Titre: Title:	Conception d'un processeur à données ancillaires intégré
Auteur: Author:	Yann Deslauriers
Date:	1999
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Deslauriers, Y. (1999). Conception d'un processeur à données ancillaires intégré [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8583/

UNIVERSITÉ DE MONTRÉAL

CONCEPTION D'UN PROCESSEUR À DONNÉES ANCILLAIRES INTÉGRÉ

YANN DESLAURIERS
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET
DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
AOÛT 1999

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

CONCEPTION D'UN PROCESSEUR À DONNÉES ANCILLAIRES INTÉGRÉ

présenté par : DESLAURIERS Yann

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. SAWAN Mohamad, Ph.D., président

M. SAVARIA Yvon, Ph.D., membre et directeur de recherche

M. BOIS Guy, Ph.D., membre

REMERCIEMENTS

Je tiens tout d'abord à remercier mon directeur, M. Yvon Savaria pour m'avoir guidé tout au long de mon projet de maîtrise.

Je tiens également à remercier Gilles Chouinard et Jocelyn Ouellet de Miranda Technologies Inc. pour leurs contributions à ce projet.

Merci à M. Jean-Marc Tremblay pour avoir donné de son temps et de ses énergies dans les innombrables simulations fonctionnelles nécessaires tout au long de ce projet.

Un merci tout spécial à Mme Marguerite Desjardins et M. André Desjardins pour m'avoir grandement aidé dans la poursuite d'études supérieures.

Finalement, merci à mes parents Murielle et Fernand. Il faudrait au moins 5 ouvrages de taille équivalente à ce mémoire pour décrire ce qu'ils ont fait pour moi.

RÉSUMÉ

La télévision numérique commence à prendre sa place peu à peu sur le globe et elle remplacera bientôt totalement la télévision analogique que nous connaissons aujourd'hui. Plusieurs produits sont requis pour traiter ces signaux de télévision numérique. La compagnie Miranda Technologies Inc. se spécialise dans le traitement des signaux de télévision et a développé une première version sous la forme d'une carte d'un processeur à données ancillaires. La fonctionnalité de ce produit est l'insertion/extraction de données ancillaires dans/d' un signal de télévision numérique. La latence de traitement de cette première version du processeur est de 40 us, ce qui est beaucoup trop lent. De plus, cette première version est incapable de traiter les signaux de télévision numériques de format HDTV (High Definition TeleVision).

Ce mémoire propose l'architecture d'une nouvelle version du processeur à données ancillaires, dont l'objectif est de diminuer considérablement le temps de traitement du processeur, de permettre le traitement des signaux de télévision numériques HDTV et d'ajouter plusieurs autres fonctionnalités destinées à permettre le traitement le plus adéquat possible des signaux de télévision, en fonction des besoins de l'utilisateur.

En raison de la fréquence élevée de traitement des signaux de télévision numériques HDTV, d'un besoin de mémoires RAM interne et d'une partie analogique (non traitée dans ce mémoire), l'architecture du processeur est implantée dans un circuit dédié. La complexité du processeur à données ancillaires intégré se chiffre en une centaine de modules qui, synthétisés, produisent un circuit d'environ 130 000 portes logiques (4 transistors par porte). L'architecture proposée a une latence de moins de 3 us et est capable de traiter les signaux numériques HDTV. De plus, cette architecture est dotée d'une grande flexibilité qui permettra éventuellement de traiter des types de données ancillaires qui ne sont pas encore définis présentement.

ABSTRACT

In these days, digital television is replacing little by little the old television system which is analog television. As a consequence, we need to develop new products to handle the digital television signals. Miranda Technologies Inc. is a company specialised in video and audio processing. They developped a board version of an ancillary data processor (ADP). The ADP functionality is to insert/remove ancillary data into/from a digital video stream. The processing latency of this first version of the ADP is 40 us, which is way too slow. The ADP is also unable to process HDTV (High Definition TeleVision) signals, which is very important for the future.

This thesis is proposing a new architecture of an ancillary data processor. The goals we are reaching for are to reduce the processing time, to be able to process HDTV signals and to add other functionalities to offer several processing possibilities on the digital video stream.

Because of the high frequency of HDTV signals, the need of internal RAM memories and the need of an analog part (not described in this thesis) to reach our goals, the architecture of the new ancillary data processor is implemented into an ASIC (Application Specific Integrated Circuit). This architecture is divided into several modules (almost 100 modules) which represent approximately 130 000 gates (4 transistors per gate). The processing latency obtained by this architecture is less than 3 us, which is excellent. The new ADP is also capable of processing HDTV video signals and has a certain flexibility allowing eventually the ADP to process ancillary data formats yet to be defined.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

INTRODUCTION

Depuis son invention, la télévision n'a cessé de gagner en popularité et est devenue un des plus importants moyens de communication de la société moderne. Depuis ses débuts, et aujourd'hui encore, nous avons accès à la télévision dite analogique. Avec le développement des processeurs, des circuits intégrés reprogrammables (exemple FPGA), des circuits intégrés dédiés, et autres composants électroniques, la télévision numérique s'est développée et est maintenant à nos portes.

Dès 1998, la télévision numérique a commencé à être implantée aux États-Unis. Dans ce domaine, le Canada a environ un an et demi de retard sur son voisin du sud. De plus, la technologie de la télévision numérique est déjà en pleine évolution avec la télévision à haute définition (High Definition TeleVision). Dans ce domaine, selon le réseau de diffusion américain FOX, la télévision à haute définition devrait faire son apparition d'ici 3 ou 4 ans. Ce type de télévision révolutionnera, entre autres, le format des écrans pour la diffuser.

Les avantages de la télévision numérique sont indéniables. C'est la qualité de l'image qui sera tout d'abord améliorée. En effet, avec la télévision numérique, l'information définissant chacun des pixels d'une image est plus précise. De plus, avec la télévision à haute définition, la taille des pixels se raffinera. Ainsi, l'information de chaque pixel donnera une qualité d'image plus précise. Les améliorations se feront également au niveau du son, où une qualité sonore équivalente à celle retrouvée sur un disque compact devrait se retrouver sur les futurs modèles. Dans ce domaine, la technologie de l'audio s'est également raffinée et certains produits associés à la télévision numérique ont pour fonction l'insertion d'échantillons audio numériques dans les signaux de télévision.

Ce processus de changement de la télévision analogique vers la télévision numérique nécessite la mobilisation de plusieurs groupes d'intervenants. Ainsi, au niveau du poste

de télévision même, ces derniers doivent être modifiés pour afficher correctement l'information codée de façon numérique. Il en va de même pour les produits nécessaires au diffuseur. Ces produits doivent être développés pour traiter adéquatement les divers signaux intervenants dans le domaine de la télévision numérique. Ainsi, plusieurs compagnies se sont lancées dans ce domaine, pour concevoir la multitude de produits nécessaires à l'émergence de cette nouvelle technologie.

La compagnie Miranda Technologies Inc. se spécialise dans le développement de divers produits reliés au traitement des signaux de télévision analogique et numérique. Cette compagnie a déjà réalisé une première version d'un produit appelé : « Ancillary Data Processor ». En matière de télévision, il existe à l'intérieur de la plage des données une région nommée région ancillaire. Cette région correspond, au niveau temporel, au temps pris par le canon à électrons d'un appareil de télévision pour revenir à la gauche de l'écran après avoir balayé les différents pixels contenus dans une ligne de l'écran. Pendant cette période de temps, le processeur à données ancillaires doit traiter le signal vidéo reçu, signal qui contient autre chose que des informations reliées à l'image.

La première version de ce processeur réalisée et commercialisée par Miranda Technologies Inc. est ce qu'on appelle une version carte, i.e. elle a été réalisée avec plusieurs puces, interconnectées ensemble sur une carte. Le CPU (Central Processing Unit) de cette carte est un processeur de traitement de signaux (DSP) : le TMS320C51. On retrouve également sur cette carte quelques circuits intégrés dédiés, des mémoires ROM et RAM, ainsi que quelques circuits intégrés reprogrammables. Cette carte a deux modes de fonctionnement : MUX et DEMUX. La latence de traitement des signaux vidéos, dans ces modes, est de 1,4 us et de 40 us. La latence de 40 us est tout simplement trop élevée et on doit donc trouver un moyen de l'abaisser à des valeurs de l'ordre de 3 à 4 us. De plus, cette version carte traite de signaux vidéo numériques qui arrivent à des fréquences maximales de 360 Mbits/s. La télévision à haute définition a une fréquence de 1,485 Gbits/s. La version carte ne peut tout simplement pas traiter les

signaux HDTV. Une des applications majeures de ce produit est l'insertion ou l'extraction d'échantillons audio numériques dans un signal vidéo. Pour cette application, la synchronisation entre les fréquences audio et vidéo est essentielle. Plus précisément, il doit exister une gigue de phase de moins de 1ns résultant du processus de synchronisation entre ces deux fréquences.

Les quelques lacunes de la première version carte du processeur à données ancillaires additionnées à la forte pression qui vise à miniaturiser cette fonctionnalité, ont jeté les bases de ce projet, qui est une étape en vue de réaliser une nouvelle version de ce produit. Le processus menant au choix de l'implantation d'un circuit électronique destiné à une certaine application est le suivant. Il existe plusieurs processeurs très puissants dont certains sont spécialisés dans le domaine du traitement de signal (DSP). On peut se demander s'il est possible de se servir d'un de ces processeurs pour réaliser la fonctionnalité voulue. Si les processeurs existants ne peuvent pas réaliser la fonctionnalité, on pourrait envisager de la faire avec des circuits intégrés programmables. Dans le cas qui nous intéresse, la prise de décision sur la façon d'implanter la fonctionnalité recherchée a été relativement facile. En effet, la première version du processeur à données ancillaires a été faite sur une carte avec comme pièce maîtresse, un processeur de traitement de signal relativement puissant. Cette version a été commercialisée et on a pu se rendre compte de ses limites. La seconde possibilité s'est vite retrouvée impossible à réaliser. En effet, pour répondre aux questions de synchronisation entre les horloges vidéo et audio, l'analyse du problème a rapidement conduit à une méthode basée sur un circuit comprenant certains éléments numériques, mais aussi plusieurs éléments analogiques. Or on sait que les circuits intégrés programmables (FPGA, etc.) n'offrent pas la possibilité de réaliser des circuits analogiques. Une autre considération relative aux circuits intégrés programmables est qu'en 1997, au début du projet, il était pratiquement impensable d'utiliser ce genre de circuit pour traiter les signaux HDTV requérant une fréquence d'opération de 1,485 Gbps. Ce faisant, il est apparu rapidement que la meilleure façon de réaliser la

fonctionnalité désirée serait de l'implanter dans un circuit intégré dédié (ASIC) qui permet de combiner des circuits numériques et analogiques.

Le premier objectif de ce mémoire est de faire un bref survol de la télévision numérique et des standards audio. Ceci amène à plusieurs autres objectifs de cet ouvrage dont notamment la description détaillée du processeur à données ancillaires intégré. Nous présentons la motivation ayant conduit à un tel circuit, les objectifs de performance ainsi que la façon dont ils ont été atteints. Finalement, cet ouvrage décrit quelques contributions qu'il amène : l'analyse de la flexibilité d'un circuit intégré, l'analyse de la latence de traitement d'un circuit intégré ainsi que l'architecture bit-sérielle. Le détail du mémoire de maîtrise s'énonce comme suit.

Le mémoire de maîtrise couvre la partie numérique de la nouvelle version du processeur à données ancillaires. Le premier chapitre du mémoire est une revue des standards de télévision numériques et de certains autres types de données. Il débute par une explication de la représentation temporelle des signaux de synchronisation d'une image de télévision numérique. Il se poursuit par la description du format d'encodage de données autres que celles reliées à l'image : le paquet ancillaire. Il se termine par la description de quelques types de données qui sont traitées par le processeur à données ancillaires.

Le deuxième chapitre du mémoire est celui de la présentation de la première version du processeur à données ancillaires et de l'établissement des spécifications fonctionnelles qui doivent être réalisées par la version intégrée du processeur. Plus en détail, une brève description de la fonctionnalité du premier processeur ouvre le chapitre. Ensuite, il se poursuit par la présentation de certains résultats obtenus avec cette version. Il se termine par la présentation de nouvelles spécifications dans laquelle on y retrouve, entre autres, la présentation sous forme algorithmique de la fonctionnalité de la version intégrée du processeur.

Le troisième chapitre présente en détail l'architecture du processeur à données ancillaires intégré. Il est décomposé en deux principales parties. La première est la description sommaire des différents modules du processeur autres que le contrôleur global. Nous y présentons entre autres, l'architecture de la machine bit-sérielle qui est un élément important du processeur. La description de l'architecture de la machine, la description du jeu d'instructions lui étant associée ainsi que la présentation de la mise en œuvre du jeu d'instructions sont inclus dans cette première partie. La deuxième est celle du contrôleur global du processeur qui comme nous le verrons, possède plusieurs modules œuvrant de manière parallèle. Ce contrôleur global est divisé en quatre modules principaux dans lesquels on explique l'architecture de la partie analysée ainsi que les différentes interactions qu'elle a avec les autres modules formant le circuit.

Le quatrième et dernier chapitre discute des principaux résultats obtenus avec cette architecture du processeur à données ancillaires. Nous discutons de la latence de traitement, d'une brève analyse de complexité du processeur au niveau nombre de portes logiques, ainsi que des résultats de simulation qui rendent compte de l'efficacité du processeur.

CHAPITRE 1

STANDARDS DE TÉLÉVISION NUMÉRIQUE

Ce chapitre rassemble les éléments de base sur lesquels s'appuie le processeur à données ancillaires. Il est divisé en quatre parties. La première explique le format des différents standards de télévision numérique. La deuxième décrit une trame de télévision numérique. La troisième est une description du format des paquets ancillaires qui, comme nous le verrons à travers cet ouvrage, reviennent à tout moment. Dans la quatrième partie, trois types de données sont présentés. On y verra l'information qu'ils contiennent, ainsi que leur transformation en format ancillaire.

1.1 Les formats de télévision numérique

Il existe sur la planète trois principaux formats de télévision. Tel que discuté plus tôt, présentement la télévision est analogique. Ainsi, ces trois formats sont les suivants :

1) NTSC (National Television System Committee) 525 lignes/30 Hz
2) PAL (Phase Alternating Line) 625 lignes/25 Hz
3) SECAM (SEquential Couleur Avec Mémoire) 625 lignes/25 Hz

Le format SECAM ne sera pas discuté dans cet ouvrage parce qu'il n'est pas traité par le processeur à données ancillaires. À coté des différents noms, on voit le nombre de lignes que contient une trame (frame) vidéo de ce format. Autrement dit, ce nombre correspond au nombre de lignes dont est formée une image dans le format de télévision indiqué. Suivant le nombre de lignes, on trouve la fréquence à laquelle sont diffusées ces trames sur un écran de télévision.

Voici à titre d'information le format de télévision employé dans quelques pays :
NTSC : Canada, États-Unis d'Amérique, Mexique, Pays-Bas, Japon, plusieurs pays d'Amérique centrale, etc.
PAL : Australie, Allemagne, Finlande, Royaume-Uni, Norvège, Chine, Israël, etc.

Avec la technologie numérique, un quatrième format de télévision a vu le jour. Il s'agit de la télévision à haute définition (HDTV). Il existe plusieurs formats de télévision à haute définition. L'un des plus courants contient 1125 lignes par trame vidéo et la fréquence de diffusion des trames est de 60 Hz.

1.2 La représentation spatiale des signaux de synchronisation d'une trame de vidéo numérique

Il existe des produits qui transforment des sources de vidéo analogique en vidéo numérique. On peut transformer ces sources d'une façon nommée « composites » ou d'une autre : « composantes ». Les composantes sont codées sur des mots de 10 bits alors que les composites le sont sur des mots de 8 bits. Il existe également d'autres différences au niveau des pulsations de synchronisation que l'on retrouve dans un signal de télévision, mais il serait ici non pertinent de les exposer. Par contre, il est important de dire que le processeur à données ancillaires ne traite que des formats de vidéo numériques exprimés selon les composantes. La figure 1.1 est la représentation spatiale des signaux de synchronisation d'une trame de vidéo numérique exprimée selon les composantes.

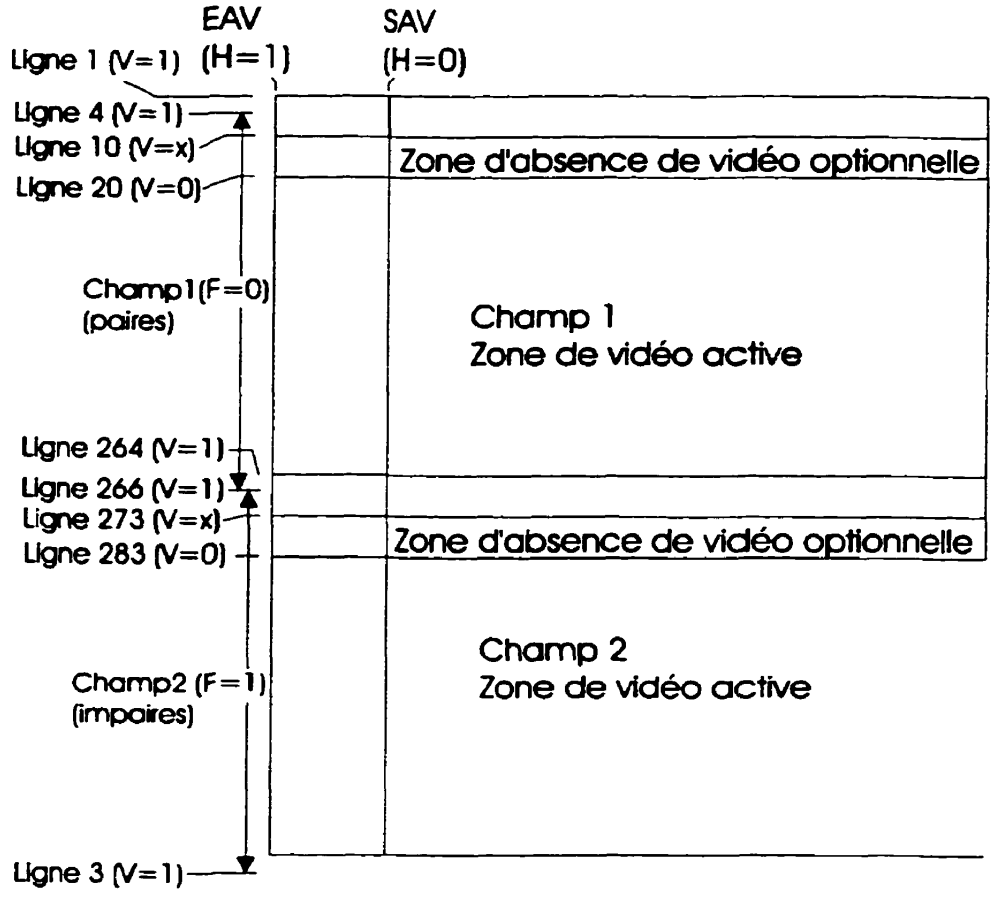


Figure 1.1 Représentation spatiale des signaux de synchronisation d'une trame de vidéo numérique

Cette représentation est celle d'une trame de vidéo numérique à 525 lignes ayant des composantes réalisées en format dit « standard », i.e. en 4 :2 :2. Ces trois chiffres ont rapport à la façon dont sont entrelacés les mots de 10 bits définissant la luminance et la chrominance d'une image de télévision. Ainsi, cet exemple pourrait bien être le résultat de la conversion numérique du format de télévision analogique NTSC.

Comme on peut le voir, une trame de vidéo numérique contient deux champs. Le champ numéro 1 contient toute l'information visuelle des lignes paires d'une image alors que le champ numéro 2 contient celle des lignes impaires. Notons ici que bien que le canon à

électrons d'un écran de télévision balaie toutes les lignes paires avant de balayer ensuite les lignes impaires, nous avons l'impression que toutes les lignes d'une image sont balayées en même temps. Chaque champ est composé de trois zones. La zone de vidéo active contient les informations liées à l'image i.e. les valeurs de luminance et de chrominance. La zone située à la gauche de la figure et délimitée par les séquences EAV et SAV (voir plus loin la signification) est nommée zone HANC (Horizontal ANCillary). On peut dire qu'on entre dans cette zone chaque fois que le canon à électrons d'un écran a fini de balayer une ligne et revient pour balayer la suivante. Entre les zones de vidéo actives se trouvent des zones de 19 lignes de largeur nommées : intervalles d'absence de vidéo (vertical blanking intervals). Cette zone est rencontrée à chaque fois que le canon à électrons a balayé le dernier pixel de la dernière ligne d'un champ et doit se positionner au premier pixel de la première ligne du champ suivant. Cette zone doit contenir au moins 9 lignes de « black vidéo ». Dans les 10 autres lignes, on peut y trouver du black vidéo ou des données d'une image.

