} = 31$:

$$g(X) = X^5 + X^2 + 1 \quad \text{pour} \quad CG(32).$$

pour $N_{max} = 63$:

$$g(X) = X^6 + X + 1 \quad \text{pour} \quad CG(64).$$

Pour $N_{max} = 127$:

$$g(X) = X^7 + X^3 + 1 \quad \text{pour} \quad CG(128).$$

TABLEAU 1
Caractéristiques des codes BCH et Reed-Solomon traités

Cardinalité du corps de Galois (M) et type de code	Nombre maximal d'erreurs pouvant être corrigées	Nombre de bits ou de Symboles de redondance nécessaires	Nombre de bits ou de Symboles d'information (min→max)
16 BCH	E=1→2 E=3 E=7	4.E 10 14	1→15−4.E 1→5 1(*)
16 RS	E=1→7	2.E	1→15−2.E
32 BCH	E=1→3 E=5 E=7 E=15	5.E 20 25 30	1→31−5.E 1→11 1→6 1(*)
32 RS	E=1→15	2.E	1→31−2.E
64 BCH	E=1→4 E=5→7 E=10 E=11 E=13 E=15	6.E 6.E−3 45 47 53 56	1→63−6.E 1→66−6.E 1→18 1→16 1→10 1→7
64 RS	E=1→31	2.E	1→63−2.E
128 BCH	E=1→7 E=9→11 E=13→15	7.E 7.E−7 7.E−14	1→127−7.E 1→134−7.E 1→141−7.E
128 RS	E=1→63	2.E	1→127−2.E

(*) Mots de code formés du même bit répété M−1 fois.

Ces polynômes ont été choisis en raison de leur faible nombre de termes non nuls, ce qui permet de minimiser la complexité des multiplieurs dans les corps de Galois.

Note : On envisage par la suite d'étendre le dispositif à des codes pour lesquels $N_{max} = 255$ [corps de Galois CG(256)], ce qui permettrait de transmettre directement des octets; seules des contraintes d'ordre technologique pourraient empêcher cette extension.

Les codes utilisés peuvent corriger un nombre t d'erreurs, compris entre 1 et un maximum qui dépend du corps de Galois et de la nature du code utilisé, sans toutefois pouvoir dépasser 15 dans la version actuelle du projet, ceci pour des raisons purement technologiques (taille des mémoires vives nécessaires).
Le tableau 1 donne la liste des codes qui peuvent être traités.

Si un code peut corriger t erreurs, sa distance de Hamming est égale à $2.t+1$; elle peut être portée à $2.t+2$, si l'utilisateur le désire, pour minimiser le taux de faux décodages (probabilité que la capacité de correction ait été dépassée, et que le décodeur ait été pris en défaut).

Ces codes sont utilisée en tant que détecteurs et/ou correcteurs d'erreurs et d'effacements : ceci est courant pour les codes RS, mais moins souvent utilisé pour les codes BCH.

En particulier, pour ces derniers, un mode de fonctionnement prévoit que chacun des bits d'un mot de code peut être répété R fois (R compris entre 1 et 127) et qu'un vote majoritaire pondéré est fait sur les R répliques d'un bit donné (sous-bits) avant le décodage proprement dit : une majorité définie au préalable est nécessaire pour que le bit en question soit considéré comme valide, sinon il est effacé. On notera qu'un sous-bit quelconque peut lui-même être effacé, tout comme un mot de m bits dans le cas des codes RS (symbole M-aire).

L'intérêt de ce vote majoritaire est de pouvoir utiliser des codes BCH sur des canaux de transmission présentant un fort taux d'erreurs décorrélées : par exemple un taux d'erreur en ligne de 20 % sur les sous-bits peut être facilement ramené à 1 % par bit en entrée du décodeur, ce qui est tout à fait compatible avec l'usage de tels codes qui ont une « efficacité » (nombre d'erreurs corrigées/redondance) moindre que celle des codes RS.

Dans un certain nombre d'applications, on a besoin d'un code pour transmettre des données, et d'un deuxième pour les signalisations.

Le dispositif proposé est donc apte à traiter deux codes différents :

– un premier code, dit « code de base », de paramètres N et K;

– un deuxième code, obtenu par raccourcissement du précédent en en positionnant implicitement un certain nombre de symboles de poids forts à 0, et, dans le cas des codes Reed-Solomon, en omettant éventuellement de transmettre un certain nombre de symbole de poids faible qui seront considérés comme effacés au décodage.