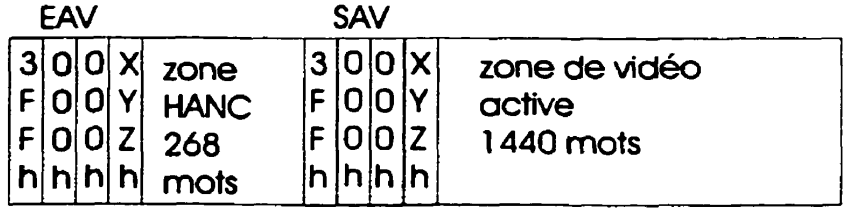


Figure 1.2 Une ligne de vidéo numérique

La figure 1.2 montre les détails d'une ligne de vidéo numérique. La séquence EAV (End of Active Video) est formée de 4 mots de 10 bits. Le dernier prend la valeur suivante : « 1 F V H P3 P2 P1 P0 0 0 » où chacun des champs est binaire et est encodé tel que spécifié:

F = 0 quand on se trouve dans le champ 1 d'une trame

F = 1 quand on se trouve dans le champ 2 d'une trame

V = 0 quand on est dans la zone de vidéo active

V = 1 quand on est dans l'intervalle vertical d'absence de vidéo

H = 1 quand il s'agit de la séquence EAV

H = 0 quand il s'agit de la séquence SAV

P3, P2, P1 et P0 sont des bits de protection.

Entre la séquence EAV et la séquence SAV (Start Active Picture), on retrouve la zone HANC qui dans la figure 1.2, contient 268 mots. Après la séquence SAV, il y a les 1440 mots qui décrivent l'image de la ligne balayée de l'exemple. Tous les formats de vidéo numérique contenant des composantes ont une structure semblable à celle présentée aux figures 1.1 et 1.2. Dans le cas de la télévision à haute définition, seules les séquences EAV et SAV diffèrent un peu. En effet, la séquence EAV se décrit comme suit :

3FFh, 3FFh, 000h, 000h, 000h, 000h, XYZh, XYZh, LN0h, LN0h, LN1h, LN1h, CR0, CR0, CR1, CR1

où la séquence (3FFh, 000h, 000h, XYZh) est doublée, LN0 et LN1 contiennent l'information renseignant sur le numéro de la ligne en cours, CR0 et CR1 sont des mots de CRC (Cyclic Redundancy Codes) utilisés pour la détection d'erreurs.

Tableau 1.1 Paramètres des différents formats de vidéo numérique exprimés en composantes

Format	Nombre de lignes/trame	Fréquence (Mbps)	Nombre de Mots dans HANC	Nombre de Mots par ligne
NTSC D1	525	270	268	1716
NTSC 16 :9	525	360	360	2288
PAL D1	625	270	280	1728
PAL 16 :9	625	360	376	2304

Le tableau 1.1 contient les informations relatives aux formats de vidéo numérique (composantes) traités par le processeur à données ancillaires. Notons que la fréquence correspond au nombre de bits qui arrivent au processeur, par seconde. En ce qui a trait à

la télévision à haute définition, un exemple de caractéristiques d'un format HDTV traité par le processeur est le suivant :

Fréquence : 1,485 Gbps

Nombre de lignes par trame : 1125

Nombre de mots dans la zone HANC : 536

Nombre de mots par ligne : 4400

1.3 Le format des paquets ancillaires

Comme on a vu dans la section précédente, il existe une zone HANC, à chaque ligne, qui ne contient pas de données reliées à l'image vidéo proprement dite. Cette zone peut être utilisée pour insérer plusieurs types de données dans des paquets ancillaires.

0 0 0 h	3 F F h	3 F F h	D I D h	D B N h	D C h	UDWs maximum 255	C S h

Figure 1.3 Paquet ancillaire

L'entête du paquet ancillaire possède toujours 6 mots. Rappelons que tous les mots d'un format de vidéo numérique exprimé en composantes sont codés sur 10 bits. Il en est de même pour tous les mots d'un paquet ancillaire. Les trois premiers mots sont des constantes. Le quatrième, DID (Data IDentification), est un mot renseignant sur le type de données que contient le paquet ancillaire. Ces DID sont gérés par l'organisme SMPTE (Society of Motion Picture and Television Engineers) qui assigne les DID pour chaque type de données. On peut retrouver ces informations en consultant le SMPTE Journal qui contient toutes les normes en matière de télévision. Le mot suivant est appelé DBN (Data Block Number) . Ce mot est utilisé pour compter une séquence de plusieurs paquets ancillaires de même DID. Bien que comprenant 10 bits, l'information

pertinente renseignant sur le numéro du paquet se retrouve sur les 8 bits les moins significatifs du mot. Les paquets sont numérotés de 1 à 255 (la valeur DBN = 0 n'est pas utilisée) et lorsque la valeur 255 est atteinte, le paquet suivant prend la valeur de 1. Le mot suivant est le DC (Data Count). Le même principe que le mot DBN s'applique à ce mot i.e. bien qu'il soit codé sur 10 bits, seuls les 8 bits les moins significatifs du mot sont pertinents à l'information qu'il contient. Cette information renseigne sur le nombre de UDWs (User Data Words) que contient le paquet. Les mots suivants sont les UDWs. Ils contiennent l'information, codée en format ancillaire, propre à un certain type de données. La section suivante traite de quelques types de données que l'on peut retrouver dans des paquets ancillaires. Encore une fois, la façon de coder les bits de tel type de données pour qu'ils puissent être insérés dans des paquets ancillaires est régie par SMPTE. Le dernier mot de tout paquet ancillaire est le CS (Check Sum). Il est utilisé pour la détection d'erreurs que pourrait contenir le paquet ancillaire. On le code en prenant les neufs bits les moins significatifs de la somme des neufs bits les moins significatifs de l'intervalle des mots du paquet allant du mot DID jusqu'au dernier UDW. Le bit numéro 9 (le 10^e bit du mot) est l'inverse du bit numéro 8.

1.4 Les types de données qu'on peut rencontrer dans un signal de vidéo numérique

Nous avons vu dans la section 1.2, la structure d'un signal de vidéo numérique. Dans cette même section, nous avons également vu qu'il existe une portion de chaque ligne qui ne contient pas de données se rapportant à l'image. Cette région se nomme HANC (Horizontal ANCillary) et peut être utilisée pour insérer des paquets ancillaires. La section précédente a décrit le format des paquets ancillaires qui contiennent des données d'un certain type. Dans cette section, nous allons voir le format de certains types de données ainsi que les transformations à opérer pour les insérer dans des paquets ancillaires.

1.4.1 Les données audio

Le type de données le plus important au processeur à données ancillaires est l'audio. En effet, une grande partie de la fonctionnalité du processeur consiste à insérer ou extraire des données audio dans/d'un signal vidéo. Le format de l'audio qui nous intéresse est le format AES/EBU. La figure 1.4 illustre ce format.

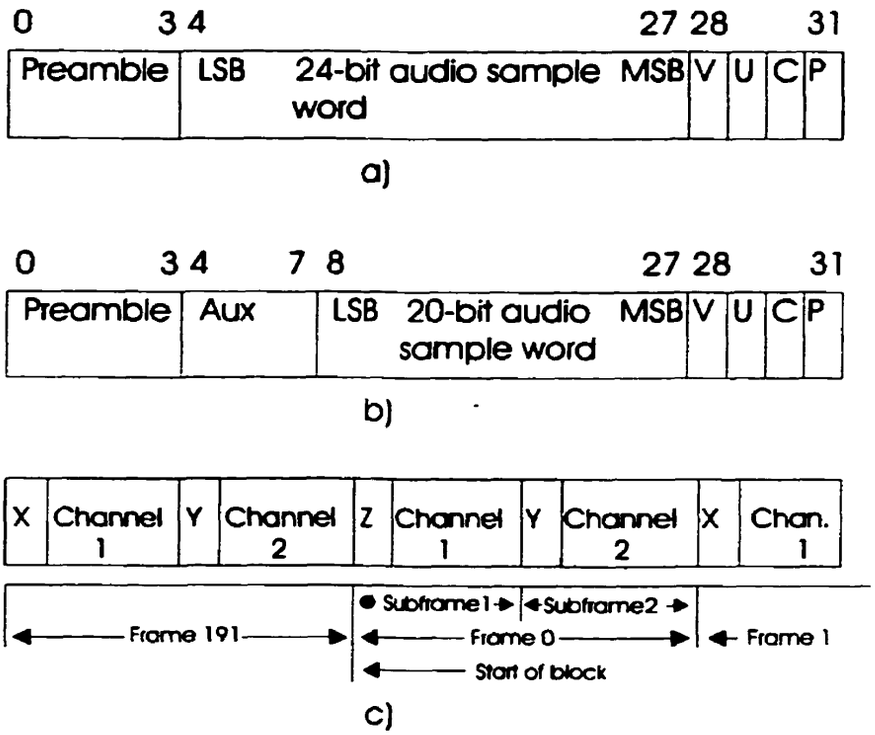


Figure 1.4 Formats d'un échantillon audio AES

Chacune des sous-trames audio est composée de 32 bits. Les 4 premiers bits forment un préambule qui indique si on a affaire à la sous-trame numéro 1 ou à la sous-trame numéro 2 d'une paire. Un bloc AES est composé de 192 trames numérotées de 0 à 191. Le premier préambule de la première sous-trame de la trame 0 est un préambule spécial (Z) qui indique le début d'un bloc AES. Les figures 1.4a) et 1.4b) illustrent le contenu des sous-trames audio. Comme on peut le voir, l'échantillon audio peut être codé sur 20 ou 24 bits. Dans le cas où il ne contient que 20 bits, les 4 bits précédents l'échantillon

contiennent des données auxiliaires (AUX) qui peuvent être de n'importe quel type. Les 4 derniers bits d'une sous-trame indiquent les informations suivantes. Le bit V, lorsqu'il est à '0', indique que l'échantillon audio est valide. Le bit U est ce qu'on appelle le « User data format » et peut être utilisé comme l'usager qui envoie les trames AES, le veut. Le bit C est le « Channel status format ». Les 192 trames AES d'un bloc forment 24 octets (mots de 8 bits) de données avec le C bit de chaque trame (le même bit C apparaît dans les deux sous-trames d'une même trame). Nous n'allons pas expliciter ici davantage l'information contenue dans ces 24 octets, mais mentionnons que ces derniers ne forment pas des informations essentielles au décodage des échantillons audio des sous-trames. Finalement, le bit P est un bit de parité des 31 bits précédents de la sous-trame. Comme la figure 1.4c) l'indique, un bloc AES contient des trames qui sont composées de 2 canaux. Un tel bloc est appelé « AES Channel-Pair 1 ». Un autre bloc indépendant prend alors le nom « AES Channel-Pair 2 ».

Chacune des sous-trames audio doit être transformée en format ancillaire pour pouvoir être insérée dans un signal vidéo. Il est intéressant de noter que les transformations à appliquer aux sous-trames sont différentes selon qu'on est en présence d'un signal vidéo HDTV ou DTV.

Tableau 1.2 UDWs des paquets ancillaires audio d'un format vidéo DTV

9	8	7	6	5	4	3	2	1	0
/5	5	4	3	2	1	0	c1	c0	Z
				X					
9	8	7	6	5	4	3	2	1	0
/P	P	C	U	V	19	18	17	16	15
				X+2					

9	8	7	6	5	4	3	2	1	0
/14	14	13	12	11	10	9	8	7	6
				X+1					

Tableau 1.3 UDWs des paquets ancillaires auxiliaires audio d'un signal vidéo DTV

9	8	7	6	5	4	3	2	1	0
/a	a	y3	y2	y1	y0	x3	x2	x1	x0

Tableau 1.4 UDWs des paquets ancillaires audio d'un signal vidéo HDTV

9	8	7	6	5	4	3	2	1	0
/P'	P'	3	2	1	0	Z	0	0	0
				X					

9	8	7	6	5	4	3	2	1	0
/P'	P'	11	10	9	8	7	6	5	4
				X+1					

9	8	7	6	5	4	3	2	1	0
/P'	P'	19	18	17	16	15	14	13	12
				X+2					

9	8	7	6	5	4	3	2	1	0
/P'	P'	P	C	U	V	23	22	21	20
				X+3					

Le tableau 1.2 démontre la transformation d'une sous-trame audio AES en 3 mots de 10 bits qui forment les UDWs des paquets ancillaires. La ligne supérieure renseigne sur le numéro du bit dans le mot et la ligne inférieure décrit son contenu. Dans le mot X, les 3 LSBs que sont Z, c0 et c1 ont les significations suivantes : Z prend la valeur '1' lorsque la sous-trame fait partie de la trame numéro 0 d'un bloc AES, autrement il prend la valeur '0'; c0 prend la valeur '1' quand il est question de la sous-trame 2 et '0' quand il est question de la sous-trame 1; c1 prend la valeur '1' quand il est question de la « AES Channel-pair 2 » et '0' quand il est question de la « AES Channel-pair 1 ». Les valeurs 19 à 0 de la ligne de contenu des mots X, X+1 et X+2 correspondent aux bits 27 à 8 des figures 1.4 a) et b). Les valeurs V, U et C sont les mêmes que dans le format AES, seul le P bit est différent. Il représente la parité calculée sur les bits 0 à 8 des mots X et X+1 ainsi que des bits 0 à 7 du mot X+2.

Lorsqu'un échantillon audio AES est codé sur 24 bits, un paquet ancillaire audio n'est pas suffisant pour contenir les bits. On doit alors fabriquer ce qu'on appelle un paquet ancillaire auxiliaire audio dont le format des UDW est représenté au tableau 1.3. Le bit numéro 8 contient le paramètre « a ». Ce paramètre prend la même valeur que le bit numéro 2 du mot X, i.e. « c1 ».

Dans le cas où on a affaire à un signal vidéo HDTV, le tableau 1.4 indique que les sous-trame audio AES sont codés en format ancillaire, sur 4 mots de 10 bits. Le format est très semblable. Nous indiquons seulement que chaque bit P' des mots est la parité des bits 7 à 0 du même mot et que le bit numéro 7 du mot X+2 (P) est le même que celui contenu dans le format AES.

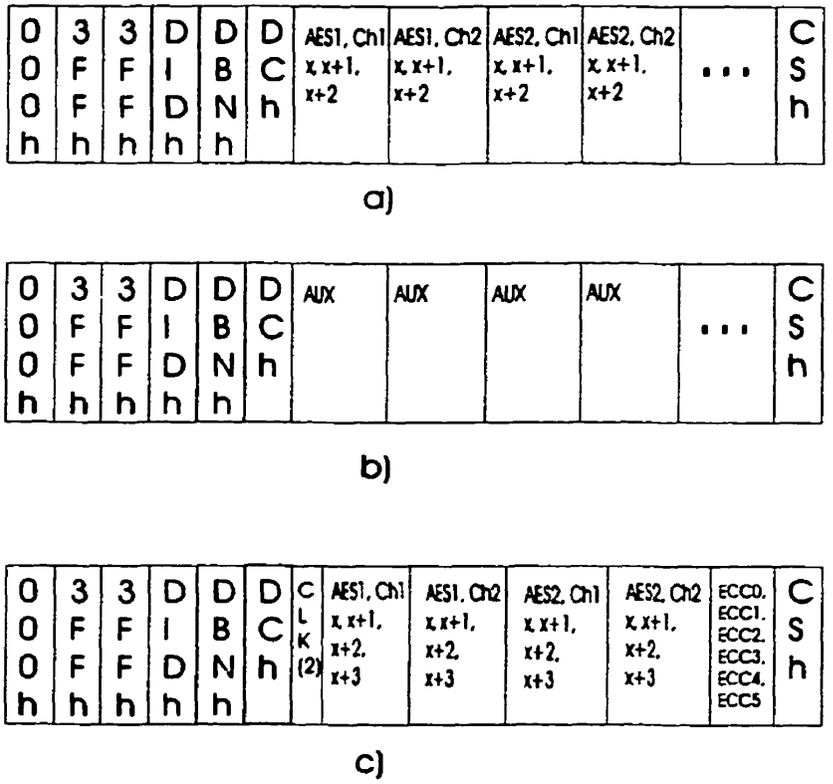


Figure 1.5 Paquets ancillaires audio

La figure 1.5 résume les différents formats de paquet ancillaire audio qu'on peut retrouver dans un signal vidéo. La figure 1.5a) représente le paquet ancillaire audio d'un

signal vidéo de DTV (Digital TeleVision). Les paires AES alternent dans un tel format. Donc, à la suite des trois mots de l'AES2 channel 2, on retrouve trois mots AES1 channel 1. Par exemple, les six premiers UDWs du paquet contiennent les sous-trames 1 et 2 de la trame 0 d'un bloc AES, le bloc AES1. Les six suivants contiennent les sous-trames 1 et 2 de la trame 0 du bloc AES2. Les six suivants contiennent les sous-trames 1 et 2 de la trame 1 du bloc AES1. Les six suivants contiennent les sous-trames 1 et 2 de la trame 1 du bloc AES2 et ainsi de suite. Il est recommandé par SMPTE que les paquets ancillaires audio de ce type ne contiennent que 3 ou 4 trames des blocs AES1 et AES2 et donc que le paquet soit formé de 36 ou 48 UDWs.

La figure 1.5b) représente la structure d'un paquet ancillaire auxiliaire audio et selon que le paquet ancillaire audio auquel il se rattache est formé de 36 ou de 48 UDWs, ce paquet sera formé de 6 ou 8 UDWs (voir section 2.1.2 pour plus d'informations).

La figure 1.5c) représente la structure d'un paquet ancillaire audio d'un signal vidéo HDTV. Un paquet audio de ce type contient uniquement une trame d'un bloc AES1 et une trame d'un bloc AES2. Les autres UDWs qui composent le paquet : CLK (2 mots) et ECC (6 mots) sont des mots servant à la synchronisation de l'audio par rapport au signal vidéo ou à la détection d'erreurs dans le paquet et ne seront pas explicités ici.

1.4.2 Les données LTC et RS422

Un autre type de données traité par le processeur est le LTC (Linear Time Code). Ce type de données est utilisé comme adresse temporelle d'une trame vidéo. Cette adresse est exprimée en heure, en minute, en seconde ainsi qu'en numéro de trame. Pour ce qui est de la partie temps de l'adresse, elle s'incrémente de 0 heure, 0 minute, 0 seconde jusqu'à 23 heures, 59 minutes, 59 secondes. Le numéro de trame s'incrémente de un à chaque nouvelle trame sur l'intervalle 0 à 23. Un seul échantillon LTC arrive par

trame. Un échantillon LTC est constitué de 80 bits, comme on peut voir dans le tableau suivant.

Tableau 1.5 Bits composant un échantillon LTC

0-3 Units of frame	4-7 1st binary group	8-9 Tens of frame	10 Drop frame flag	11 Color frame flag
12-15 2nd binary group	16-19 Units of second	20-23 3rd binary group	24-26 Tens of second	27 Polarity correction
28-31 4th binary group	32-35 Units of minute	36-39 5th binary group	40-42 Tens of minute	43 BGF0
44-47 6th binary group	48-51 Units of hour	52-55 7th binary group	56-57 Tens of hour	58 BGF1
59 BGF2	60-63 8th binary group	64-79 0011111111111101		

Outre les heures, les minutes, les secondes et le numéro des trames, un LTC contient huit « binary group » dans lesquels l'usager peut insérer des données spéciales et dont les « flags » BGF0 (Binary Group Flag), BGF1 et BGF2 indiquent ce qu'elles contiennent. Les 16 derniers bits du LTC sont un mot de synchronisation qui dit que l'échantillon reçu est bel et bien un LTC. Ceci est nécessaire car les bits du LTC sont transmis sériellement. Un paquet ancillaire formé de données LTC s'appelle ATC (Ancillary Time Code) et à la forme suivante :

Tableau 1.6 Paquet ancillaire contenant des données LTC

000h	3FFh	3FFh	DID	DBN	DC	UDW 1-16	CS

Où le format général des UDWs est le suivant :

Tableau 1.7 Format général des UDWs d'un paquet ancillaire ATC

B0	0
B1	0
B2	0
B3	0
B4	Anc. binary group (lsb)
B5	Anc. binary group
B6	Anc. binary group
B7	Anc. binary group (msb)
B8	Even parity (b0-b7)
B9	/b8

Le tableau 1.8 donne le contenu de chacun des 16 UDWs formant un paquet ancillaire ATC. À titre de précision, TC0 est le bit de position 0 de l'échantillon LTC tel que montré au tableau 1.5.