1.2. Type de codage

Le codage adopté est de type systématique : les symboles d'information sont inchangés, et placés en tête des mots de code, tandis que les symboles de redondance calculés sont mis à leur suite; les symboles de poids forts non utilisés sont implicitement positionnés à 0.

1.3. Types de décodage

La capacité effective de correction d'un code donné, notée t_{eff}, peut être modulée à volonté par l'utilisateur, qui précise au dispositif le nombre maximal d'erreurs qu'il admet de corriger. Ce nombre, t_{eff}, peut aller de 0 à t, t étant défini comme la capacité théorique du code (nombre maximal d'erreurs algébriquement corrigeables pour lequel il a été construit). Dans ces conditions, la correction n'est effectuée que si la relation suivante est vérifée :

2. nombre d'erreurs
$$+ \text{nombre d'effacements} \leqq 2.t_{eff}.$$

1.4. Modes de fonctionnement

Le processeur est conçu pour pouvoir fonctionner en chargeant le moins possible le microprocesseur hôte.

A la mise sous tension, ou quand on désire changer de codes, on lui donne les caractéristiques des deux codes utilisés [cardinalité (corps de Galois utilisé), nature : BCH ou RS, longueurs, capacités maximales]; il procéde alors à une série de calculs d'initialisation, après quoi il devient disponible pour effectuer des codages ou des décodages, sur ordre du microprocesseur.

Par exemple, si l'on lui demande une opération de décodage, il attend que l'extérieur lui ait fourni le nombre convenable de symboles, puis se met à effectuer le décodage, et enfin attend que l'on vienne lire le mot décodé. En l'absence d'ordres, il se remet ensuite automatiquement en attente de symboles à décoder.

Le microprocesseur peut à tout moment interrompre le dispositif de façon prioritaire.

Il peut aussi en connaître l'état, par le système classique de lecture transparente du mot d'état, ou bien lui demander de lui transmettre une interruption (IRQ) lorsque le traitement demandé est terminé.

1.5. Entrées-sorties

Les entrées et les sorties de données peuvent se faire en série (avec un modem par exemple) ou en parallèle (avec le microprocesseur).

Les deux sens sont indépendants.

Par exemple, pour une utilisation « transparente », entrées et sorties peuvent être toutes deux sous forme série, aux deux extrémités de la liaison.

Dans d'autres applications, où le microprocesseur doit entrelacer les symboles de différents blocs codés, l'entrée des symboles à coder peut être sous forme série ou parallèle, tandis que la sortie du mot codé se fait sous forme parallèle; au décodage, l'entrée des mots à décoder se fait sous forme parallèle, et la sortie après décodage sous forme série ou parallèle.

1.6. Débit en ligne

Le débit en ligne (débit net, en bits utiles par seconde) dépend du nombre de bits par symbole, du code utilisé, et de la vitesse de l'horloge de base du codeur-décodeur.

Il est limité par le temps de décodage dans le pire des cas, le codage étant toujours beaucoup plus rapide.

Par exemple, en supposant un temps de cycle de 200 ns (horloge à 5 MHz), le débit atteint plus de 64 kbit/s pour un code RS (31, 15) à symboles de 5 bits (décodage en 1,0 ms au plus), et plus de 128 kbit/s avec un code RS (127,97) à symboles de 7 bits (décodage en 5,2 ms au plus).

Note : Les temps indiqués tiennent compte des entrées-sorties.

2. Algorithmes de codage et de décodage

2.1. Généralités

Le système doit se présenter sous la forme d'un circuit intégré, et doit de plus être suffisamment rapide.

On a donc retenu comme algorithmes de codage et décodage ceux d'entre eux qui permettent de trouver des compromis favorables entre la complexité du circuit et la vitesse d'exécution. En particulier, on a préféré dans certains cas utiliser des procédés qui peuvent paraître peu élaborés du point de vue théorique ou qui nécessitent plus d'opérations que certains autres, mais qui, en revanche, peuvent être facilement implémentés sous forme câblée.

En outre, les algorithmes classiques ont été modifiés de telle sorte que les seules opérations arithmétiques à effectuer sur les éléments des divers corps de Galois se réduisent à l'addition et à la multiplication.

En effet, la division, l'évaluation d'exponentielles et de logarithmes dans un corps de Galois se font usuellement par consultation de tables, d'autant plus pénalisantes qu'ici il en faut une pour chacun des corps de Galois utilisables. Une table représentant de l'ordre de 1,5 kbit de mémoire morte pour l'ensemble des quatre corps de Galois utilisés, on a préféré alléger cette première version du circuit en supprimant la totalité des 4,5 kbits nécessaires aux trois tables.

A de minimes détails près, les mêmes algorithmes sont utilisés pour les codes BCH (où les symboles sont des bits), et pour les codes RS (symboles M-aires de m bits).

Enfin, on envisage, toujours dans le but de simplifier le circuit, d'implémenter le codage sous forme d'un décodage correcteur d'effacements de positions connues à l'avance : il nécessite alors un temps comparable à celui du décodage, ce qui peut être considéré comme parfaitement acceptable dans la plupart des applications.

2.2. Codage

Un mot de code est représentable sous la forme d'un polynôme C (Z) :

$$C(Z) = C_{N-1} Z^{N-1} + C_{N-2} Z^{N-2} + \cdots + C_1 Z + C_0.$$

Suivant la définition des codes BCH et RS, il doit être multiple d'un polynôme G(Z), dit générateur du code, dont les racines comportent au moins $2.t$ puissances successives d'un élément du corps de Galois concerné pour pouvoir corriger t erreurs.

Dans l'application, cet élément, noté α, est choisi primitif (chacun des éléments du corps de Galois est égal à une puissance entière de α, à l'exception du zéro). α est racine du polynôme $g(X)$ cité en 2.1.

On adopte la convention suivante, pour le codage :

$$G(\alpha^p) = 0, \qquad p = 1, 2, \ldots, 2.t$$

(codes de distance $2.t+1$)

et

$$G(\alpha^p) = 0, \qquad p = 0, 1, 2, \ldots, 2.t$$

(codes de distance $2.t+2$).

Le codage se fait de la façon suivante :

– recopie des K symboles d'information en tête du mot de code;

– calcul du premier symbole de redondance par sommation pondérée des K premiers symboles (information); si N, nombre de symboles par mot de code est inférieur à N_{max}, , les N_{max}-N symboles de poids fort (degrés élevés dans la représentation polynômiale) sont positionnés à 0;

– calcul de chacun des symboles de redondance suivants par sommation pondérée des symboles existants.

Les coefficients de pondération sont calculés dans la phase d'initialisation, et sont ceux du polynôme H(Z) défini comme suit, et qui ne dépend que du corps de Galois, de la nature du code, et de sa distance :

$$H(Z) = \frac{Z^{M-1} - 1}{G(Z)} = H_0 + H_1 Z + H_2 Z^2 + \ldots + H_{M-1-D} Z^{M-1-D}$$

[D : degré de G(Z)].

M est la cardinalité du corps de Galois, et G(Z) est le polynôme générateur du code, tel que défini précédemment.

2.3. Décodage

Le décodage s'effectue en cinq étapes, dont certaines peuvent être abrégées, voire supprimées suivant les cas.

Ces étapes sont les suivantes :

(*a*) repérage des effacements et calcul d'une première version du polynôme localisateur d'erratums, $L(Z)$ qui a pour racines les inverses des « positions » desdits erratums (un erratum est une erreur franche – non localisée –, ou un effacement – symbole à valeur douteuse ou non définie –); la « position » X_i d'un erratum affectant le coefficient de degré p dans la représentation polynômiale du mot reçu est égale à la puissance p-ième de α;

$$L(Z)=(1-X_1.Z).(1-X_2.Z)$$
$$\ldots\ldots.(1-X_e.Z)$$

(e : nombre d'effacements);

(*b*) calcul des syndrômes S_i, qui sont les valeurs prises par le polynôme $C(Z)$ après réception; ce polynôme est noté $C'(Z)$, l'on a :

$$C'(Z)=C(Z)+E(Z),$$

$$S_i=C'(\alpha^i), \qquad i=1, 2, \ldots, 2.t.$$

Le polynôme $E(Z)$ est appelé polynôme d'erratums; ses seuls coefficients non nuls ne peuvent être qu'à des positions correspondant aux erratums, et ont des valeurs Y_i égales à celles des erratums en question, lorsque la capacité du code n'est pas dépassée;

(*c*) prolongement de $L(Z)$, par l'algorithme de Berlekamp; $L(Z)$ se voit alors affecter des racines supplémentaires qui sont les inverses des positions d'erreurs, si la capacité du code n'est pas dépassée;

(*d*) calcul de la « transformée de Fourier » du polynôme d'erratums; cette transformée se calcule à partir des syndrômes et du localisateur d'erratums;

(*e*) calcul du polynôme $E(Z)$ par transformation de Fourier inverse; ses coefficients sont les valeurs des erratums;

(*f*) correction des erreurs, et reconstitution des effacements.