Tableau 1.8 UDWs d'un paquet ancillaire ATC

UDW	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
1	0	0	0	0	TC0	TC1	TC2	TC3	Even parity (b0-b7)	/b8
2	0	0	0	0	TC4	TC5	TC6	TC7	Even parity (b0-b7)	/b8
3	0	0	0	0	TC8	TC9	TC10	TC11	Even parity (b0-b7)	/b8
4	0	0	0	0	TC12	TC13	TC14	TC15	Even parity (b0-b7)	/b8
5	0	0	0	0	TC16	TC17	TC18	TC19	Even parity (b0-b7)	/b8
6	0	0	0	0	TC20	TC21	TC22	TC23	Even parity (b0-b7)	/b8
7	0	0	0	0	TC24	TC25	TC26	TC27	Even parity (b0-b7)	/b8
8	0	0	0	0	TC28	TC29	TC30	TC31	Even parity (b0-b7)	/b8
9	0	0	0	0	TC32	TC33	TC34	TC35	Even parity (b0-b7)	/b8
10	0	0	0	0	TC36	TC37	TC38	TC39	Even parity (b0-b7)	/b8
11	0	0	0	0	TC40	TC41	TC42	TC43	Even parity (b0-b7)	/b8
12	0	0	0	0	TC44	TC45	TC46	TC47	Even parity (b0-b7)	/b8
13	0	0	0	0	TC48	TC49	TC50	TC51	Even parity (b0-b7)	/b8
14	0	0	0	0	TC52	TC53	TC54	TC55	Even parity (b0-b7)	/b8
15	0	0	0	0	TC56	TC57	TC58	TC59	Even parity (b0-b7)	/b8
16	0	0	0	0	TC60	TC61	TC62	TC63	Even parity (b0-b7)	/b8

Un autre type de données manipulé par le processeur est le type RS422. Ce type de paquet a été créé par la compagnie Miranda. Le format ancillaire de ces paquets n'est pas définitif et c'est pourquoi nous ne le présenterons pas en détail comme les autres. Mentionnons simplement que présentement, les paquets ancillaires contenant des données RS422 sont constitués d'un seul UDW. L'information que contient ce type de données à trait aux sous-titres que l'on peut remarquer quelques fois au bas de l'écran d'un téléviseur.

CHAPITRE 2

LES SPÉCIFICATIONS FONCTIONNELLES DU PROCESSEUR

Ce chapitre est divisé en deux. La première partie traite de la version carte du processeur à données ancillaires telle que réalisée par la compagnie Miranda. Nous y expliquons les fonctionnalités qui, dans la plupart des cas, se retrouvent sur la version intégrée du processeur. Nous discutons aussi de quelques résultats obtenus avec cette version. Dans la deuxième partie, nous présentons les fonctionnalités de la version intégrée à l'aide, entre autres, d'un algorithme.

2.1 La version carte du processeur à données ancillaires

On doit tout d'abord mentionner que la version carte ainsi que la version intégrée du processeur ont deux modes de fonctionnement : le mode MUX et le mode DEMUX. Dans le mode MUX, le processeur reçoit sur des lignes sérielles, plusieurs types de données. Avec ces données, le processeur fabrique des paquets ancillaires selon les standards SMPTE et insère ces paquets aux endroits appropriés dans le signal de vidéo numérique. Dans le mode DEMUX, l'inverse se produit. Le processeur reçoit les paquets ancillaires déjà présents dans le signal vidéo. De ces paquets, il traite seulement les UDWs qu'il doit sérialiser (toujours selon certains standards) vers le monde extérieur. La figure 2.1 résume bien la fonctionnalité globale des deux versions du processeur à données ancillaires, bien qu'elle représente la fonctionnalité de la version carte. On représenterait simplement la version intégrée en faisant passer les ports « vidéo d'entrée » et « vidéo de sortie » de 10 à 20 bits.

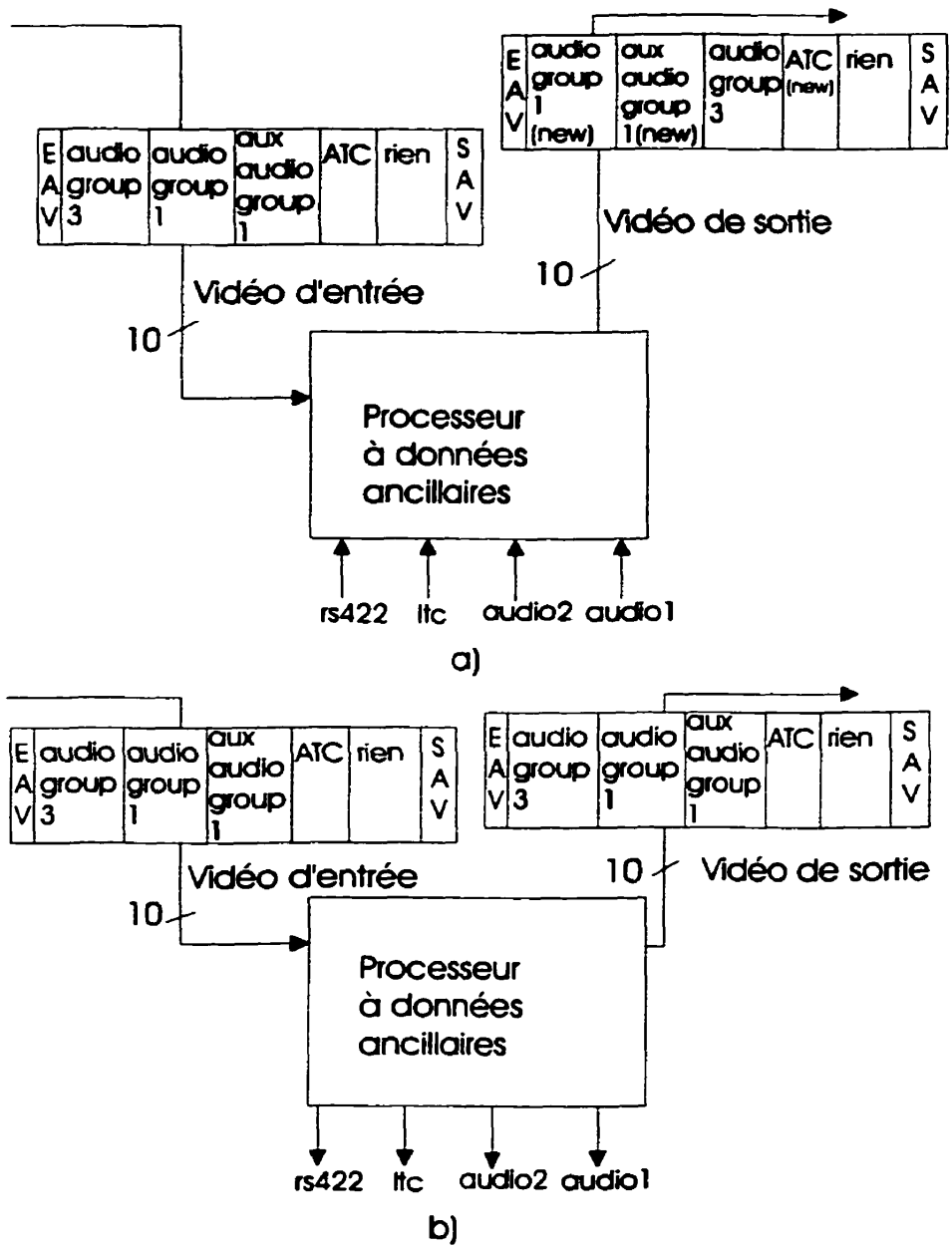


Figure 2.1 Fonctionnalité de la version carte de l'ADP

La figure 2.1 a) représente la fonctionnalité du mode MUX dans lequel on forme des nouveaux paquets ancillaires à partir de lignes sérielles de données et ces paquets ancillaires sont ensuite insérés dans le signal vidéo. La figure 2.1 b) représente la

fonctionnalité du mode DEMUX dans lequel les UDWs de certains paquets ancillaires sont sérialisés vers le monde extérieur.

2.1.1 Caractéristiques générales des deux modes de fonctionnement

Dans les deux modes du processeur, il y a cinq sous modes :

1) normal
2) bypass
3) tones
4) colour bar
5) tones and colour bar.

Un des modules du processeur est responsable de la détection du format vidéo en présence. Ceci est évidemment nécessaire au reste de la carte pour le traitement adéquat du signal vidéo qui rappelons-le, a des caractéristiques, telles le nombre de lignes par trame, différentes selon le format vidéo en présence.

Une autre des fonctionnalités rattachées à la version carte et propre aux deux modes de fonctionnement est l'EDH (Error Detecting and Handling). Cette fonctionnalité opère sur chaque champ de chaque trame vidéo. Elle peut se diviser en trois. L'EDH reliée aux paquets ancillaires, l'EDH reliée à l'information de la zone active de vidéo et l'EDH reliée au champ en entier. L'EDH des paquets ancillaires est simplement le calcul du CS (Check Sum) de chaque paquet ancillaire. Celle des deux autres est un calcul de CRC (Cyclic Redundancy Codes) sur leurs zones respectives. Nous n'entrerons pas dans les détails ici, mais mentionnons que la version carte du processeur fabrique des paquets ancillaires EDH. Ces paquets contiennent des flags qui renseignent sur les types d'erreurs détectées par le processeur, les flags d'erreurs détectés préalablement par d'autres circuits, ainsi que les résultats des nouveaux calculs de CRC faits par le processeur. Un paquet ancillaire EDH doit être inséré deux fois par trame vidéo (une

fois par champ) à des lignes spécifiques. De plus, ces paquets doivent être insérés à la fin de la zone HANC des lignes ciblées.

Une autre fonctionnalité également rencontrée est la communication de divers paramètres du processeur vers le monde extérieur via une interface sérielle I²C. L'interface I²C a été développée par la compagnie Philips. Mentionnons ici que quelques paramètres communiqués par le processeur sont : son mode de fonctionnement, son sous-mode de fonctionnement, le groupe audio traité (voir plus loin), les flags d'erreurs résultant des modules EDH, le format vidéo en présence, etc. Mentionnons également qu'avec cette interface sérielle, il est possible de changer certains paramètres du processeur. Ainsi, on peut changer son sous-mode de fonctionnement, changer le groupe audio qu'il traite (voir plus loin), etc.

Rappelons ici que la version carte ne traite pas les signaux vidéo HDTV. Notons également que la version carte traite des signaux vidéo formés de composantes ou de composites. Parce que ce dernier format est peu utilisé et que la version intégrée du processeur ne le traite pas, il ne sera pas explicité, car il ne ferait qu'alourdir le texte.

2.1.2 Caractéristiques du mode MUX

Sous-mode normal :

C'est le mode le plus courant du processeur. Dans ce sous-mode, on fabrique tous les types de paquets ancillaires à partir des lignes sérielles de données. Le cas de l'audio est celui qui nous intéresse le plus. Comme on a vu dans le chapitre précédent, deux paires de canaux audio AES sont nécessaires pour la fabrication de paquets ancillaires audio (AES Channel Pair 1,2). C'est pourquoi le processeur à données ancillaires a deux ports sériels « entrée/sortie » : audio1 et audio2. Un paquet ancillaire audio contient ce qu'on appelle un groupe audio (audio group sur la figure 2.1). Il y a seulement quatre groupes

audio possibles, numérotés de 1 à 4. On identifie qu'un paquet ancillaire contient un groupe audio donné par le mot DID du paquet. En effet, SMPTE a associé à chaque paquet ancillaire audio d'un certain groupe audio, un seul DID (il y a également un DID réservé pour chaque paquet ancillaire auxiliaire audio d'un certain groupe lorsque les échantillons audio sont codés sur 24 bits, DID différent de celui du paquet ancillaire audio et ce même s'ils contiennent leurs données respectives du même groupe audio). Nous pouvons dire de là que deux paires de canaux audio AES audio1 et audio2 forment un paquet ancillaire audio et un paquet ancillaire auxiliaire audio si les échantillons audio sont codés sur 24 bits, d'un certain groupe audio.

Lorsque l'on choisit de traiter des données audio dans un signal vidéo avec un processeur à données ancillaires, l'usager doit sélectionner le groupe audio qu'il désire former s'il veut opérer dans le mode MUX ou qu'il désire sérialiser s'il veut opérer dans le mode DEMUX. Dans le cas du mode MUX, plusieurs règles sont à suivre pour l'insertion de paquets. Rappelons tout d'abord formellement que les échantillons audio sont codés soit sur 20 bits, soit sur 24 bits. Des échantillons audio codés sur 24 bits sont transformés en format ancillaire en deux paquets distincts : le paquet ancillaire audio et le paquet ancillaire auxiliaire audio. Des échantillons audio codés sur 20 bits sont transformés seulement en paquet ancillaire audio. Ceci étant dit, il est obligatoire, lorsque l'on traite des échantillons audio codés sur 24 bits, d'insérer le paquet ancillaire auxiliaire audio immédiatement à la suite du paquet ancillaire audio de même groupe audio.

Deux recommandations importantes suivent. Il est d'une part fortement recommandé d'insérer les paquets audio au début de la zone HANC et d'autre part, de concaténer au début de cette zone tous les paquets ancillaires (sauf le paquet ancillaire EDH). Ceci étant dit, la version carte du processeur préconise d'insérer les nouveaux paquets audio formés au début de l'espace HANC, de faire le tri des paquets présents dans le signal vidéo et d'insérer lorsque possible, les autres paquets fabriqués par le processeur. Faire

le tri signifie que si un paquet ancillaire auxiliaire audio et/ou un paquet ancillaire audio ayant le même groupe audio que celui traité par le processeur se trouvent dans le signal vidéo d'entrée, le processeur ne les retransmet pas en sortie.

Une autre règle à suivre découle de la façon dont est défini le signal de télévision. On se rappelle qu'une trame de vidéo est composée de deux champs, l'un contenant les lignes paires d'une image et l'autre, les lignes impaires. On appelle ligne de transition la première ligne de chacun des deux différents champs d'une image. Il est interdit d'insérer des paquets ancillaires audio dans ces lignes ainsi que dans les lignes qui les suivent immédiatement.

Les données LTC (Linear Time Code) ne sont traitées que dans ce sous-mode. Un paquet ATC (Ancillary Time Code) est formé par trame vidéo, si bien qu'il est inséré également une seule fois par trame. Cette version considère que le LTC reçu est erroné si l'information du numéro de trame qu'il contient n'est pas incrémentée de un par rapport au dernier LTC reçu. Également, cette version tolère des erreurs sur le LTC sur une période de 10 trames vidéo consécutives. Après cette période, le processeur ne traite plus les données LTC. Tout comme l'audio, les paquets ancillaires ATC ne peuvent être insérés aux lignes de transition dans le signal vidéo.

Les paquets ancillaires RS422 peuvent être insérés à n'importe quelle ligne, lorsqu'ils sont prêts. Ce sont les derniers paquets ancillaires à être insérés dans une ligne, exception faite des paquets ancillaires EDH, si on se trouve dans une ligne où ces paquets sont insérés. Lorsque ces paquets sont détectés dans le signal vidéo d'entrée, ils sont automatiquement retirés.

Sous-mode bypass :

Comme son nom l'indique, dans ce sous-mode, on ne fait que retransmettre en sortie le signal vidéo que l'on reçoit sans rien ajouter ou retirer.

Sous-mode tones :

Dans ce sous-mode, le principal élément est la façon de traiter l'audio. En effet, on ne fabrique plus de paquets ancillaires audio avec les données sérielles des lignes audio1 et audio2, mais bien avec ce que l'on appelle des « tones ». Les tones sont des échantillons audio d'une seule fréquence. Dans le cas de la version carte du processeur, la fréquence choisie pour les tones est de 1kHz. Rappelons que dans ce sous-mode, les données LTC ne sont pas traitées. Par contre, les données RS422 sont traitées comme dans le cas du sous-mode normal.

Sous-mode colour bar :

Le sous-mode colour bar est spécial dans le sens où il est le seul à modifier la zone active d'un signal vidéo. En effet, lorsque l'on se retrouve dans ce sous-mode, on insère des valeurs de chrominance et de luminance qui divisent l'écran en huit bandes verticales de couleur différentes. La région HANC demeure dans ce cas, inchangée.

Sous-mode tones and colour bar :

Ce sous-mode n'est que l'intégration complète des deux sous-modes que sont le tones et le colour bar.

2.1.3 Caractéristiques du mode DEMUX

Sous-mode normal :

Ce sous-mode du mode DEMUX est le plus courant au processeur. Il est caractérisé par la sérialisation des bits formant les UDWs des paquets ancillaires que l'on veut traiter. Encore une fois, commençons par les paquets ancillaires audio. Le processeur traite les paquets ancillaires audio et les paquets ancillaires auxiliaires audio dont le DID correspond au groupe audio que l'on veut traiter. Les UDWs formant les paquets en question sont transformés en format AES pour être sérialisés vers le monde extérieur. On doit ici tout de suite dire que si aucun paquet audio du groupe sélectionné n'est présent dans le signal vidéo, le processeur envoie vers le monde extérieur des échantillons de « black audio », i.e. des échantillons dont tous les bits sont à zéro.

Dans ce mode de fonctionnement, le processeur est particulièrement attentif à deux choses. L'une d'entre elle est le DBN des paquets ancillaires audio du groupe sélectionné. En effet, on se rappelle que le mot DBN d'un paquet ancillaire indique le numéro du paquet dans une série de paquets de même DID. Parce que certains autres circuits de traitement d'audio dans des signaux vidéo se soucient peu du DBN, le processeur considérera un fonctionnement correct si le paquet ancillaire audio qu'il reçoit a son DBN égal ou supérieur de un au DBN du paquet ancillaire audio précédent. Lorsque ce n'est pas le cas, le processeur met à '1' le bit V des échantillons audio contenus dans le paquet indiquant que ces échantillons audio ne sont pas valides (voir figure 1.4). La deuxième chose est la différenciation des échantillons audio des paires AES1 et AES2 d'un paquet ancillaire audio. En effet, bien qu'il soit indiqué de fabriquer ce genre de paquets comme l'indique la figure 1.5, certains circuits les fabriquent en inversant les paires AES1 et AES2. Le processeur regarde donc le bit « c1 » des mots X du paquet reçu qui lui indique à quelle paire il a affaire (voir tableau 1.2). Les données audio émises sériellement vers le mode extérieur le sont en « biphase

mark modulation » à une fréquence de 6,144 MHz. Cette horloge doit être générée par le processeur et sa synchronisation avec l'horloge du signal vidéo est très importante.

En ce qui concerne les paquets ATC, ils ne sont que traités dans ce sous mode. Il n'y a rien de plus à dire des paquets RS422 qu'ils sont extraits et transmis au port approprié. Dans le cas des données LTC à transmettre, le processeur reçoit une horloge extérieure. C'est la même chose pour les données RS422.

Sous-mode bypass :

Ce sous-mode fonctionne presque de la même façon que le sous-mode normal. Tous les paquets ancillaires audio ayant le groupe audio sélectionné par l'usager et les paquets ancillaires RS422 (on enlève le paquet ou on le retransmet selon le choix de l'usager) sont traités par le processeur et émis sériellement vers le monde extérieur.

Sous-mode tones et colour bar :

Les sous-modes tones et colour bar fonctionnent toujours de pair dans la version carte et sont donc regroupés dans le sous-mode tones et colour bar. À l'instar du mode MUX, la zone active vidéo est modifiée en huit bandes verticales de couleurs différentes. La zone HANC reste inchangée. Pour ce qui est de l'audio, on émet sériellement vers le monde extérieur, en format AES, des échantillons spéciaux de fréquence constante de 1 kHz.

2.1.4 Performances de la version carte

En mode MUX, la latence de traitement du signal vidéo est de 40 us, ce qui est une valeur beaucoup trop élevée. Miranda a déterminé que cette latence était trop élevée en fonction des commentaires reçus des clients qui ont acheté leurs produits et en fonction de la comparaison avec d'autres produits semblables sur le marché. La latence maximale

du mode DEMUX est de 2,6 us pour un format vidéo formé de composites. Cette version du processeur traite les signaux vidéo formés de composantes ainsi que ceux formés de composites, mais ne traitent pas les signaux HDTV.

2.2 La version intégrée du processeur à données ancillaires

Plusieurs fonctionnalités de la version carte du processeur se trouvent sur la version intégrée. Ainsi, la figure 2.1 résume bien cette fonctionnalité pour autant que les bus d'entrée et de sortie comprennent 20 bits au lieu de 10 et que cette figure représente un format vidéo DTV, car en HDTV, il n'y a pas de paquets ancillaires auxiliaires audio comme mentionné dans le chapitre précédent. En effet, pour un standard HDTV, la fréquence de 1,485 Mbps permet de traiter 20 bits à la fréquence de 74,25 MHz. On doit préciser ici une chose, si les 20 bits reçus lorsque l'on se trouve dans la zone de vidéo active contiennent tous de l'information indispensable, il en est autrement de la zone HANC. En effet, c'est dans cette région où sont les paquets de données ancillaires, ces derniers se trouvent soit dans les 10 MSBs, soit dans les 10 LSBs du bus d'entrée de 20 bits. Par conséquent, la moitié de la zone HANC d'un signal HDTV ne contient pas de données pertinentes.

Dans cette section, nous présentons les fonctionnalités de cette version qui se trouvent sur la version carte ainsi que les nouvelles fonctionnalités. La version intégrée possède également les deux mêmes modes de fonctionnement.

2.2.1 Caractéristiques générales des deux modes de fonctionnement

Le processeur intégré a toujours cinq sous-modes de fonctionnement pour le mode MUX et trois pour le mode DEMUX :

1) normal (MUX, DEMUX)

2) bypass (MUX, DEMUX)
3) tones (MUX)
4) colour bar (MUX)
5) tones et colour bar (MUX, DEMUX)

On doit tout d'abord indiquer au processeur s'il traite un signal vidéo HDTV ou DTV. Ceci s'explique par le fait qu'en format HDTV, le processeur doit traiter les 20 bits, alors qu'il ne s'occupe que des 10 LSBs dans le cas d'un signal DTV. Ceci étant dit, le processeur est doté d'un module de détection du format vidéo en présence qui distingue les formats du tableau 1.1. De plus, on doit pouvoir également forcer un format vidéo du monde extérieur. On doit toujours transmettre via une interface sérielle semblable à celle de l'I^2C les paramètres du processeur au monde extérieur. Plus précisément, on veut pouvoir transmettre le groupe audio sélectionné, le sous-mode d'opération, le format vidéo en présence, les groupes audio présents dans le signal vidéo d'entrée à chaque trame vidéo, certains « flags » reliés au traitement de la zone HANC qui seront expliqués plus loin, le fait qu'il y ait ou non de l'asynchronisme audio par rapport au signal vidéo en mode DEMUX (voir plus loin), etc. De plus, on veut que l'usager puisse changer ces paramètres, lorsqu'ils peuvent l'être.

Cette version n'inclut pas de modules EDH, tout simplement parce que les nouveaux récepteurs vidéo qui d'un signal vidéo sériel le transforment en mots de 10 ou 20 bits et les nouveaux transmetteurs qui font l'inverse, incluent cette fonctionnalité.

Avec ce processeur, on veut également pouvoir traiter des types de données qui ne sont pas encore régis par SMPTE. On veut également être capable, et ce dans les deux modes de fonctionnement, d'extraire ou de retransmettre tous les paquets que l'usager doit traiter ainsi que des types de paquets spécifiés par l'usager.

2.2.2 Caractéristiques du mode MUX

Sous-mode normal :

Encore une fois, c'est l'audio qui a priorité par rapport à tout autre type de données dans cette version. La façon d'insérer les paquets ancillaires audio est semblable à celle expliquée en section 2.1.2 et ce pour les signaux vidéo HDTV et DTV. On insère toujours les nouveaux paquets ancillaires audio au début de la zone HANC, lorsque c'est permis (en HDTV, seules les lignes de transition de champ sont interdites à l'insertion de l'audio), on concatène les autres paquets présents à la suite en raffinant le tri, i.e. en

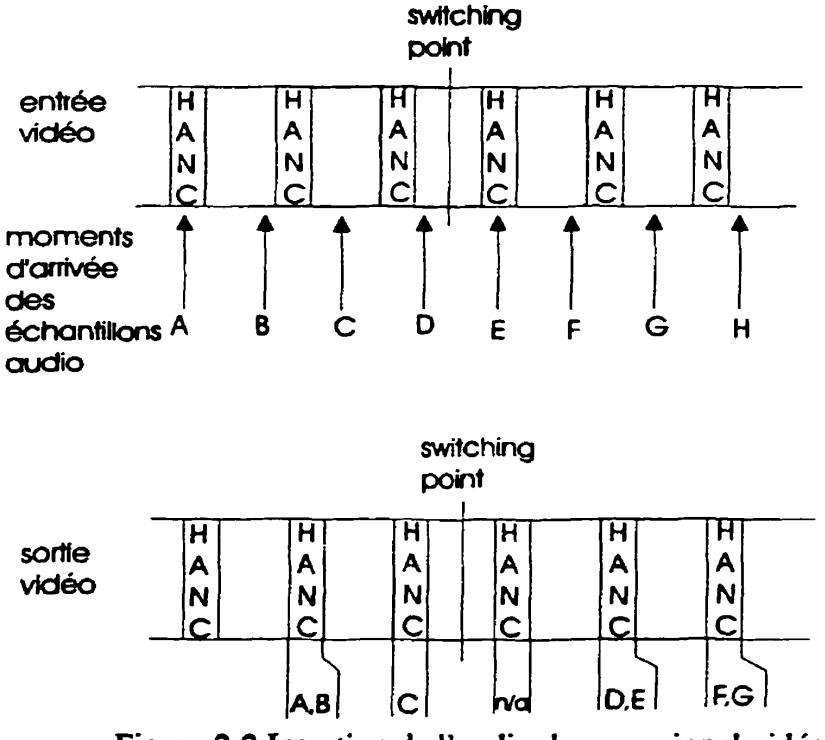


Figure 2.2 Insertion de l'audio dans un signal vidéo HDTV

ne retransmettant pas tous les paquets que le processeur traite si l'usager l'indique (ATC, audio, RS422) et on insère les autres paquets lorsqu'ils sont prêts. En format

HDTV, la façon d'insérer les nouveaux paquets ancillaires audio est quelque peu différente. La figure 2.2 illustre la façon de procéder. Comme on peut voir, on ne peut insérer que 2 paquets ancillaires audio au maximum, à chaque ligne vidéo. De plus, il est interdit d'insérer de l'audio à la ligne suivant le « switching point » (ligne de transition de champ).

Pour les paquets contenant du LTC, on veut pouvoir insérer ce type de paquet à une ligne en particulier et cette ligne doit être programmable. La façon de traiter le RS422 est exactement la même que celle de la version carte. Une fonctionnalité intéressante supplémentaire concernant ces deux lignes sérielles (LTC, RS422) est que, au choix de l'usager, et ce seulement lorsque l'on traite un signal vidéo HDTV, ces deux lignes peuvent traiter de l'audio en formant deux paires AES supplémentaires. Lors de l'insertion, les paquets ancillaires audio fabriqués avec la paire AES conventionnelle sont insérés d'abord et ensuite, c'est le tour des paquets ancillaires audio fabriqués avec les lignes sérielles normalement réservées pour le LTC et le RS422.

Autres sous-modes :

Le traitement des autres sous-modes est presque identique à celui réalisé sur la version carte du processeur. On peut ajouter que le traitement de tones sur les paires AES supplémentaires est également possible dans le cas où on a affaire à un signal vidéo HDTV, dans les sous-modes le permettant.

2.2.3 Caractéristiques du mode DEMUX

Sous-mode normal :

Une fonctionnalité intéressante ajoutée dans ce sous-mode sur la version intégrée est la possibilité de faire un tri des paquets que l'usager a décidé de traiter. En effet, la version précédente ne faisait que retransmettre les paquets qu'elle recevait. Dans la version intégrée, on veut pouvoir extraire tout paquet que l'on juge pertinent du signal vidéo. L'asynchronisme entre les données audio et le signal vidéo est un élément important sur lequel s'attarde le processeur intégré. Une autre fonctionnalité importante est la minimisation des erreurs que pourraient causer un DC erroné d'un paquet ancillaire audio reçu.

Tout comme pour le mode MUX, le processeur, lorsque l'usager le désire, peut traiter deux paires audio AES supplémentaires sur ses lignes sérielles LTC et RS422.

Autres sous-modes :

Comme pour le cas du mode MUX, les autres sous-modes sont pratiquement traités de façon semblable à celle de la version carte du processeur. Encore une fois, il est possible de traiter une seconde paire audio de tones lorsque l'usager le désire, en format vidéo HDTV.

2.2.4 Algorithme général de la version intégrée du processeur à données ancillaires

À titre de synthèse et de rappel de ce chapitre, voici présenté sous forme algorithmique, le traitement effectué par le processeur à données ancillaires intégré. Cet algorithme ne

présente que les grandes lignes du traitement. L'architecture du processeur présentée au chapitre suivant le présentera en détail.

Si on est en mode MUX alors

Si on est en sous-mode colour bar alors

On envoie en sortie les bandes de couleur verticales lorsque l'on se trouve dans la zone de vidéo active. On retransmet l'espace HANC intégralement.

Si on est en sous-mode bypass alors

On retransmet le signal vidéo intégralement, sans le modifier.

Si on est en sous-mode normal alors

Si on est dans la zone HANC alors

On tamponne les premiers mots du signal vidéo d'entrée (74)

On insère dans le signal vidéo le ou les nouveau(x) paquet(s) ancillaire(s) audio (et le paquet ancillaire auxiliaire audio si on traite des échantillons audio de 24 bits en format vidéo DTV) selon qu'il(s) est(sont) prêt(s) et que l'on traite un signal vidéo DTV ou HDTV.

On trie les paquets ancillaires audio selon ce qu'a spécifié l'usager.

On insère les paquets ancillaires formés de LTC et de RS422 lorsque les contraintes sont respectées et que ces paquets sont disponibles.

Si on est dans la zone vidéo active

On procède à la fabrication des différents paquets traités par le processeur. On retransmet le signal vidéo.

Si on est en sous-mode tones alors

On fait la même chose que dans le cas du sous-mode normal sauf qu'on fabrique les paquets ancillaires audio avec des échantillons tones et qu'on ne procède pas au traitement des données LTC.

Si on est en sous-mode tones et colour bar alors

On combine le traitement des sous-modes tones et colour bar.

Si on est en mode DEMUX alors

Si on est en sous-mode tones et colour bar

On transmet les bandes de couleur lorsque l'on est dans la zone de vidéo active.

On retransmet le contenu de l'espace HANC.

On sérialise les échantillons tones vers le monde extérieur et les paquets RS422 du signal vidéo d'entrée (si on ne choisit pas de traiter quatre paires AES audio en format vidéo HDTV).

Si on est en sous-mode normal alors

On sérialise les UDWs des différents paquets ancillaires présents dans le signal vidéo d'entrée.

On fait le trie des paquets ancillaires présents dans le signal vidéo d'entrée selon ce que spécifie l'usager.

Si on est dans la zone vidéo active alors

On traite l'asynchronisme entre l'audio et le vidéo. On retransmet le signal vidéo.

Si on est dans l'espace HANC alors

On traite les erreurs pouvant survenir sur le mot DC des paquets ancillaires audio ainsi que les particularités reliées au mot DBN de ces mêmes paquets.

Si on est en sous-mode bypass alors

On fait la même chose qu'en sous-mode normal sauf qu'on ne procède pas au traitement des données LTC.

CHAPITRE 3

L'ARCHITECTURE DU PROCESSEUR À DONNÉES ANCILLAIRES INTÉGRÉ

Dans ce chapitre, nous allons discuter l'architecture du processeur à données ancillaires. Nous y présentons plusieurs schémas blocs des différents modules de l'architecture, une description des ces modules, ainsi qu'une description des différentes interactions des modules avec le reste de l'architecture.

3.1 L'architecture globale du processeur

La figure 3.1 de la page suivante présente le schéma bloc général du processeur. Un contrôleur principal est la moelle épinière de tout le circuit. À la fois en mode MUX et en mode DEMUX, le signal vidéo entre dans le bloc « Input module » et sort par le module « Output module ». Il est à noter sur cette figure que seule l'EEPROM est à l'extérieur de la puce. Cette dernière sert à programmer les différentes informations nécessaires au bon fonctionnement du circuit. On remarque aussi la présence d'une interface de communication sérielle (port sériel #2) qui dans notre cas est l'I^2C (Inter Integrated Circuit). À l'aide de cette interface, il est possible de communiquer avec le circuit et de modifier certains de ses paramètres de fonctionnement. Le processeur contient quatre blocs de données qui transforment les données reçues par le processeur. En mode MUX, les données arrivent sériellement aux différents blocs que sont le bloc audio1, le bloc audio 2, le bloc LTC/GPS et le bloc RS422. En mode DEMUX, les données sont émises sériellement par ces mêmes quatre blocs. Nous retrouvons également dans l'architecture, des blocs de mémoire RAM (Vidéo RAM, DATA RAM). La « Video RAM » est une mémoire à double port d'adresse. La « Data RAM » est une

mémoire à port d'adresse simple, permettant de ranger les différents paquets ancillaires où les User Data Words que l'on veut traiter.

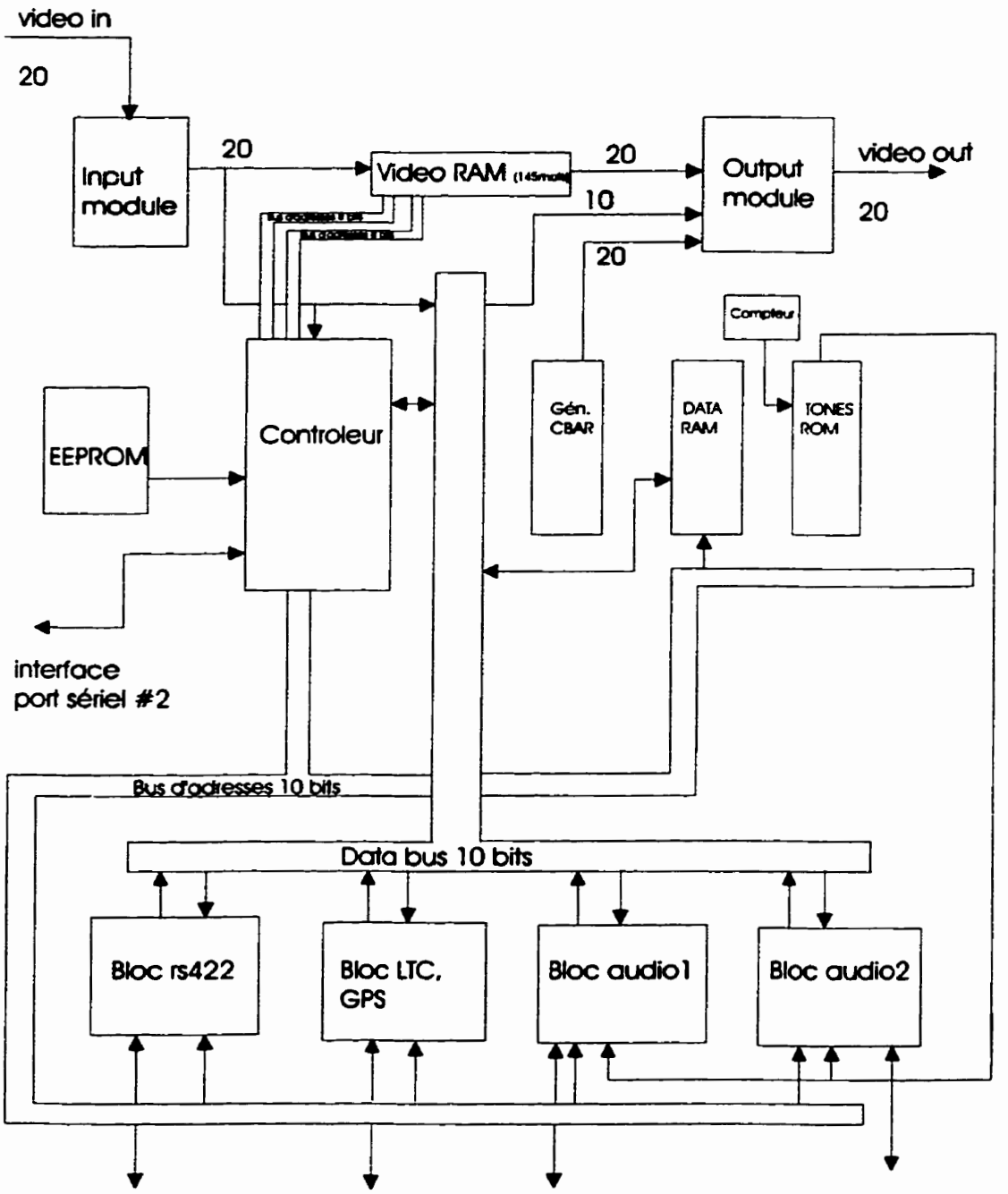


Figure 3.1 Architecture globale du processeur à données ancillaires

Notons finalement la présence de mémoires ROM qui contiennent différentes valeurs d'échantillons audio ou de valeur de chrominance et de luminance nécessaires à certains sous-modes de fonctionnement.

3.2 Le module d'entrée du processeur

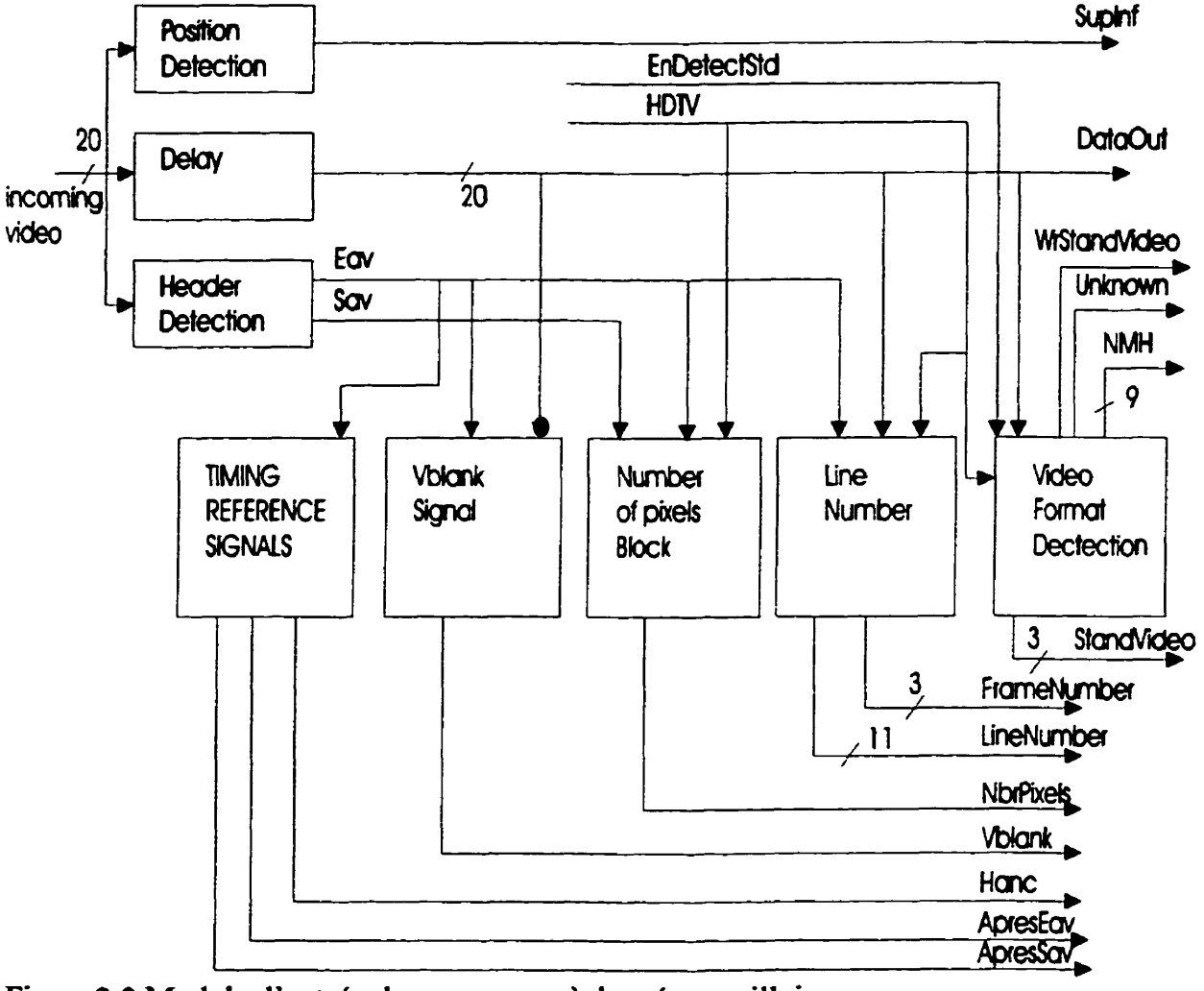


Figure 3.2 Module d'entrée du processeur à données ancillaires

La figure 3.2 montre l'architecture du module d'entrée du circuit. Tout d'abord, le lecteur attentif se souvient que tous les mots de données d'un signal vidéo numérique

sont codés sur 10 bits. Pourquoi alors un bus d'entrée de 20 bits? La réponse à cette question est qu'un signal vidéo HDTV a une fréquence de 1,485 Gbits/s. En utilisant un bus d'entrée de 20 bits, il est possible de traiter le signal vidéo HDTV à une fréquence raisonnable de 74,25 MHz. De plus, il a été prévu que dans l'espace HANC, seuls les 10 LSBs ou les 10 MSBs contiennent les paquets ancillaires. Lorsque l'on a affaire à un signal vidéo numérique DTV, le signal entre seulement dans les 10 LSBs. Le signal « HDTV » de la figure 3.2 indique au processeur si il doit traiter un signal DTV (10 bits à l'entrée) ou un signal HDTV (20 bits à l'entrée).