On trouvera en Annexe une présentation plus détaillée du procédé de décodage.

A chaque étape, des vérifications sont faites pour éviter les faux décodages qui peuvent se produire lorsque la capacité du code est dépassée, et qui se traduisent par l'adjonction d'erreurs supplémentaires à celles déjà présentes dans le mot code avant décodage.

Par exemple, pour les codes BCH, on vérifie dans la dernière étape que les valeurs des erratums sont bien égales à 0 ou 1, lorsqu'il y a des effacements : il se peut en effet que, dans ce cas, quand la capacité de correction du code est dépassée, la valeur calculée des erratums ne soit pas binaire.

De même, si la capacité de correction effective demandée est dépassée – même si la capacité maximale ne l'est pas – l'algorithme se termine par un diagnostic de non-décodage.

Enfin, lorsque la distance du code est égale à $2.t+2$, on se borne à vérifier que le sydrôme S_0 [valeur de $C'(Z)$ pour $Z=1$] à la valeur adéquate, sinon on déclare qu'il y a non décodage.

Remarques : D'autres méthodes de décodage existent, dont la plus connue consiste à évaluer séparément le polynôme localisateur d'erreurs seules [noté $\sigma(Z)$] en utilisant un polynôme supplémentaire (polynôme de Fornay) qui est obtenu à partir des syndrômes et du localisateur d'effacements.

La version finale de $L(Z)$ nécessaire à l'étape (*d*) est alors le produit de sa valeur initiale et de $\sigma(Z)$.

Le nombre total d'opérations à effectuer est peu différent de celui qui est nécessaire dans l'algorithme tel que décrit ci-dessus car, si l'évaluation de $\sigma(Z)$ est plus rapide que l'étape (*c*) en présence d'effacements, il est nécessaire d'effectuer une multiplication de polynômes supplémentaire.

D'autre part, la méthode de décodage proposée (décodage dit « par transformée ») est préférée à d'autres, plus classiques et moins consommantes en nombre d'opérations, car elle met en jeu des opérations simples et répétitives que le circuit réalise très rapidement grâce à un opérateur câblé spécialement optimisé.

3. Structure du codeur-décodeur intégré

3.1. Architecture générale

Compte tenu de la complexité et de la variété des traitements que doit réaliser le circuit, on a choisi une architecture de machine microprogrammée. Le programme, écrit une fois pour toutes, est stocké dans une ROM interne non accessible à l'utilisateur.

Un compromis a dû être trouvé entre les fonctions réalisées par logiciel et les fonctions câblées. Pour donner un exemple, le choix du corps de Galois utilisé n'a rigoureusement aucune répercussion sur le programme interne de la machine : un registre spécifique en stocke l'identité, pour reconfigurer automatiquement certains dispositifs, tels que par exemple un générateur de constantes ou un multiplieur.

Les principaux modules constitutifs du circuit sont les suivants (voir *fig.* 1) :

– un séquenceur, avec la mémoire morte associée (ROM) qui se charge de la gestion de l'ensemble du circuit : opérations d'entrée-sortie, de codage, et de décodage;

– une horloge, qui peut être pilotée de l'extérieur, et dont la vitesse est ajustée suivant l'opération en cours;

– un opérateur « Arithmétique », effectuant des opérations arithmétiques simples sur des nombres entiers signés, et des opérations logiques sur des octets;

– un opérateur « Galois », traitant uniquement des éléments du corps de Galois, spécialement optimisé

pour les opérations répétitives mises en œuvre lors d'un codage ou d'un décodage;

– un interface, assurant le dialogue avec l'extérieur, dont la mise en forme des données, et le stockage de certaines commandes ainsi que du mot d'état du circuit.

Les trois derniers modules sont interconnectés par un BUS interne de données qui peut véhiculer indifféremment entiers ou symboles M-aires.

3.2. Séquenceur et mémoire programme

Le séquenceur dispose d'un jeu réduit d'instructions, parmi lesquelles on trouve, outre l'exécution d'opérations en séquence :

– le branchement, l'appel de sous-programme, le retour de sous-programme, tous pouvant être soit conditionnels soit inconditionnels;
– l'exécution conditionnelle d'opérations;
– l'exécution répétitive d'opérations (permet entre autres d'effectuer en une seule instruction l'évaluation d'un polynôme);
– l'attente d'entrée-sortie;
– quatre interruptions câblées, dont les trois premières correspondent à l'arrivée d'une commande externe, et la quatrième à la fonction de test du circuit.

Le format des instructions est optimisé pour l'application, et la signification des différents champs d'une instruction donnée dépend de son type. Un même champ peut suivant les cas être une adresse mémoire, ou bien comporter le numéro d'une constante, ou encore être une partie d'une adresse de branchement.

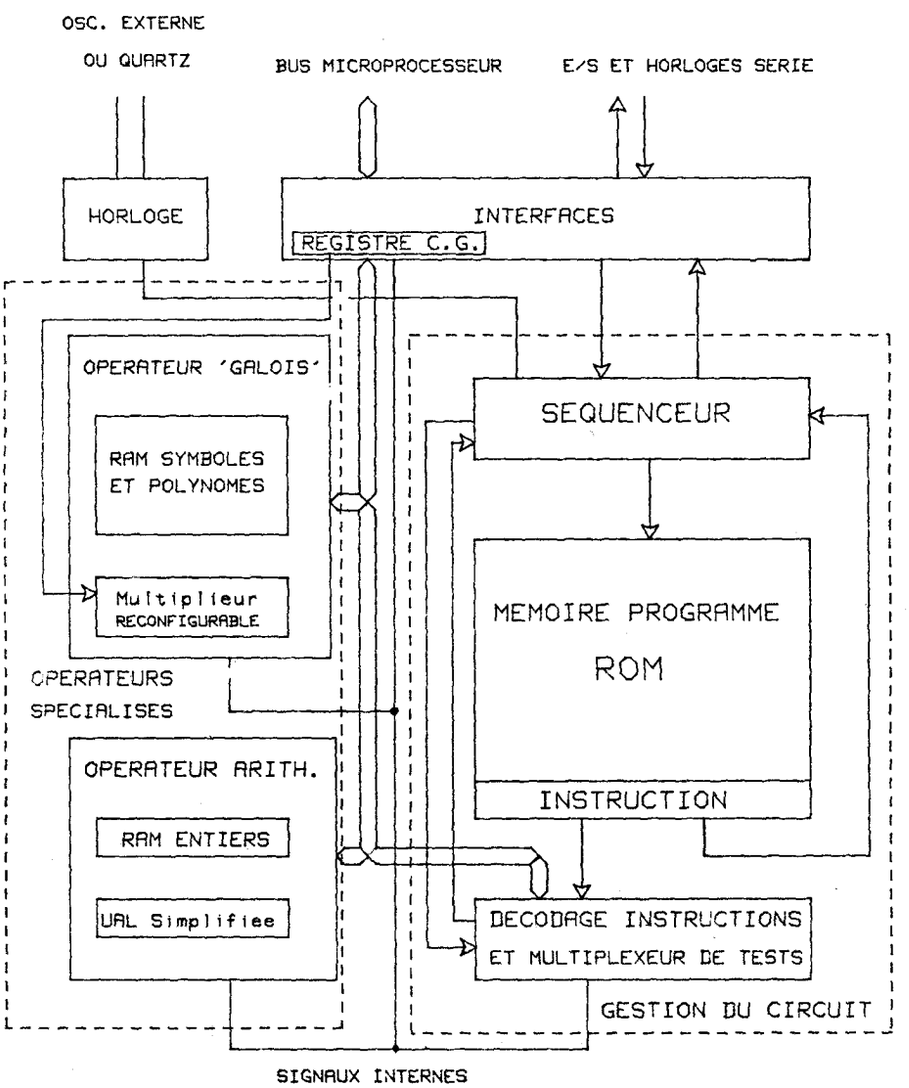


Fig. 1. – Architecture du codeur-décodeur intégré.

3.3. Opérateur arithmétique

L'opérateur arthmétique comporte le strict nécessaire pour effectuer les traitements sur nombres entiers signés (p. ex. des indices), ou des octets (p. ex. des mots de commande ou d'état).