Le module d'entrée a deux fonctions principales. La première est de déterminer les signaux de référence d'une trame vidéo pour indiquer dans quelle région on se trouve (espace HANC, région d'active picture, espace VBI). La deuxième est de déterminer le format vidéo en présence. La première fonction est réaliser en décodant l'information des séquences EAV et SAV qu'on retrouve à chaque ligne vidéo. En effet, l'information indiquant où on se trouve dans une trame vidéo y est inscrite sur quelques bits. La seconde fonction est réalisée en comptant le nombre de mots dans l'espace HANC. On a vu que dépendamment du standard vidéo en présence, le nombre de mots de l'espace HANC variait. Il s'agit donc de compter ce nombre de mots (disponible sur le signal « NMH » : Nombre de Mots dans l'espace Hanc) et de vérifier s'il est dans un intervalle de mots jugé acceptable pour identifier le standard. Un intervalle d'incertitude de quelques mots est nécessaire au cas où des erreurs s'introduiraient dans certaines lignes de trame vidéo. Cet intervalle est requis pour éviter de juger la présence d'un standard vidéo inconnu lorsqu'une ligne vidéo contient un peu plus ou un peu moins de mots que ce qu'elle devrait contenir. Une fonctionnalité supplémentaire a été ajoutée au processeur permettant de palier à une mauvaise détection de standard. En effet, il est possible à l'usager qui connaît exactement le format vidéo en présence et qui n'a pas été correctement déterminé par le processeur de lui imposer le format. Ceci est fait en désactivant le signal « EnDetectStd » de la figure 3.2 ce qui empêche le module d'entrée d'écrire dans le registre contenant le format vidéo en présence (avec le signal

« WrStandVideo ») les données contenues sur les signaux « StandVideo » et « unknown ». En d'autres mots, la désactivation de ce signal empêche le processeur la détection automatique du standard vidéo, standard devant être imposé par l'usager.

Outre ces fonctions, le module d'entrée détermine le numéro de la ligne en traitement, le nombre de pixels contenus dans la région « active picture », ainsi que le numéro de trame vidéo. Le numéro de ligne est requis pour que le contrôleur puisse procéder correctement (nous verrons plus loin le besoin de ce signal). Mentionnons que pour un signal HDTV, le numéro de la ligne est codé à la suite de la séquence EAV alors qu'on doit le déduire des séquences EAV et SAV d'un signal vidéo DTV. Le numéro de trame n'est pas un paramètre rigoureux. Le module d'entrée ne fait que compter des séquences de 5 trames. Ce paramètre est également nécessaire au contrôleur.
Un délai de 4 mots (block Delay) est nécessaire pour réaliser une autre fonction de ce module. En effet, nous avons préalablement dit que pour un signal vidéo HDTV, les paquets ancillaires se trouvent dans les 10 LSBs ou les 10 MSBs du bus d'entrée de 20 bits. Le sous bloc « Position Detection » détermine dans quelle moitié se trouvent les paquets de données ancillaires et l'indiquent au reste du circuit en assignant correctement le signal « SupInf » (= '1' => 10 MSBs, '0' => 10 LSBs).

3.3 La RAM Vidéo

Pour que le contrôleur puisse correctement « ré-assembler » l'espace HANC en mode MUX, le circuit a besoin d'un élément de retard qui enregistre un certain nombre de mots pour les retransmettre en temps voulu, mais en permettant d'en omettre un certain nombre. C'est pourquoi une mémoire RAM à double port d'adresse (cette RAM a en fait deux ports d'adresse et deux ports de données) a été choisie. Nous devons tout d'abord mentionner ses caractéristiques. Cette mémoire RAM a une capacité de 145 mots de 20 bits et deux ports d'adresse de 8 bits chacun. Nous allons maintenant justifier le besoin de ses caractéristiques. Rappelons tout d'abord que le processeur à

données ancillaires met surtout l'accent sur les données de type audio. Ainsi, chaque nouveau paquet audio prêt à être inséré l'est au tout début de l'espace HANC. La taille maximale de mots audio que l'on insère au début de l'espace HANC est de 70 mots et ce peu importe le format vidéo en présence. Un paquet ancillaire audio fabriqué avec 4 échantillons AES/EBU a une taille de 55 mots dans un format vidéo DTV. De plus, si les échantillons AES/EBU sont codés sur 24 bits, nous rappelons au lecteur que les bits seront assemblés dans un paquet ancillaire audio ET un paquet ancillaire auxiliaire audio. La taille de ces deux paquets mis bout à bout, dans un signal vidéo DTV, avec des échantillons AES/EBU codés sur 24 bits et formés tous les deux de 4 échantillons audio AES/EBU est la taille maximale rencontrée : 70 mots de 10 bits. Notons que pour un signal HDTV, au maximum 2 nouveaux paquets ancillaires audio peuvent être insérés au début de l'espace HANC et que chaque paquet est formé de 31 mots, qui mis bout à bout nécessitent 62 mots. Rappelons également que chaque trame vidéo est composée de deux lignes de transition entre chaque champ d'une trame et qu'il est d'une part interdit d'insérer des paquets ancillaires audio dans ces lignes, ainsi que dans chaque ligne suivant immédiatement la ligne de transition en format vidéo DTV et d'autre part, qu'il est interdit d'insérer des paquets ancillaires audio dans les lignes de transition en format vidéo HDTV.

Ceci dit, observons quelques cas pratiques qui décrivent le besoin de ce type de RAM. La figure 3.3 a) est divisée en deux. La partie i) montre la RAM Vidéo peu de temps après que le processeur ait été démarré. En effet, les 74 premiers coups d'horloge servent à emplir la RAM de 74 mots. La partie ii) démontre ensuite le fait qu'on lit un mot de la RAM Vidéo et qu'on en inscrit un en même temps. Ceci se déroule généralement lorsque le processeur traite la partie « active picture » d'une ligne vidéo, où le processeur ne traite pas les mots de luminance et de chrominance qui la composent. La partie i) de la figure 3.3 b) montre l'état de la RAM Vidéo lorsque le dernier mot de la séquence EAV a été lu de la RAM. À ce moment, la RAM contient toujours 74 mots, mais cette fois ci, ce sont des mots de l'espace HANC que le

processeur doit traiter. Dans cet exemple, supposons que le processeur fabrique des paquets ancillaires audio de groupe 1 avec des échantillons AES/EBU codés sur 24 bits (donc le processeur fabrique également des paquets ancillaires auxiliaires audio de groupe 1) et qu'il doit extraire les paquets ancillaires audio de groupe 1 (auxiliaire audio de groupe 1 également) déjà présents dans le signal vidéo d'entrée. De plus, cet exemple, cela va de soi, est un exemple de DTV.

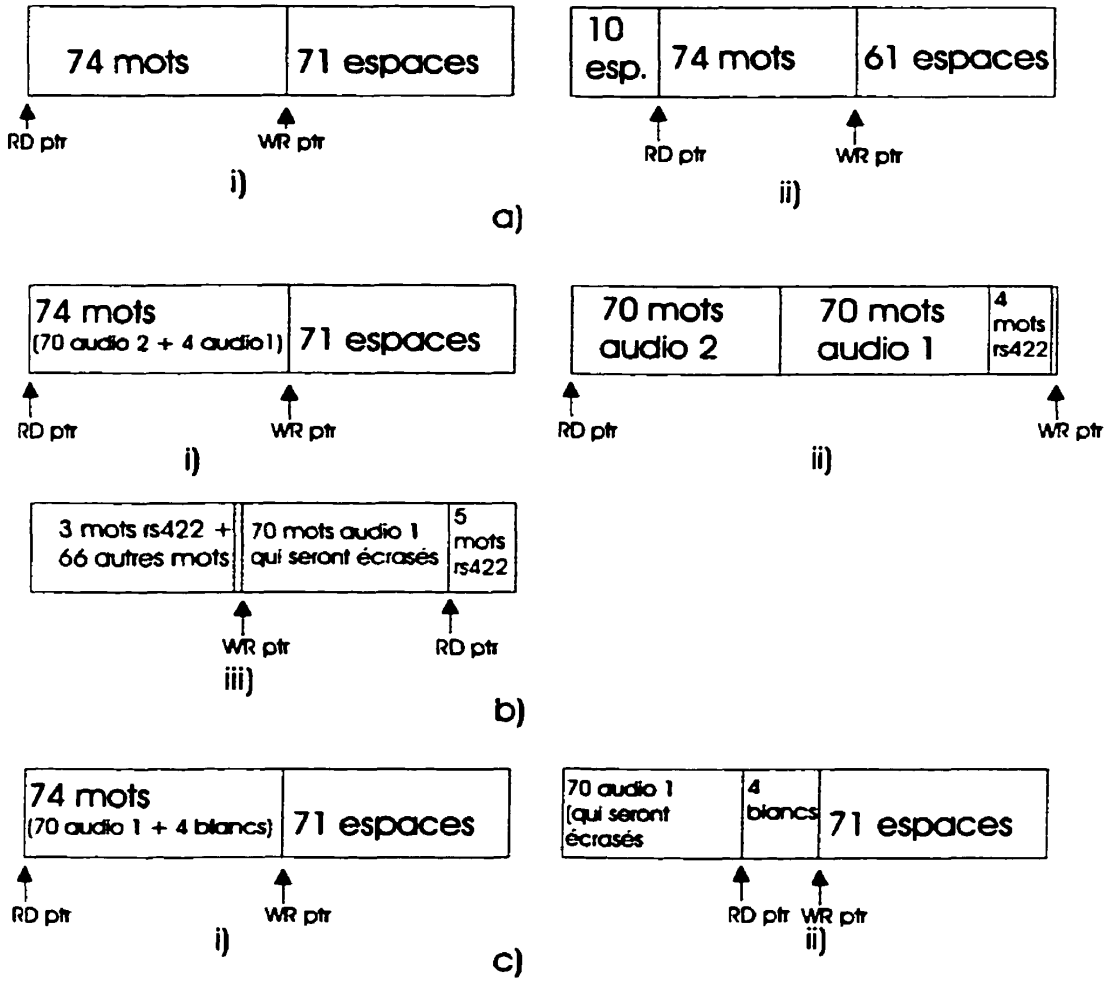


Figure 3.3 Exemple d'utilisation de la RAM Vidéo

Le contenu de la RAM Vidéo a ce moment est un paquet ancillaire audio de groupe 2 (55 mots) suivi d'un paquet ancillaire auxiliaire audio de groupe 2 (15 mots) suivi des 4 premiers mots d'un paquet ancillaire audio de groupe 1. À ce moment précis, le

contrôleur commence à envoyer en sortie, un nouveau paquet ancillaire audio de groupe 1 (55 mots) qui sera suivi d'un paquet ancillaire auxiliaire audio de groupe 1 (15 mots) (cet événement n'est pas représenté sur la figure). Soixante-dix coups d'horloge plus tard, i.e. lorsque le contrôleur a fini d'insérer ces paquets ancillaires audio de groupe 1, la partie ii) de la figure 3.3 b) montre que la RAM Vidéo contient maintenant les 70 même mots des paquets ancillaires audio de groupe 2 définis précédemment, ainsi que les 70 mots des paquets ancillaires audio et auxiliaire audio de groupe 1 (à ne pas retransmettre) et les 4 premiers mots d'un paquet ancillaire RS422. La partie iii) de la figure 3.3 b) montre l'état de la RAM Vidéo 70 coups d'horloge après la partie ii) de cette même figure. À ce moment, les paquets ancillaires audio de groupe 2 ont été lus (retransmis dans le signal vidéo de sortie) et le pointeur de lecture se trouve maintenant au début du prochain paquet ancillaire à retransmettre, i.e., le paquet ancillaire RS422. Notons que pendant ces 70 coups d'horloge, le reste du paquet ancillaire RS422 a eu le temps d'être inscrit (longueur totale : 8 mots) et que d'autres mots divers (66), présents dans l'espace HANC du signal vidéo d'entrée, ont également été inscrits. Ceci démontre que les paquets ancillaires audio de groupe 1 n'ont pas été retransmis et seront écrasés dans la RAM Vidéo.

La figure 3.3 c) démontre une situation indésirable, présente dans le signal vidéo d'entrée et qui est corrigée par le processeur. En effet, supposons que l'on se trouve dans une ligne de transition où il n'est pas permis d'insérer des données ancillaires et que la RAM Vidéo contienne 70 mots de paquets ancillaires audio de groupe 1 (55 pour audio ancillaire et 15 pour audio ancillaire auxiliaire) ainsi que 4 autres mots divers (blancs) (partie i) de la figure 3.3 c)). À ce moment, qui est immédiatement après que le dernier mot de la séquence EAV ait été retransmis en sortie (nous sommes donc au début de l'espace HANC), le processeur peut positionner le pointeur de lecture de la RAM à la suite de ce paquet et ainsi, ne pas le retransmettre comme la règle le veut. Ceci est représenté sur la partie ii) de la figure 3.3 c). Nous devons mentionner ici une limite du système lors du traitement des lignes de transition d'une trame vidéo. Bien que la figure

3.3 c) démontre une possibilité de correction du signal vidéo, cette situation le sera seulement si le groupe audio sélectionné par l'usager (et donc que le processeur doit traiter) est le même que le ou les paquet(s) retrouvé(s) dans la RAM Vidéo. Ainsi, un paquet ancillaire audio de groupe 2 (55 mots) suivi d'un paquet ancillaire auxiliaire audio de groupe 2 (15 mots), alors que le groupe audio sélectionné par l'usager est le groupe audio 1 fait en sorte que ces deux paquets se retrouveront dans le signal vidéo de sortie du processeur, bien que contraire à la règle. Rappelons ici au lecteur qu'il est interdit d'insérer sur une ligne de transition de champ ou la ligne qui la suit immédiatement (cas DTV), des paquets audio de quelque groupe que ce soit. Mentionnons que ceci n'est pas très grave et que pour solutionner le problème, il aurait fallu accumuler dans une Vidéo RAM, une quantité de mots beaucoup plus grande que les 74 de la présente façon de fonctionner. Il est nécessaire ici de rappeler quelques faits au lecteur. Pour un groupe audio traité avec des échantillons codés sur 24 bits, il y aura un maximum de 4 échantillons audio qui seront encodés en un paquet audio ancillaire (55 mots) et un paquet ancillaire auxiliaire audio (15 mots), totalisant 70 mots. Puisqu'il y a quatre groupes audio, ces quatre groupes audio pourraient être encodés selon la situation décrite dans la phrase précédente et pourraient tous se retrouver sur une ligne de transition de champ, ce qui fait qu'il faudrait extraire 4*(70)=280 mots de l'espace HANC.

En résumé, on doit accumuler 74 mots dans la RAM Vidéo lorsque l'on met le processeur en marche pour prévenir la situation décrite sur la figure 3.3 c) (situation dans laquelle on a choisi de « réparer » pour un seul groupe audio et non quatre). Cette situation est celle où on se trouve dans une ligne de transition de champ dans une trame vidéo DTV, qu'il y a un paquet ancillaire audio de 55 mots suivi d'un paquet ancillaire auxiliaire audio de 15 mots dans l'espace HANC du signal de vidéo d'entrée, les deux ayant le même groupe audio que celui spécifié par l'usager, de présents. Les 4 mots supplémentaires sont nécessaires pour connaître ce qui suit ces 70 mots audio dans l'espace HANC (000h, 3FFh, 3FFh, DID pour tout paquet ancillaire). Rappelons que la

règle veut qu'on ne doive pas insérer de paquet ancillaire audio dans les lignes de transition, ce qui fait que le processeur doit extraire ces 70 mots audio du signal vidéo. Ces mots sont perdus, i.e. ils ne seront jamais retransmis.

La taille de 145 mots de la RAM Vidéo est justifiée par le fait qu'on doit accumuler tout d'abord 74 mots comme décrit ci-haut. On doit ajouter 70 mots à cette taille en raison de la situation décrite dans la figure 3.3 b). Cette situation est la suivante, après avoir accumulé les 74 premiers mots dans la RAM Vidéo, on retransmet éventuellement le dernier mot de la séquence EAV d'une ligne vidéo autre qu'une ligne de transition de champ (ou de la ligne suivant une ligne de transition de champ lorsqu'en présence d'un signal de vidéo DTV). À la suite de la séquence EAV, le processeur insère un nouveau paquet ancillaire audio de 55 mots suivi d'un nouveau paquet ancillaire auxiliaire audio de 15 mots. Pendant ce temps, 70 nouveaux mots doivent être inscrits dans la RAM Vidéo (70 + 74 = 144), sans qu'on puisse en lire puisque la source des nouveaux paquets est autre que la RAM Vidéo. Enfin un mot supplémentaire est nécessaire pour éviter que les pointeurs de lecture et d'écriture pointent sur une même case mémoire de la RAM Vidéo. Nous aurons l'occasion plus tard de revenir sur la RAM Vidéo.

Cette partie décrivant la RAM vidéo met en lumière une première contribution du projet au sujet de l'étude de la latence d'un circuit intégré. A ce stade de l'ouvrage, le lecteur commence à avoir une bonne vision de l'architecture du processeur à données ancillaires bien qu'il lui manque encore plusieurs détails. En fait, hormis une petite latence dans le module d'entrée, les 74 mots accumulés dans la RAM vidéo, lorsque l'on met le processeur en marche, constituent l'essentiel de la latence de traitement du processeur. Dans certains circuits intégrés ou parties de circuits intégrés comme des multiplicateurs, on essaie d'abaisser la latence de traitement le plus possible. Dans d'autres cas comme celui qui nous intéresse présentement, la latence de traitement est directement reliée à la fonctionnalité qu'on désire implanter dans un circuit intégré. Ainsi, la latence de 74 mots est basée sur un compromis entre la fonctionnalité à réaliser (basée sur un ou des

standard(s) de données) et la rapidité du traitement. Rappelons ici que les 74 mots viennent du calcul suivant. Quatre échantillons audio codés sur 24 bits en format DTV forment un paquet ancillaire audio de 55 mots et un paquet ancillaire auxiliaire audio de 15 mots, selon le standard SMPTE 272M. Les quatre autres mots viennent de l'entête (000h, 3FFh, 3FFh) et du mot DID d'un paquet ancillaire. Dans notre cas, on veut pouvoir retirer 70 mots d'un signal vidéo sans créer de trous dans l'espace HANC, tout en connaissant le contenu du paquet ancillaire suivant ces 74 mots à retirer.

Cette étude de latence met en lumière le plus petit délai de traitement possible, considérant la fonctionnalité à réaliser. Ce n'est que parce que nous avons jugé inacceptable d'insérer des échantillons (paquets) audio dans les lignes de transition de champs que nous avons défini une latence de 74 mots. Nous aurions pu tout aussi bien raccourcir ou allonger cette latence tenant compte d'une autre stratégie permettant de faire moins lorsqu'elle est raccourcie et plus en l'allongeant.

Dans bien des cas en matière de circuits intégrés, la latence de traitement maximale est le résultat d'un compromis entre le délai de traitement acceptable (une mesure de vitesse de traitement) et la fonctionnalité à implanter dans ce circuit.

3.4 La RAM de données

Cette mémoire est une RAM à un port d'adresse et 1k x 10 bits de capacité. En mode MUX, son rôle est de contenir les différents paquets ancillaires fabriqués par le processeur, paquets qui seront éventuellement insérés dans le signal vidéo. L'espace mémoire est divisé ainsi : 262 mots réservés aux paquets RS422, 262 mots réservés aux paquets LTC, 500 mots réservés aux paquets ancillaires audio et ancillaires auxiliaires audio. Bien que les paquets RS422 soient définis sur 8 mots et les paquets LTC sur 23 mots, le processeur a été conçu de manière à assurer une certaine flexibilité pour le traitement de certains types de données partiellement définis ou non définis à cette date.

Pour les paquets RS422 et LTC, une taille de bloc mémoire de 262 mots se justifie en rappelant que chaque paquet ancillaire peut avoir au maximum 255 UDWs. En ajoutant les six mots toujours présents au début d'un paquet ancillaire (000h, 3FFh, 3FFh, DID, DBN, DC) ainsi que le mot CS terminant tout paquet ancillaire, on en vient à une taille de bloc de 255 + 6 + 1 = 262 mots. De plus, il est relativement facile de dire qu'un processeur à données ancillaires spécialisé dans le traitement des données audio, insérera des paquets de grande taille uniquement dans des lignes spécifiques, telles que les lignes de transition de champ pour ne pas empiéter sur l'espace des paquets ancillaires audio qui normalement sont insérés à chaque ligne vidéo (excepté les lignes de transition de champ et celles qui les suivent immédiatement) (rappelons ici au lecteur que dépendamment du format vidéo, certains espaces HANC n'ont que 268 ou 280 mots; si on insère un paquet de 262 mots dans une des lignes du signal vidéo, il ne reste plus de place, ou presque, pour d'autres types de paquets ancillaires). Ces lignes se rencontrent deux fois par trame vidéo et il est donc évident que le processeur ne conservera pas plusieurs paquets de 262 mots de longueur d'un même type de données en mémoire (dans la RAM de données) puisqu'il a amplement le temps d'en fabriquer un entre deux insertions. Il est aussi évident que la réception de ce type de données se fera à base fréquence d'horloge.