Il contient :

– une mémoire vive à double accès, de faible capacité;

– une unité arithmétique et logique simplifiée, capable seulement des opérations suivantes, sur des mots de 8 bits :

- addition signée,
- soustraction signée;
- OU bit à bit (positionnement de bits à 1),
- ET bit à bit (positionnement de bits à 0, et test de bits);

– un registre accumulateur, pouvant servir entre autres à l'adressage de la mémoire;

– des commutateurs, autorisant la configuration des chemins de données et d'adresses suivant les besoins.

3.4. Opérateur galois

L'opérateur Galois a une structure globalement comparable à celle de l'opérateur arithmétique. Cependant, un effort particulier a été fait pour optimiser son architecture afin d'accélérer au maximum les traitements répétitifs.

Il se compose des éléments suivants :

– une mémoire vive sont certaines parties sont à double accès, découpée en pages, elles-mêmes découpées en sous-pages; la sous-page de rang le plus faible mémorise des scalaires, et les autres des polynômes;

– deux compteurs d'adresse, l'un fonctionnant en compteur ou décompteur, l'autre en décompteur seulement; il peuvent être chargés par une valeur provenant de l'opérateur arithmétique;

– un additionneur dans les corps de Galois;

– un multiplieur dans les corps de Galois qui est automatiquement reconfiguré suivant le CG utilisé;

– un registre accumulateur;

– des commutateurs, réalisant la configuration des chemins de données et d'adresses en fonction de l'opération scalaire ou polynômiale à effectuer.

Cet ensemble est le plus important de la machine, tant au point de vue de sa taille qu'au point de vue fonctionnel : c'est en effet à son niveau que se situent les opérations les plus complexes et les plus consommantes en temps de calcul.

3.5. Interface

L'interface est conçu pour que le fonctionnement des parties programmées du codeur-décodeur soit aussi indépendant que possible de l'environnement extérieur.

Il est réalisé sous forme de circuits de logique combinatoire complétés par un faible nombre de registres statiques.

Il se charge de la mise en forme et de la synchronisation des différents signaux, des aiguillages et masquages des diverses informations entrantes et sortantes, du stockage de certains des paramètres de fonctionnement, et de la mémorisation des mots de commande et du mot d'état. Ce dernier peut être lu par l'extérieur sans répercussion sur le fonctionnement du circuit.

Conclusion

Le codeur-décodeur de codes de blocs qui vient d'être présenté peut être considéré comme un composant standard, d'usage général dans les transmissions numériques, tant en raison de sa très grande flexibilité que parce qu'il se présente sous forme d'un composant unique assurant la « transparence » entre l'utilisateur (origine ou destinataire de l'information utile) et le canal de transmission, où une protection plus ou moins élevée contre les perturbations est nécessaire.

Son architecture détaillée a fait l'objet d'une simulation logico-temporelle; de même, son programme interne a été validé par vérification des valeurs des bits/symboles en sortie dans une grande variété de configurations.

Son intégration sous forme d'un circuit intégré monolithique n'a pas encore été réalisée, bien que l'on sache déjà qu'il pourrait se présenter sous la forme d'un boîtier standard à 24 broches.

Toutefois, sa complexité, sa surface, et sa consommation ont fait l'objet d'une première estimation, basée sur l'état de l'art actuel ou prévisible à court terme : le circuit comporterait moins de 50 000 transistors, pour une consommation ne dépassant pas quelques centaines de milliwatts avec une horloge à 5 MHz. Ceci est tout à fait comparables aux caractéristiques des circuits LSI actuels ou en développement.

Annexe

Décodage par transformée

Soit un mot de code, représenté par le polynôme $C(Z)$.

Après transmission, il est transformé en $C'(Z) = C(Z) + E(Z)$, où le polynôme $E(Z)$ est appelé polynôme d'erratum (son coefficient de degré n n'est pas nul s'il y a une erreur sur le symbole C_n).

S'il y a e erratums aux positions $p_1, p_2, \ldots, p_e$, et que la différence $C'_i - C_i$ entre la valeur erronée du symbole C_i et sa valeur vraie est notée Y_i, on peut écrire $E(Z)$ sous la forme :

$$E(Z) = \sum_{i=1}^{e} Y_i Z^{p_i}.$$

Par commodité, on appelle $X_n = \alpha \wedge p_n$ la « position » de la n-ième erreur.

En adoptant cette convention, si l'on calcule les valeurs de $C'(Z)$ pour $Z = \alpha, \alpha^2, \ldots, \alpha \wedge (2.t)$, appelées syndrômes, et sachant que par définition $C(Z)$ est nul pour ces mêmes valeurs de Z, on obtient :

$$S_k = C'(\alpha^k) = C(\alpha^k) + E(\alpha^k) = E(\alpha^k), \quad 1 \leqq k \leqq 2.t,$$

$$\boxed{S_k = \sum_{i=1}^{e} Y_i (\alpha^k)^{p_i} = \sum_{i=1}^{e} Y_i X_i^k, \quad 1 \leqq k \leqq 2.t.}$$

Soit alors ce que l'on appelle la transformée de Fourier du polynôme d'erreurs, définie par le polynôme $T(Z)$ tel que :

$$\boxed{T(Z) = \sum_{k=0}^{M-2} T_k Z^k,}$$

avec :

$$\boxed{T_k = E(\alpha^k) = \sum_{i=1}^{e} Y_i . X_i^k, \; 0 \leqq k \leqq M-2,}$$

Par définition des syndromes S_i, on identifie immédiatement les $2.t$ premiers coefficients $T_1, T_2, \ldots, T_{(2.t)}$ aux syndromes S_1 à $S_{(2.t)}$.

Soit alors le polynôme $L(Z)$, appelé « localisateur d'erratums » dont les racines sont les inverses des positions d'erratums (un erratum est soit un effacement de position connue, soit une erreur de position inconnue) :

$$\boxed{L(Z) = \prod_{i=1}^{e} (1 - X_i . Z) = L_0 + L_1 Z + \ldots + L_e Z^e.}$$

Si l'on effectue le produit des polynômes $T(Z)$ et $L(Z)$, le coefficient de degré n, est donné par :

$$W_n = \sum_{i=0}^{e} L_i T_{(n-i) \text{ modulo } M-1}.$$

Compte tenu des définitions de $T(Z)$ et de $L(Z)$, il vient :

$$W_n = \sum_{i=0}^{e} L_i . \left(\sum_{j=1}^{e} Y_j X_j^{n-i} \right)$$

$$= \sum_{j=1}^{e} Y_j . X_j^n \left(\sum_{i=0}^{e} L_i . X_j^{-i} \right),$$

$$W_n = \sum_{j=1}^{e} Y_j . X_j^n . L(X_j^{-1}) = 0.$$

Si le polynôme $L(Z)$ est connu (toutes les positions d'erratums sont connues), compte tenu d'une part du fait que son terme de degré 0 vaut 1, d'autre part que $T_1, T_2, \ldots, T_{(2.t)}$ sont connus, on peut calculer par récurrence les coefficients inconnus de $T(Z)$:

$$\boxed{T_n = - \sum_{i=1}^{e} L_i T_{(n-i) \text{ modulo } M-1}, \quad n = 2.t+1, \ldots, M-1.}$$

Alors, il suffit d'effectuer la transformation de Fourier inverse pour trouver les valeurs Y_i des erreurs aux positions p_i.

On a en effet :

$$\boxed{\begin{aligned} T(X_i^{-1}) &= \sum_{n=0}^{M-1} T_n X_i^{-n} \\ &= \sum_{n=0}^{M-1} \left(\sum_{j=0}^{e} Y_j X_j^n \right) . X_i^{-n} \\ &= \sum_{j=1}^{e} Y_i \left(\sum_{n=0}^{M-1} (X_j . X_i^{-1}) \right) \\ &= \sum_{j=1}^{e} Y_i \left(\sum_{n=0}^{M-1} \alpha^{p_j - p_i} \right) = Y_i. \end{aligned}}$$

La seule difficulté de l'algorithme réside dans l'évaluation du polynôme localisateur d'erratums, $L(Z)$.

En ce qui concerne les effacements, les positions X_i sont connues.

Par contre, les positions X_j des erreurs sont par définition inconnues : il faut les trouver pour que $L(Z)$ soit complet (sauf si le nombre d'effacements est tel qu'il n'y a pas de possibilité de corriger en plus des erreurs).

Or, il existe une relation liant $L(Z)$ et les syndromes.