Pour ce qui est de la taille des blocs audio, mentionnons que 7 paquets ancillaires audio (55 mots) et 7 paquets ancillaires auxiliaires audio (15 mots) peuvent être contenus dans un bloc mémoire de 500 mots de 10 bits. Un autre rappel est ici nécessaire. Le processeur traite quatre standards DTV et quelques standards HDTV. Selon le tableau 1.1, le temps de ligne des différents standards (nombre de mots sur un ligne * période de l'horloge vidéo) est d'environ 64us. Un échantillon audio de fréquence (d'échantillonnage) de 48 kHz, qu'il soit encodé en 20 ou 24 bits, prends 21us (1/48000 Hz) pour arriver au processeur. Selon le format vidéo en présence, il arrive environ 3,05 échantillons par ligne (64us/21us). Environ un paquet sur vingt est fabriqué avec quatre échantillons audio pour évacuer le 0,05 échantillon qui s'accumule lorsque l'on fabrique

les paquets audio avec trois échantillons (3,05 – 3 = 0,05). Puisqu'on reçoit 3,05 échantillons par ligne et qu'on insère 19/20 trois échantillons par ligne et 1/20 quatre échantillons par ligne, on voit qu'il n'y a aucune accumulation d'échantillons audio. La taille du bloc de mémoire audio est donc amplement suffisante (une bonne marge de sécurité est assurée). Nous n'entrerons pas dans les détails ici, mais en raison de l'interdiction d'insérer des paquets audio sur les lignes de transition de champ et les lignes suivant immédiatement ces dernières (DTV), on doit conserver quelques fois jusqu'à 10 échantillons audio en mémoire. Ces échantillons occupent une taille d'environ 180 mots de 10 bits, encodés en format ancillaire. Le lecteur attentif comprendra que la proportion de paquets insérés et fabriqués avec trois échantillons en rapport avec celle des paquets insérés et fabriqués avec quatre échantillons n'est pas exactement de 19/20. En effet, avec les lignes de transition de champ où aucun paquet audio n'est inséré (alors que les échantillons audio continuent d'arriver au processeur), il faut fabriquer et insérer un peu plus souvent des paquets audio fait avec quatre échantillons.

À ce stade ci de la présentation, le lecteur n'a pas encore la vue d'ensemble de l'architecture. Nous pouvons quand même dire (voir figure 3.1) que les divers mots formant les paquets ancillaires sont fabriqués dans plusieurs unités fonctionnelles du processeur (blocs audio, LTC, RS422, contrôleur, etc.). Une raison supplémentaire pour justifier la nécessité d'un tel bloc de RAM de données est que les paquets ancillaires sont assemblés au complet dans la RAM (sauf le mot CS). Ceci fait en sorte que le réseau logique nécessaire au contrôleur pour aller lire les paquets complétés dans la RAM (et donc les insérer dans le signal vidéo de sortie) est moins complexe que celui qui serait nécessaire pour insérer tout nouveau paquet ancillaire dans le vidéo de sortie, à partir de plusieurs unités fonctionnelles réparties dans l'architecture.

En mode DEMUX, la RAM contient les User Data Words de tout paquet ancillaire que le processeur doit traiter. Encore une fois, des tailles de bloc de 255 mots pour les

données RS422, 255 mots pour les données LTC, 257 mots pour les mots de données de la ligne sérielle audio #1 et 257 mots pour les mots de données de la ligne sérielle audio #2 s'expliquent sensiblement pour les mêmes raisons qu'en mode MUX. Nous verrons plus loin dans ce chapitre que la RAM de données aide à traiter l'asynchronisme des données audio par rapport à la fréquence vidéo, toujours dans le mode DEMUX.

3.5 Les ROM de tonalité (tones) et de bande de couleur (colour bar)

La ROM de tonalité et le générateur de bandes de couleur sont nécessaires lorsque le processeur traite les sous modes de fonctionnement « tones, colour bar et tones & colour bar ». En effet, la ROM contient les différents échantillons audio AES/EBU qui forment un signal sonore de 1 kHz de fréquence (12 échantillons audio à 48 kHz pour chaque quart de période d'un signal de 1 kHz) nécessaires à la fabrication et l'insertion de paquets ancillaires audio dans le signal vidéo de sortie en mode MUX ou la transmission de ces échantillons de manière sérielle vers le monde extérieur en mode DEMUX. Le générateur de bandes de couleur quant à lui, contient les différentes valeurs de chrominance et de luminance nécessaires à diviser la région active picture en 8 bandes de couleur verticales.

3.6 Le module de sortie du processeur

La figure 3.4 montre le schéma bloc de ce module. Le module de sortie a deux fonctions : multiplexer les différentes sources de données de manière à produire un signal vidéo de sortie cohérent; calculer et insérer le mot CS (Check Sum) de tout paquet ancillaire. Le module de sortie a également la fonction de calculer et insérer les mots ECC (Error Correcting Codes) qui font partie de tout paquet ancillaire audio à être inséré dans un signal vidéo HDTV. Comme l'indique la figure, le signal vidéo émis en sortie du processeur doit passer par deux multiplexeurs. Le premier laisse passer une entrée en provenance de la RAM de donnés (FromDataRAM) ou une entrée en provenance de la

RAM Vidéo (FromVideoRAM) ou une entrée en provenance de la ROM contenant les valeurs de bandes de couleur (FromColorBar). Le deuxième laisse passer en sortie soit l'entrée provenant du premier multiplexeur, soit les mots CS ou ECC calculés par le module, soit une entrée de « blackvideo ».

L'entrée « FromDataRAM » du premier multiplexeur est sélectionnée à chaque fois que le processeur veut insérer des nouveaux paquets ancillaires dans le signal vidéo. Pour ce faire, le deuxième multiplexeur doit laisser passer l'entrée venant du premier multiplexeur. Comme mentionné précédemment, les nouveaux paquets ancillaires sont assemblés au complet dans la RAM de données sauf pour les mots CS et ECC. Ainsi, lorsque les nouveaux paquets ancillaires sont insérés, le contrôleur (machine à états) du module de sortie doit lire la sortie du premier multiplexeur pour savoir quand actionner ses mécanismes de calcul de CS et d'ECC si nécessaire. Puis, après le dernier UDW du paquet ancillaire nouvellement inséré, le module insère le mot CS qu'il vient de calculer précédé si nécessaire des mots ECC également frais calculés.

L'entrée « FromVideoRAM » du premier multiplexeur est sélectionnée lorsque le processeur veut retransmettre des mots présents dans le signal vidéo d'entrée. Ces groupes de mots peuvent être des paquets ancillaires si on se trouve dans l'espace HANC ou des valeurs de chrominance et de luminance si on se trouve dans la région active picture. La troisième sortie du premier multiplexeur (FromColorBar) est sélectionnée uniquement quand le processeur est dans le sous mode « colour bar » ou « tones & colour bar » et qu'on se trouve dans la région active picture.

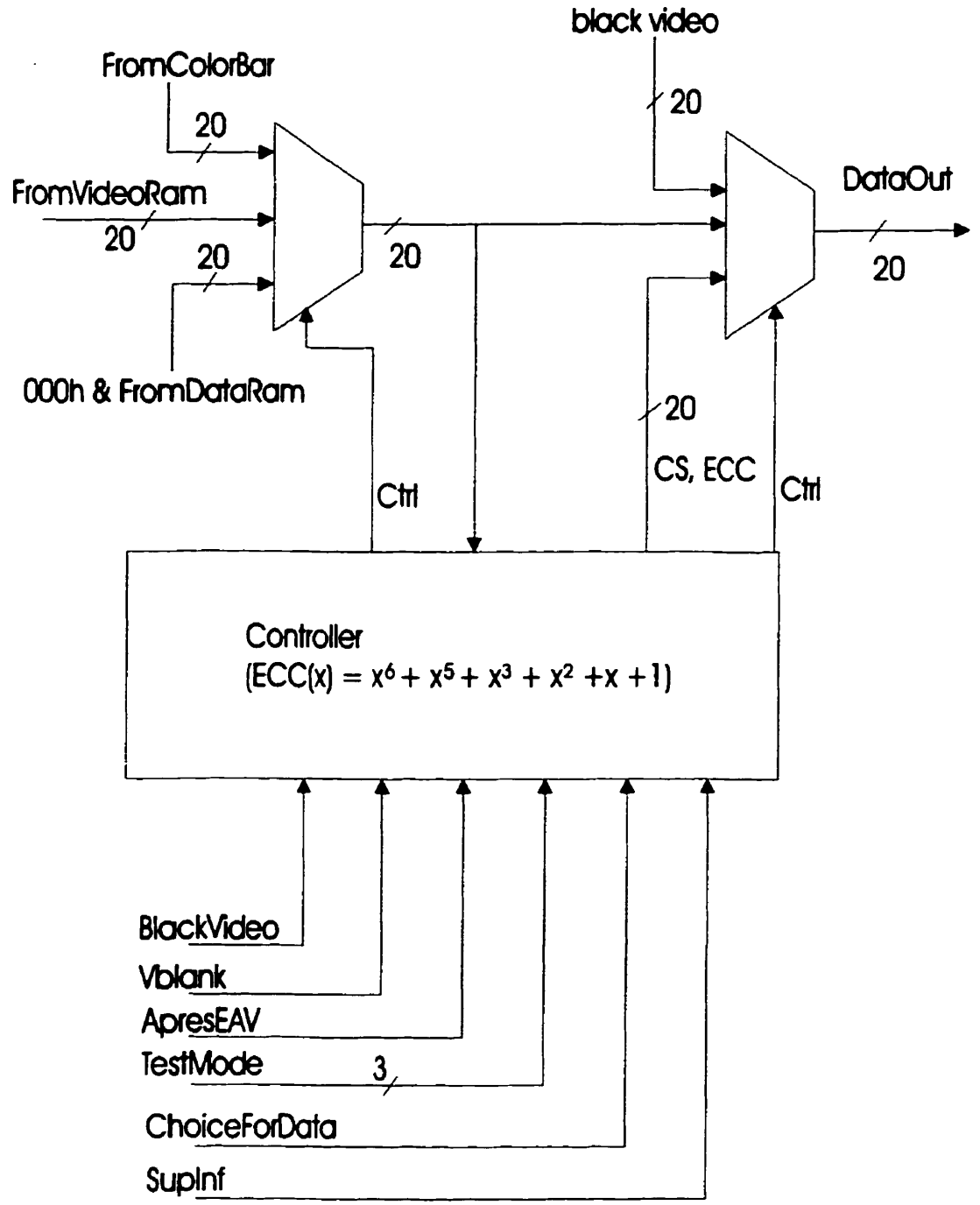


Figure 3.4 Module de sortie du processeur

Pour bien gérer les différents choix qui s'offrent, le module de sortie reçoit plusieurs signaux en provenance du contrôleur. Ainsi, les signaux « testmode », « ChoiceForData », « BlackVideo », « ApresEAV », « SupInf » et « Vblank » sont autant d'indices renseignant le module sur ce qu'il doit faire. De plus, le module lit le contenu qui sort du premier multiplexeur et est donc capable de se synchroniser sur les séquences EAV et SAV pour savoir quand laisser passer les bandes de couleur verticales en provenance du générateur Colour Bar, lorsque dans le bon sous mode. Notons que bien que le signal de sortie du premier multiplexeur entre dans le contrôleur du module de sortie, aucun cycle de latence supplémentaire n'est inséré.

L'entrée du deuxième multiplexeur provenant du premier est retransmise à chaque fois que les signaux « BlackVideo » et « ChoiceForData » sont à '0'. L'entrée du deuxième multiplexeur et qui vient du contrôleur du module de sortie est transmise en sortie quand il est temps pour le module d'insérer le mot CS (précédé des mots ECC lorsque nécessaire) lorsque le signal « ChoiceForData » est à '1'. En effet, cela indique qu'un nouveau paquet ancillaire va être inséré et que le module de sortie doit calculer le mot CS pour ensuite l'insérer au bon moment. La troisième entrée du deuxième multiplexeur (blackvideo) est transmise lorsque le signal « BlackVideo » est à '1'. Cela se fait lorsque l'espace HANC n'est pas rempli de paquets ancillaires. Dans ce cas, le processeur emplit le reste de l'espace de ce qui est appelé du « black vidéo » et qui n'est autre que les valeurs de chrominance et de luminance de la couleur noire.

3.7 La machine bit-sérielle

Nous allons maintenant passer à un module de très grande importance. En fait, c'est le deuxième module en importance dans l'architecture du processeur à données ancillaires, le premier étant le contrôleur global. La figure 3.1 nous montrait quatre blocs de données. Un bloc peut être vu de la façon suivante.

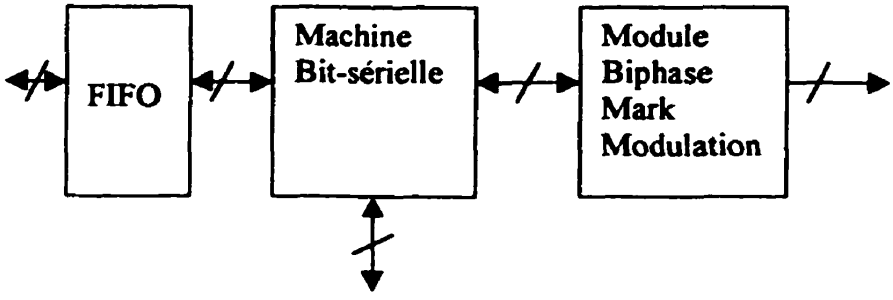


Figure 3.5 Structure d'un bloc de données

La machine bit-sérielle est le module principal d'un bloc de traitement de données. Il interagit avec un FIFO, un module dit module Biphase Mark Modulation (seulement actif en mode DEMUX) et le monde extérieur. La fonction de ce type de bloc est de transformer les données du format ancillaire au format sériel de chaque type de données et vice versa. Par exemple, un bloc de traitement de données audio recevra des échantillons audio AES/EBU sériellement et les transformera en format ancillaire (UDWs) dans le mode MUX.

3.7.1 Les besoins d'une machine bit-sérielle

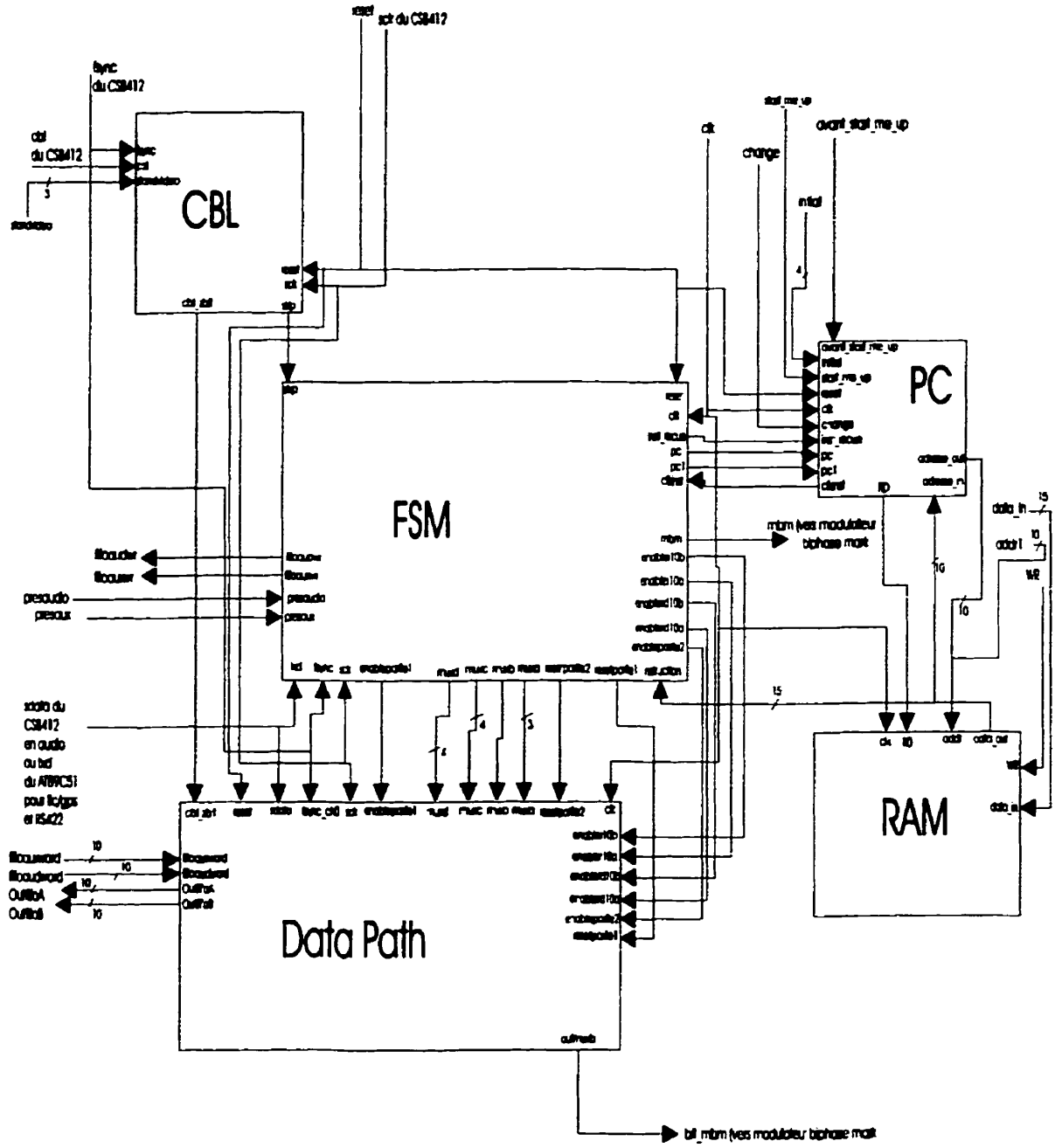


Figure 3.6 Schéma bloc de la machine bit-sérielle

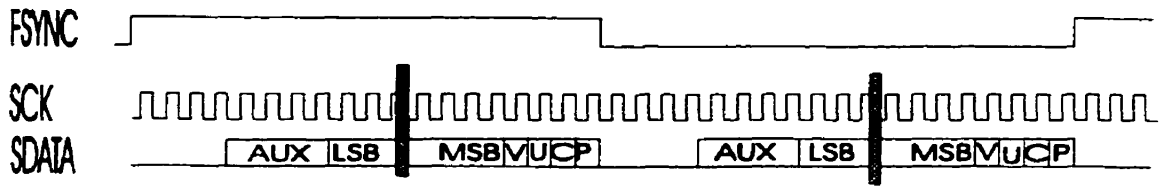
La figure 3.6 montre l'architecture de la machine bit-sérielle. Elle est composée de cinq parties. La machine à états finis (FSM = Finite State Machine) est le contrôleur de la machine. Elle interagit avec toutes les autres parties en gérant plusieurs signaux d'entrée et de sortie. Une mémoire RAM fait partie de chaque machine bit-sérielle. Elle contient des instructions qui devront être exécutées par la FSM. Le chemin de données (data path) est la partie de la machine sur laquelle la machine traite ses instructions sur les données. Enfin, la partie CBL est un bloc spécialement conçu pour interagir entre la FSM et la puce CS8412 (audio receiver), externe au processeur et dont la fonction est de transmettre les bits de donnée audio au processeur.

Le besoin d'une telle architecture pour traiter les données sérielles vient du fait que le traitement requis sur les données de différents types est somme toute assez simple. De plus, une certaine flexibilité est appréciée du fait que le processeur est réalisé dans un ASIC. En effet, il est impossible après qu'il soit fabriqué par une fonderie, de le modifier. Une méthode pour le rendre flexible est de le concevoir avec des blocs de mémoire RAM insérés à des endroits stratégiques de l'architecture. L'un de ces endroits est la machine bit-sérielle. En effet, la RAM contient les instructions à être exécutées par le processeur sur les données sérielles pour les transformer en format ancillaire et vice versa. Une certaine flexibilité est requise dans le séquencement et la répétition des instructions à opérer en fonction du type de donnée traité. De plus, il est souhaitable que le processeur puisse traiter des types de données non définis à ce jour dans les normes existantes. Pour ces raisons, ce type d'architecture qui est un genre de machine micro-programmée, est très utile.

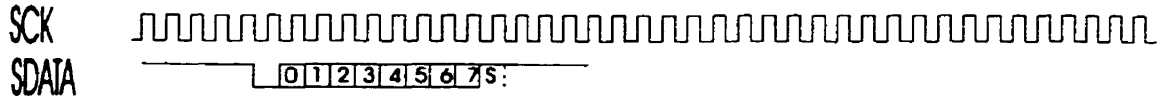
3.7.2 Les modes de réception et de transmission sérielle

Mentionnons tout d'abord que lorsque le processeur opère en mode MUX, il reçoit les données audio d'une puce externe : le CS8412 (audio receiver). Tous les autres types de données, toujours en mode MUX, arrivent par un micro-contrôleur du type AT89C51.