Soit en effet le polynôme $S(Z)$ défini comme suit :

$$S(Z) = \sum_{i=0}^{2.t-1} S_{i+1} Z^i.$$

Si l'on effectue le produit de $L(Z)$ et de $S(Z)$, on montre facilement que les termes de degrés e à $(2.t-1)$ sont tous nuls, si $e < 2.t$. On peut en effet écrire, si W_n est un tel terme :

$$W_n = \sum_{i=0}^{e} L_i S_{n+1-i}$$

$$= \sum_{i=0}^{e} L_i \left(\sum_{j=1}^{e} Y_j X_j^{n+1-i} \right)$$

$$= \sum_{j=1}^{e} Y_j X_j^{n+1} L(X_j^{-1}) = 0.$$

Le polynôme $L(Z)$, qui est de degré e, est donc tel que tous les termes de degré e à $2.t-1$ dans son produit avec $S(Z)$ soient nuls.

Cette condition conduit à résoudre un système d'équations non linéaires, donc à solutions multiples,

ce qui n'est pas un problème trivial. Cependant, divers algorithmes existent, qui conduisent à la solution la plus vraisemblable, à savoir celle qui correspond au plus faible nombre d'erratums.

L'algorithme que l'on propose ici est l'algorithme de Berlekamp qui, au bout d'un nombre limité d'itérations, permet de compléter L(Z) lorsque la capacité de correction du code n'est pas dépassée.

On ne décrira pas ici cet algorithme que l'on trouve dans tous les ouvrages traitant des codes correcteurs en blocs.

Il suffit de savoir qu'il met en jeu des opérations simples qui sont les suivantes :

- évaluation du coefficient de degré n dans un produit de deux polynômes [S(Z) et version courante de L(Z)];
- multiplication d'un polynôme par une constante;
- multiplication d'un polynôme par Z;
- addition de deux polynômes;
- modification et test de scalaires.

De plus, cet algorithme peut être adapté pour que seules les opérations d'addition-soustraction et de multiplication soient nécessaires, ce qui permet de simplifier matériellement le décodeur.

En résumé, le décodage comporte les étapes suivantes :

(*a*) Comptage des effacements, et calcul simultané d'une version provisoire de L(Z); si le nombre d'effacements est trop élevé (supérieur à $2.t$), le décodage est impossible.

(*b*) Calcul des syndrômes; s'ils sont tous nuls, le décodage est terminé, puisqu'il n'y a aucune erreur détectable *a priori*.

(*c*) Sous réserve que le nombre d'effacements soit inférieur à $2.t$, mise en œuvre de l'algorithme de Berlekamp pour tenter de compléter L(Z); quand l'algorithme est terminé, et si L(Z) a été modifié, on recherche les racines de L(Z) qui ne sont pas des inverses de positions d'effacements; s'il y en a, et si diverses conditions complémentaires sont satisfaites, ces racines sont les inverses des positions d'erreurs; sinon, il y a non-décodage.

(*d*) Calcul de la transformée de Fourier du polynôme d'erratums, à partir de L(Z) et des syndrômes.

(*e*) Calcul des erratums par transformée de Fourier inverse, puis correction des erratums.

BIBLIOGRAPHIE

Théorie des codes et algorithmes généraux

[1] E. R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, 1968.

[2] W. W. Peterson et E. J. Weldon, *Error Correcting Codes*, MIT Press, 1972.

[3] G. C. Clark et J. B. Cain, *Error Correction Coding for Digital Communications*, Plenum Press, NY et London, 1982.

Décodage programmé, par transformée

[4] R. L. Miller, T. K. Truong et I. S. Reed, Efficient Program for Decoding the (255,223) RS Code over GF(2 ∧ 8) with both Errors and Erasures, using Transform Decoding, *IEE Proc.*, 127, Pt E, n° 4, juillet 1980.

Codeurs et décodeurs câblés rapides

[5] E. Lenormand, Réalisation d'un codeur-Décodeur (31,15) de Reed-Solomon, *Revue Technique THOMSON-CSF*, 12, 1980, p. 597.

[6] L. M. H. E. Driessen et H. T. Kanters, A High Speed Encoder-Decoder for a RS code With Minimum Distance at most seven, *International Zurich Seminar on Digital Communications*, 1984, p. G3.1-6.

Codeurs et décodeurs VLSI modulaires

[7] P. G. Farrel, Influence of LSI and VLSI Technology on the Design of Error-Correction Coding Systems, *IEE proc.*, 129, Pt, F, n° 5, octobre 1982.

[8] K. Y. Liu, Architecture for VLSI Design of RS Encoders, *IEEE Trans. on Computers*, C-31, n° 2, février 1982, p. 170-175.

[9] K. Y. Liu, Architecture for VLSI Design of RS Decoders, *IEEE Trans. on Computers*, C-33, n° 2, février 1984, p. 178-189.