La figure 3.7 illustre le processus de réception des données audio en provenance du CS8412 et de l'AT89C51.



a)



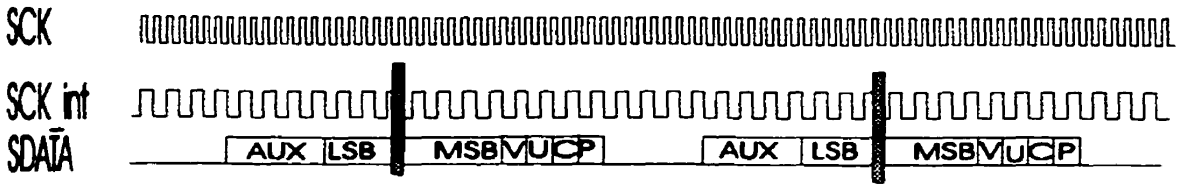
b)

Figure 3.7 Diagrammes temporels de transmission des puces CS8412 et AT89C51

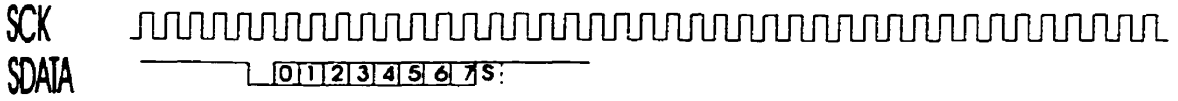
La figure 3.7 a) montre les trois signaux fournis au processeur par le CS8412. Ce dernier, rappelons le, transmet les données audio en format AES/EBU au processeur. Le signal « FSYNC » est une horloge à 48 kHz et indique le début d'une sous-trame audio à chacun de ses fronts. Le signal « SCK » est l'horloge d'échantillonnage des bits audio. Le processeur lit ces bits au front montant de « SCK » alors que le CS8412 les fait varier à chaque front descendant de cette même horloge. Cette horloge « SCK » a une fréquence de 3,072 MHz. Finalement, le signal « SDATA » contient l'information audio, c'est-à-dire, les bits. Sur la figure 3.7 b), on voit la manière par laquelle sont transmises au processeur les données autres que les données audio. Dans cet exemple, un micro-contrôleur externe au processeur (AT89C51) lui envoie deux signaux. Le

signal « SCK » est l'horloge d'échantillonnage des bits transmis. L'horloge est transmise au processeur qui lit les bits au front montant de l'horloge. Le signal « SDATA » contient les bits de données. Le protocole de communication sur ce signal est le suivant. La communication débute par un « start bit » qui est une transition de '1' à '0' de la ligne « SDATA ». Vient ensuite un octet d'information, i.e. 8 bits de données. Le tout se conclut avec un « stop bit » qui est toujours à '1'. La transmission de données ne se fait pas de façon continue d'où le besoin d'un signal « start bit ». De plus, on veut offrir à l'usager le choix de la vitesse de transmission des données d'où la nécessité d'utiliser un signal d'horloge également

Dans le mode DEMUX, le processeur à données ancillaires transmet les données sériellement au monde extérieur. La figure 3.8 a) montre la façon de transmettre les données audio en format AES/EBU. Le signal « SCK_int » n'est pas transmis hors du puce, c'est un signal interne. Sa fréquence est toujours de 3,072 MHz et c'est l'horloge des bits AES/EBU. Par contre, l'horloge « SCK » dont la fréquence est de 6,144 MHz est transmise hors du processeur. Cette horloge est synthétisée à partir de l'horloge vidéo selon une méthode de synthèse directe (DDS) (de l'article Jitter Model of Direct Digital Synthesis Clock Generators, Dorin Emil Calbaza, Yvon Savaria, paru dans Proceedings on Intrernational Symposium on Circuit and Systems) dont nous ne discuterons pas dans ce mémoire. Nous avons besoin de cette horloge, car la façon de transmettre des données audio AES/EBU est de le faire selon une modulation appelée « biphase mark ». Nous ne décrirons pas cette méthode, mais il convient de dire que chaque bit de donnée audio sera codé sur deux bits en biphase mark modulation, d'où le besoin de l'horloge à 6,144 MHz. Dans cette figure, le processeur transmet les données sur le front descendant de l'horloge « SCK ». Le lecteur désirant en savoir plus sur la modulation « biphase mark » des données audio peut consulter le standard AES3-1992.



a)



b)

Figure 3.8 Diagramme de transmission des données par le processeur à données ancillaires

La figure 3.8 b) présente la façon de transmettre les données autres que les données audio. Le protocole de transmission est le même sur la ligne « SDATA » qui cette fois est configurée en sortie par le processeur. L'horloge « SCK » doit cependant être fournie au processeur. Encore une fois, le processeur transmet les bits sur le front descendant de l'horloge.

3.7.3 Le chemin de données (data path) de la machine bit-sérielle

La figure 3.9 illustre le chemin de données de chaque machine bit-sérielle présente dans chacun des quatre blocs de données du processeur. Comme on peut voir, il est relativement simple. Il est constitué de deux blocs pouvant calculer des parités (bascule D, ou-exclusif, multiplexeur 2 à 1), de deux registres à décalage de 10 bits chacun, d'un inverseur, de trois registres 1 bit (bascule D) mémorisant respectivement les valeurs de SDATA (bit d'entrée), CBL (bit provenant du bloc CBL de la machine bit-sérielle) et de la valeur de FSYNC à chaque coup d'horloge de SCK. De plus, quelques autres multiplexeurs sont requis pour compléter le chemin de données (data path). Tous les signaux de contrôle tels les « enable » de parité, les signaux de sélection des multiplexeurs et les « enable » d'écriture des registres à décalage 10 bits sont transmis par le contrôleur de la machine bit-sérielle. Ces signaux de contrôle ainsi que les registres à décalage de 10 bits fonctionnent à la fréquence vidéo (27 MHz, 36 MHz ou 74,25 MHz).

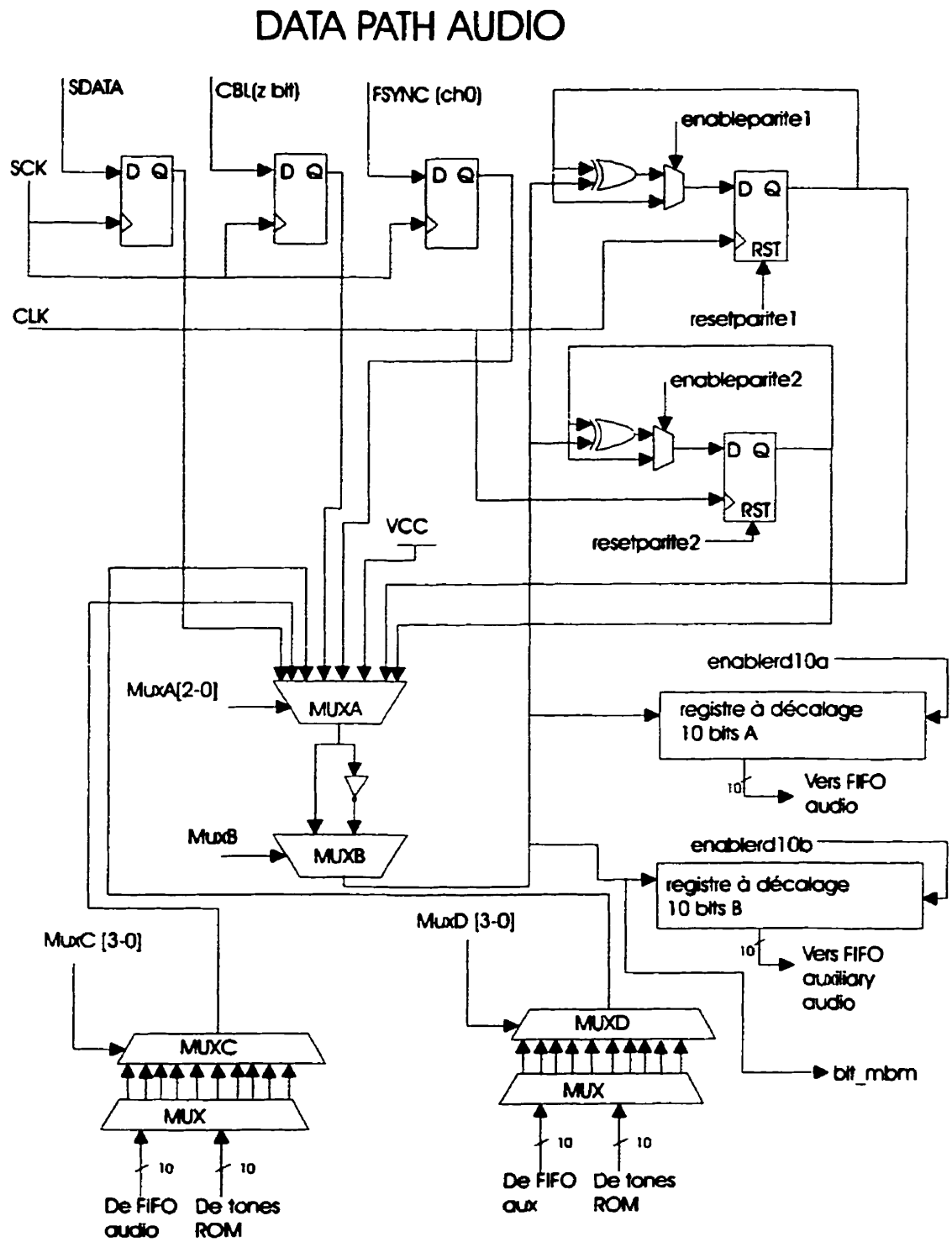


Figure 3.9 Chemin de données de la machine bit-sérielle

3.7.4 La RAM

La mémoire RAM associée à chaque machine bit-sérielle est une mémoire à un port d'adresse et une capacité de 256 mots de 16 bits. Elle contient les instructions à être effectuées par la FSM sur le data path.

3.7.5 Le Compteur ordinal (Program Counter)

La figure 3.10 montre le compteur ordinal en relation avec la mémoire RAM et quelques signaux. Ce compteur ordinal n'est pas un compteur ordinal classique. La façon dont il a été conçu est la suivante. Tout d'abord, nous avons décidé que la RAM pourrait contenir jusqu'à 16 programmes différents, programmes de traitement des données de quelque type que ce soit. L'adresse de départ de chacun des 16 programmes en question est inscrite dans les 16 premières cases de la RAM. Bien sur, ce ne sont que les 8 LSBs du mot de 16 bits qui contiennent l'adresse en question. C'est pourquoi on peut voir que le signal « instruction », en plus d'aller à la FSM, va également vers le compteur ordinal.

La façon de pointer vers le bon programme à effectuer se divise en deux étapes. La première est qu'un des modules du contrôleur global de la puce envoie sur le signal «initial » la valeur binaire encodée sur 4 bits du numéro du programme que la machine bit-sérielle devra effectuer. Le compteur ordinal reçoit ce signal, forme une adresse sur huit bits et va pointer sur la case mémoire désignée par le signal « initial ». Pour clarifier le tout, allons-y d'un petit exemple. Supposons que l'ADP veuille exécuter le programme numéro 3 de la RAM de la machine bit sérielle du bloc audio 1. Le contrôleur envoie alors sur « initial » la valeur « 0011 ». Quand le compteur ordinal reçoit le feu vert pour commencer le travail, il va lire le contenu de l'adresse « 0000&0011 ». Il reçoit et mémorise le résultat disponible sur le signal « instruction »

qui indique l'adresse de départ du programme numéro 3. C'est le seul moment où l'information transmise sur le

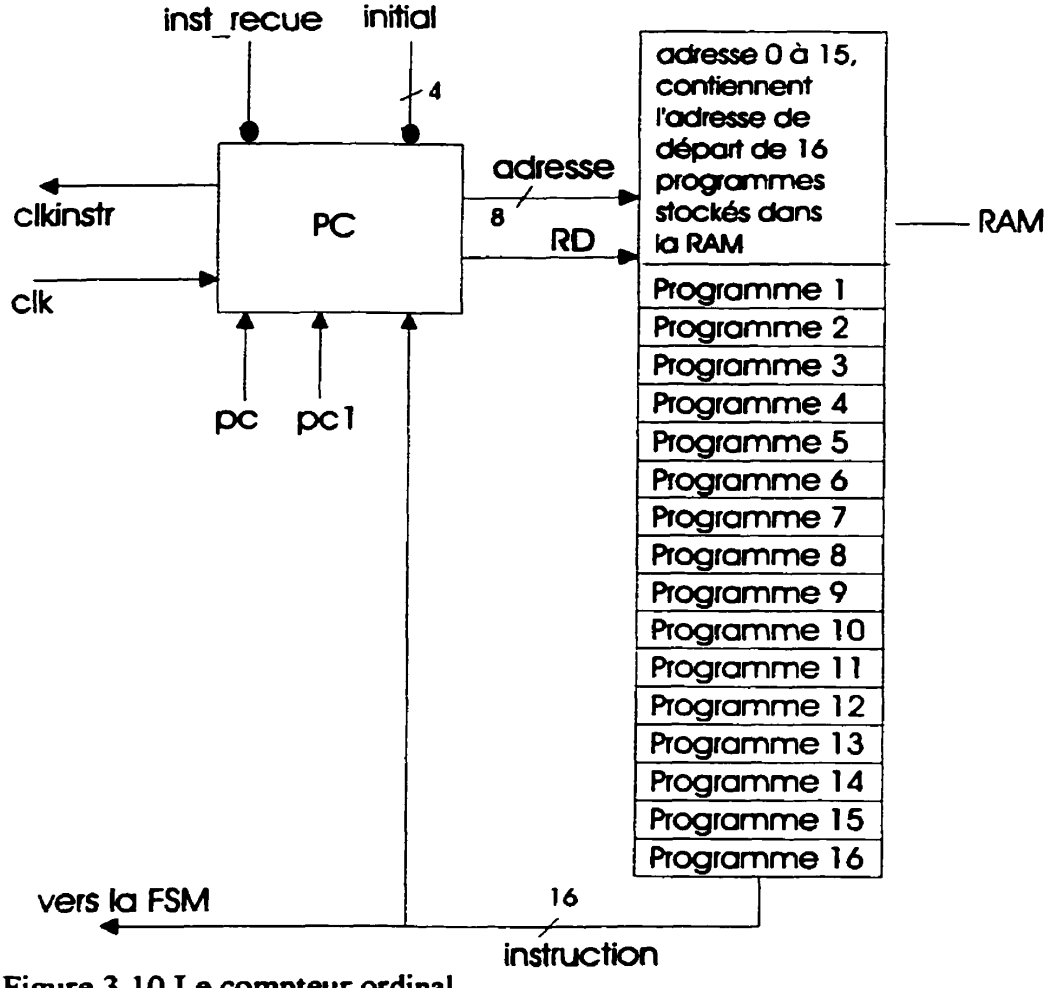


Figure 3.10 Le compteur ordinal

signal « instruction » s'adresse au compteur ordinal et non à la FSM de la machine bit-sérielle. La deuxième étape est d'aller se positionner à l'adresse de départ du programme à effectuer et d'attendre le signal de départ.

Nous allons maintenant traiter du mécanisme de communication entre la FSM, le contrôleur de la machine bit-sérielle, et le compteur ordinal. Reprenons la situation précédente. Le programme numéro 3, contenu dans la mémoire RAM, est celui que l'on

veut exécuter. Toute l'étape un est réalisée et le feu vert pour commencer l'exécution du programme est donné par le contrôleur global du processeur. Le compteur ordinal envoie l'adresse de départ à la RAM et active le signal « RD » pour un coup d'horloge. Le coup d'horloge suivant, le compteur ordinal active le signal « clkinst » qui indique à la FSM qu'une instruction valide est disponible. Le signal « clkinst » est maintenu actif jusqu'à ce que la FSM active le signal « inst_recue ». Ces deux signaux font en sorte partie d'un protocole de « hand shake ». Par la suite, la FSM décode les instructions, les exécute et active soit le signal « pc », soit le signal « pcl » à l'intention du compteur ordinal. Lorsque le signal « pc » est actif, le compteur ordinal ne fait qu'incrémenter l'adresse courante. Lorsque le signal « pcl » est actif, cela indique que le compteur ordinal doit envoyer l'adresse du début du programme numéro 3 qu'il avait mémorisé lors de la première étape. Ceci correspond à une instruction « jump », permettant de recommencer le programme sur un nouvel envoie ou un nouvel échantillon de données d'un certain type.

3.7.6 Le contrôleur (FSM) de la machine bit-sérielle

3.7.6.1 Le jeu d'instructions

La création d'un jeu d'instructions ne peut se faire sans tenir compte de la façon dont il sera implanté. Ainsi, l'architecture de la machine bit-sérielle a été développée en tenant compte du jeu d'instructions et vice versa. L'une des choses que l'on recherche en développant un jeu d'instructions et une architecture permettant de l'implanter est la simplicité et l'efficacité. Somme toute, l'architecture de la machine bit-sérielle présentée dans la figure 3.6 est assez simple. En effet, peu de blocs sont nécessaires à réaliser le traitement voulu sur les bits de données et à lui assurer une grande flexibilité.

Un des points majeurs de l'architecture de la machine bit-sérielle est que l'horloge des signaux vidéo est toujours beaucoup plus rapide que celle des données qui arrivent au

processeur ou qui doivent être transmises du processeur. L'horloge d'arrivée des bits audio en format AES/EBU est de 3,072 MHz alors que la fréquence vidéo la plus lente est de 27 MHz. La priorité du processeur est le traitement des données audio. De plus, les données audio sont, de celles qui sont définies à date, celles qui arrivent à la fréquence la plus élevée au processeur. Tout ceci pris en compte, les instructions ont été conçues de manière à pouvoir s'exécuter en un maximum de 8 coups d'horloge vidéo. Ceci vient du fait qu'entre deux bits de données reçus, il y a au moins 8 coups d'horloge vidéo, dépendant du standard en présence. Pour réaliser ce type d'instruction, il a fallu tenir compte de problèmes de synchronisation qui seront traités plus loin dans cette partie.

Le jeu d'instructions est défini par 12 instructions. Les 4 MSBs représentent le code opérationnel de l'instruction. Les autres bits du mot d'instruction (16 bits en tout) sont divers champs qui contrôlent les éléments suivants :

- reset du bloc de parité paire 1
- reset du bloc de parité paire 2
- enable du bloc de parité paire 1
- enable du bloc de parité paire 2
- enable write du registre à décalage A
- enable write du registre à décalage B
- enable write du FIFO principal
- enable write du FIFO auxiliaire
- enable read du FIFO principal
- enable read du FIFO auxiliaire
- signal d'incrémentation au compteur ordinal
- signal de saut au compteur ordinal
- sélection de l'entrée du multiplexeur A (3 bits)
- sélection de l'entrée du multiplexeur B

- sélection de l'entrée du multiplexeur C (4 bits)
- sélection de l'entrée du multiplexeur D (4 bits)
- signal de présence d'un bit valide pour le module biphase mark modulation

La description des instructions ne mentionne que ce qui est fait dans l'instruction, sans rapport aux instructions passées ou suivantes d'un programme. Le lecteur a intérêt à considérer les instructions individuellement dans cette section, sans faire de rapprochement entre elles. Les instructions sont les suivantes :

1. NOP

Comme son nom l'indique, cette instruction est une instruction qui ne fait rien qu'attendre l'arrivée d'un nouveau bit de données.

2. NOP MUX

Cette instruction attend une transition montante du signal « fsync » indiquant le début d'un nouvel échantillon de données. Cette instruction a été crée à la base pour la réception des données audio selon le protocole du CS8412. Elle peut cependant être utilisée pour synchroniser n'importe quel type de données.

3. MOVEBIT 1

Cette instruction sélectionne une des entrées du multiplexeur A du chemin de données ou son inverse (multiplexeur B) et l'inscrit dans un des registres à décalage de ce même chemin de données.

4. MOVEBIT 2

Cette instruction attend une transition montant sur « sck » (arrivée d'un nouveau bit de données), sélectionne une des entrées du multiplexeur A ou son inverse (multiplexeur B) et l'inscrit dans un des registres à décalage.

5. MOVEBIT 3

Cette instruction attend une transition montante sur « sck » (arrivée d'un nouveau bit provenant de la ligne de données externe au processeur). Elle inscrit ce bit dans le registre à décalage A et fait compter ce bit dans le calcul de parité paire 1. Elle inscrit le résultat du calcul de parité 1 ainsi que son inverse dans le même registre à décalage. Finalement elle inscrit le mot de 10 bits formé par le registre à décalage A dans le FIFO rattaché au bloc (i.e. audio1 ou audio2 ou LTC ou RS422) et remet à zéro le calcul de parité paire 1.

6. MOVEBIT 4

Cette instruction attend une transition montante sur « sck » (arrivée d'un nouveau bit provenant de la ligne de données externe au processeur). Elle inscrit un bit d'une des entrées du multiplexeur A dans le registre à décalage A. Elle permet d'ajouter ce bit dans le calcul de parité paire 1. Elle inscrit l'inverse de ce bit dans le registre à décalage A. Elle indique au compteur ordinal s'il doit incrémenter l'adresse ou faire un saut d'instruction. Elle inscrit le mot de 10 bits formé par le registre à décalage A dans le FIFO rattaché au bloc.

7. MOVEBIT 5

Cette instruction attend une transition montante sur « sck » (arrivée d'un nouveau bit provenant de la ligne de données externe au processeur). Elle inscrit ce bit dans le registre à décalage B. Elle inscrit un '0' ou un '1' suivi de son inverse dans le registre à décalage B. Finalement elle inscrit le mot de 10 bits formé par le registre à décalage B dans le FIFO auxiliaire présent dans les blocs audio 1 et audio 2 seulement.

8. MOVEBIT 6

Cette instruction attend une transition montante sur « sck » (arrivée d'un nouveau bit de donnée), sélectionne une des entrées du multiplexeur A ou son inverse (multiplexeur B) et l'inscrit dans un des registres à décalage. Elle fait compter ce bit dans le calcul de parité paire 1. Elle indique au module biphase mark qu'un bit est valide à son entrée.

9. MOVEBIT 7

Cette instruction permet de sélectionner un bit du multiplexeur A, de l'inverser et de l'inscrire dans un des registres à décalage A ou B. Elle permet de faire compter ce bit au calcul de la parité paire. Elle inscrit le mot formé par le registre à décalage A dans le FIFO associé au bloc.

10. MOVEWORD 1

Cette instruction inscrit le mot de 10 bits formé par le registre à décalage B dans le FIFO auxiliaire uniquement présent dans les blocs audio 1 et audio 2.

11. MOVEWORD 2

Cette instruction s'assure que les FIFO principal et auxiliaire (s'il y a le cas) contiennent des données et lit le ou les FIFO lorsque c'est le cas.

12. EDGEDETECT

Cette instruction permet de détecter une transition descendante sur la ligne des données sérielles et attend un coup d'horloge (sck) signifiant l'arrivée d'un nouveau bit (start bit).

Les instructions présentées plus haut permettent de transformer les bits en mots de 10 bits qui forment les UDWs des paquets ancillaires ou de sérialiser ces mêmes mots de 10 bits tout en les complétant avec d'autres bits, vers le monde extérieur. Comme on peut le voir dans le tableau 3.1, elles sont simples et ont toutes un temps d'exécution en deçà de 8 coups d'horloge vidéo. Un petit exemple commenté d'un bout de programme d'une machine bit-sérielle est disponible en annexe A.

Tableau 3.1 Nombre de cycles (horloge vidéo : clk) des instructions de la machine bit-sérielle

Instruction	Nombre de cycles d'horloge vidéo : clk
1. NOP	3
2. NOP MUX	4
3. MOVEBIT 1	3
4. MOVEBIT 2	4
5. MOVEBIT 3	7
6. MOVEBIT 4	6
7. MOVEBIT 5	6
8. MOVEBIT 6	5
9. MOVEBIT 7	5
10. MOVEWORD 1	4
11. MOVEWORD 2	5
12. EDGEDETECT	3

3.7.6.2 L'implantation du jeu d'instructions

La méthode retenue pour implanter ce jeu d'instructions est une machine à états finis synthétisée en parties IFL (Input Forming Logic), registres d'états et OFL (Output Forming Logic). Une fois synthétisée, il est impossible de changer quoi que ce soit aux instructions. La question qu'on peut se poser est pourquoi choisir une telle approche ? Les instructions réalisent très bien les différents traitements nécessaires pour les données audio, ltc, RS422 et il est fort probable qu'elles seront efficaces pour les autres types de données qui ne sont pas encore définis. Une approche alternative est de réaliser les instructions dans une RAM, ce qui assure un degré de flexibilité supplémentaire. En effet, cette approche permet d'avoir des instructions programmables. Puisque les instructions en place font le travail et qu'il est plus simple de synthétiser et de tester une

machine à états réalisée selon la méthode retenue avec les outils actuels (Synopsys), la RAM ou les autres méthodes possibles n'ont pas été retenues.

3.7.6.3 Métastabilité

Comme il a été indiqué plus haut, les instructions sont une suite d'états qui réalisent certaines actions sur les données et qui s'exécutent au plus en 8 coups d'horloge vidéo. De plus, on a vu que certaines de ces instructions devaient attendre l'arrivée d'un nouveau bit ou le signal du début d'un nouvel échantillon de données. Si on considère le cas des données audio, on doit à partir d'une machine à états, qui fonctionne à une des trois fréquences suivantes (27 MHz; 36 MHz; 74,25 MHz), se synchroniser sur des signaux qui ont des fréquences de 48 kHz ou 3,072 MHz. Ces signaux de basse fréquence peuvent, dans certaines situations, changer de niveau logique un bref moment avant le front montant de l'horloge de la machine à états, occasionnant des problèmes de métastabilité. Pour solutionner le problème, nous avons mis deux bascules D en série pour chacun des trois signaux d'entrée suivants (échantillonnés à la fréquence vidéo) de chaque bloc de données :

- SCK
- FSYNC
- SDATA.

Puisque l'instruction la plus longue a une durée de 7 cycles, l'addition d'un cycle de latence pour contrer l'effet de la métastabilité est acceptable. En effet, 7 cycles + 1 cycle de perdu (pour cause de métastabilité) font 8 cycles : ce qui est la limite supérieure permise de la durée d'une instruction.

3.7.7 L'apport d'une machine bit-sérielle

Nous venons de voir dans cette section, l'architecture et le besoin d'une machine bit-sérielle. Cette machine est une autre contribution de ce mémoire au niveau du traitement des données bit par bit. Certains processeurs, pour manipuler des mots de données, auraient tout d'abord reçu les données sériellement, les aurait assemblées en mot et aurait fait le traitement approprié. Pour manipuler les données d'une telle façon dans le processeur à données ancillaires intégré, il aurait fallu ajouter quelques autres registres et éléments de décalage au chemin de données de la machine bit-sérielle. Ce faisant, la machine bit-sérielle n'aurait pas été plus rapide que ce qu'elle est présentement. En effet, la machine bit-sérielle fonctionne à la fréquence vidéo qui est beaucoup plus rapide que le fréquence des données audio, LTC ou autres. En bout de ligne, nous n'aurions ajouter que plus de composants au chemin de données de la machine bit-sérielle, ce qui aurait augmenter la surface du circuit pour rien. Rappelons ici que la machine bit-sérielle est très efficace dans le traitement des données.

3.8 Le bus de données principal

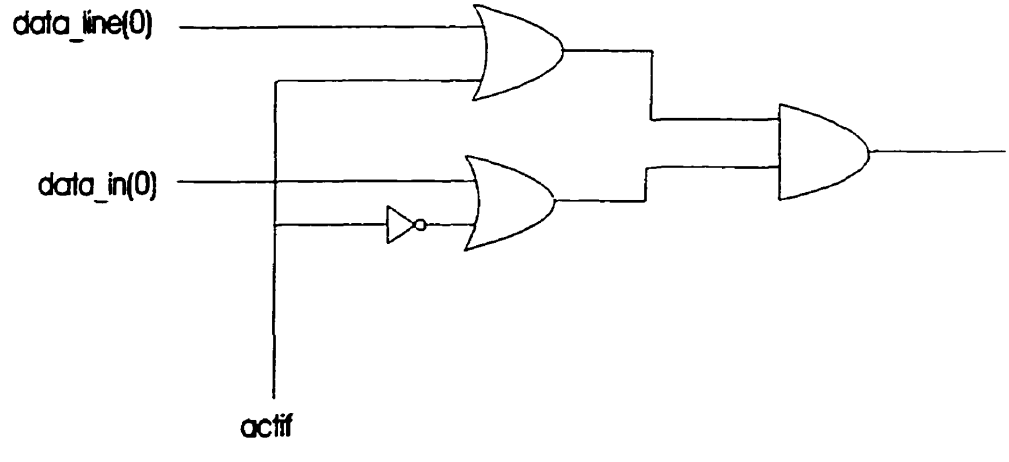


Figure 3.11 Cellule du bus de données

La figure 3.11 illustre la façon dont sont construits les bus de données et d'adresses du processeur. Cette figure montre une cellule de bus d'un bit. Pour faire un bus de 10 bits, il faut 10 cellules de ce genre mises en parallèle. Le signal « data_in(0) » correspond à une nouvelle entrée du bus, en provenance, par exemple, du fifo audio principal du bloc de données audio 1. Le signal « data_line(0) » correspond à la sortie d'une cellule de bus identique qui précède cette cellule. Le signal « actif » est un enable destiné à faire passer le bit sur « data_in(0) » en sortie lorsqu'il est à '1'. Autrement, il laisse passer en sortie le bit sur « data_line(0) ». Tous les signaux d'activation (enable) de tous les bus sont gérés par le contrôleur global du processeur. La figure 3.12 présente le bus de données principal du processeur. Douze cellules, qui représentent autant d'entrées sur le bus, le composent. Ces entrées viennent des différentes parties suivantes : Data RAM, FIFO principal du bloc audio 1, FIFO auxiliaire du bloc audio 1, FIFO principal du bloc audio2, FIFO auxiliaire du bloc audio 2, FIFO du bloc RS422, FIFO du bloc LTC, contrôleur global du processeur, 10 MSBs du bus d'entrée du processeur, 10 LSBs du bus d'entrée du processeur, module clock word relié aux blocs audio 1 et audio 2, module clock word relié aux blocs LTC et RS422. Au sujet des modules « clock word », ils ne seront pas expliqués au cours de cet ouvrage. Le lecteur doit savoir que le processeur est doté de ces blocs pour réaliser certains « User Data Words » d'un paquet ancillaire audio destiné à un format vidéo HDTV (voir figure 1.5c, les mots CLK sur la figure) (voir également SMPTE 299M pour de plus amples détails).

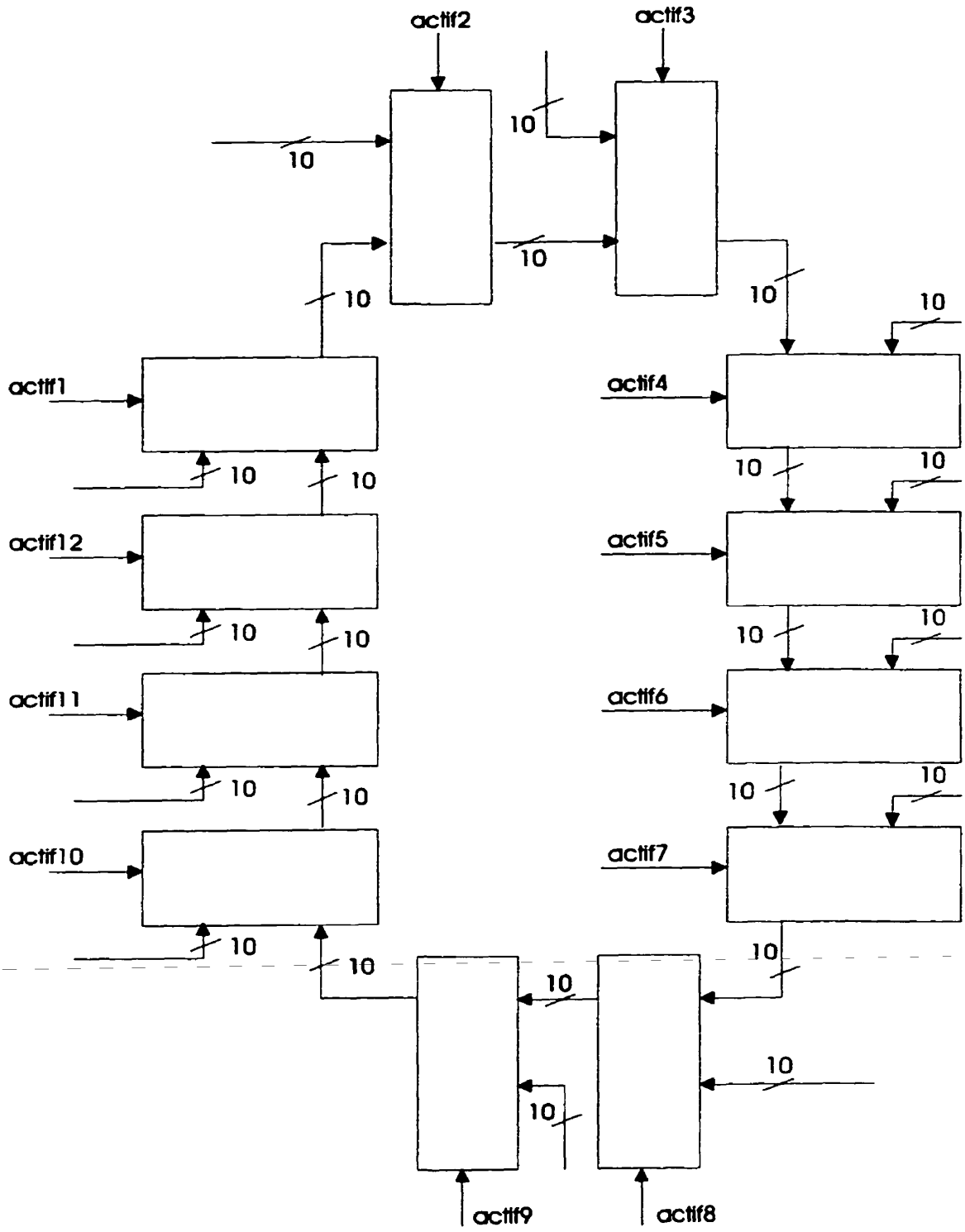


Figure 3.12 Bus de données principal

3.9 Le chemin de données (data path) d'un bloc de traitement de données

La figure 3.13 montre le chemin de données d'un bloc de transformation de données. Dans cet exemple, le bloc de données audio 1 a été choisi. Le bloc data path audio1 est celui de la machine bit-sérielle telle que discutée plus haut. Nous devons mentionner ici que les signaux de contrôle des différents multiplexeurs n'apparaissent pas sur la figure pour éviter d'alourdir cette dernière. Mentionnons tout de même que ces signaux sont gérés par le contrôleur global du processeur. Outre le FIFO audio 1 et le FIFO auxiliaire audio 1, ce schéma comprend aussi les blocs TONES audio 1, Copie audio 1 et Copie auxiliaire audio 1. Le bloc TONES contient les valeurs des différents échantillons audio destinés à produire un signal de 1 kHz de fréquence en sous-mode « tones » ou « tones & colour bar ». Les blocs Copie audio 1 et Copie auxiliaire audio 1 sont des blocs nécessaires lorsque le processeur est en mode DEMUX et qu'il y a de l'asynchronisme audio par rapport au vidéo. Leur utilité sera expliquée dans la sous-section suivante qui traite du contrôleur du processeur. On peut voir qu'en sous mode « normal », dépendamment du mode de fonctionnement en présence, les mots de 10 bits s'inscrivent dans les FIFOs en provenance du bus de données pour être ensuite sérialisés par la machine bit-sérielle (via le data path) ou bien les bits sont assemblés en mots de 10 bits, inscrits dans les FIFOs et mis sur le bus de données en temps opportun. Il est à noter également que sur cette figure, les signaux RD et WR des différents FIFOs n'apparaissent pas. Mentionnons également que les FIFOs ont été réalisés à base de bascules D et ce, en raison des disponibilités du moment (librairie de composants en technologie 0,35 um incomplète).

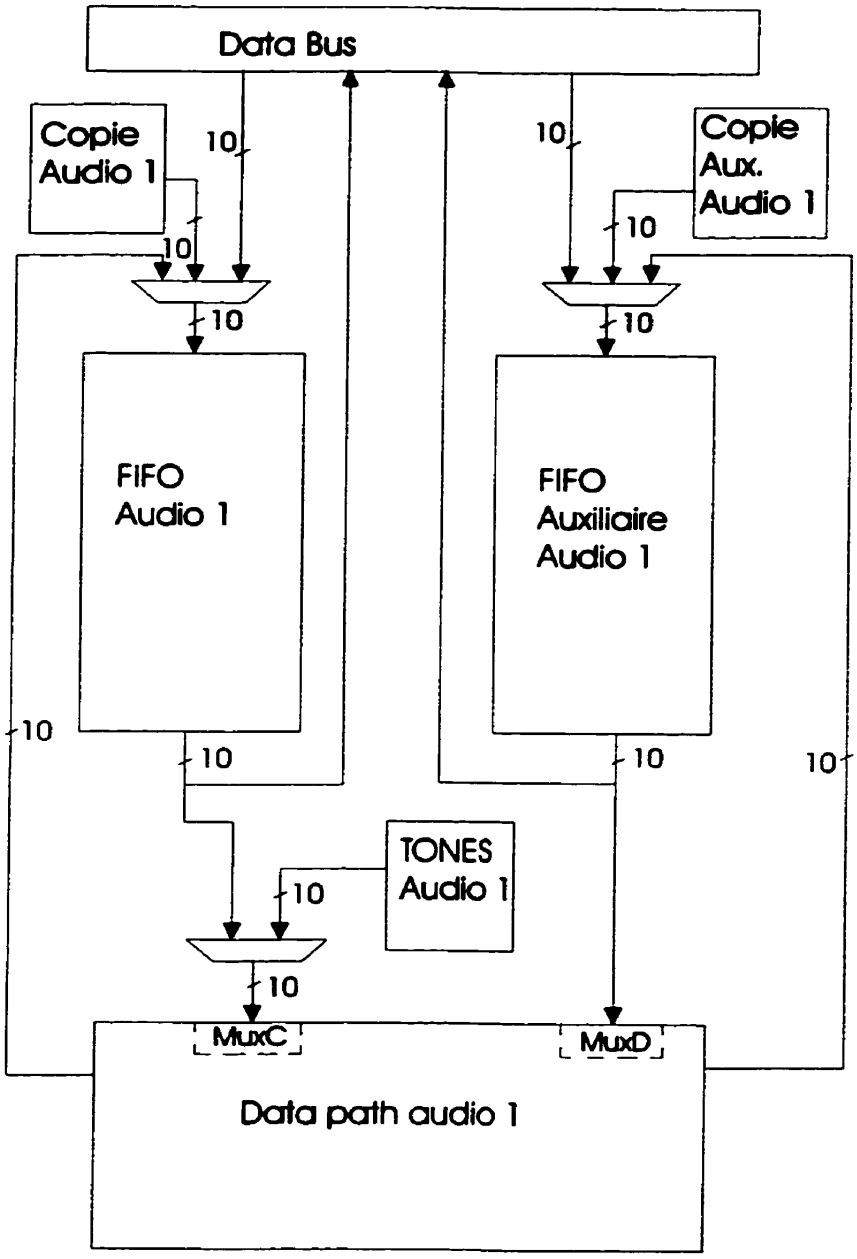


Figure 3.13 Chemin de données du bloc de données audio 1

3.10 Le contrôleur

Dans cette section, nous allons présenter le module le plus important du processeur, i.e. le contrôleur. La structure du contrôleur est faite de plusieurs modules. La fonctionnalité du processeur étant décomposée en deux volets (insertion, extraction), la plupart de ces modules sont actifs seulement en mode MUX ou en mode DEMUX. La manière adoptée pour partitionner le contrôleur en modules a consisté à décomposer sa fonctionnalité selon les différentes tâches à accomplir. Nous présentons dans cette section les principaux modules du contrôleur.

3.10.1 Le module A du contrôleur

La fonctionnalité de ce module est d'assembler les différents paquets fabriqués par le processeur, dans la RAM de données. Nous rappelons au lecteur que lorsque le processeur reçoit l'espace HANC d'une ligne vidéo en mode MUX, il doit insérer les nouveaux paquets assemblés dans la RAM de données et multiplexer les paquets déjà présents dans le vidéo d'entrée selon certains paramètres spécifiés par l'usager. Il eut été extrêmement difficile, voir même impossible, d'écrire les UDWs formant les paquets ancillaires en construction tout en lisant (pour les envoyer dans le signal vidéo de sortie) les paquets ancillaires déjà assemblés dans la RAM de données. L'espace HANC des différents formats vidéo représentent moins de 20% d'une ligne vidéo. Puisque 80% du temps, le processeur n'est pas occupé à insérer ou multiplexer des paquets ancillaires, nous avons choisi d'exploiter ce temps pour fabriquer et écrire dans la RAM de données, les différents paquets ancillaires que le processeur traite. À ce stade ci, le lecteur peut voir une des raisons qui justifie l'emploie de FIFOs pour les quatre blocs de données du processeur. En effet, en mode MUX, pendant la durée de l'espace HANC d'une ligne vidéo, le module A du contrôleur est inactif. Cependant, comme nous l'avons vu auparavant, cela n'empêche pas les bits des différents types de données traitées d'arriver par les quatre lignes de données sérielles.

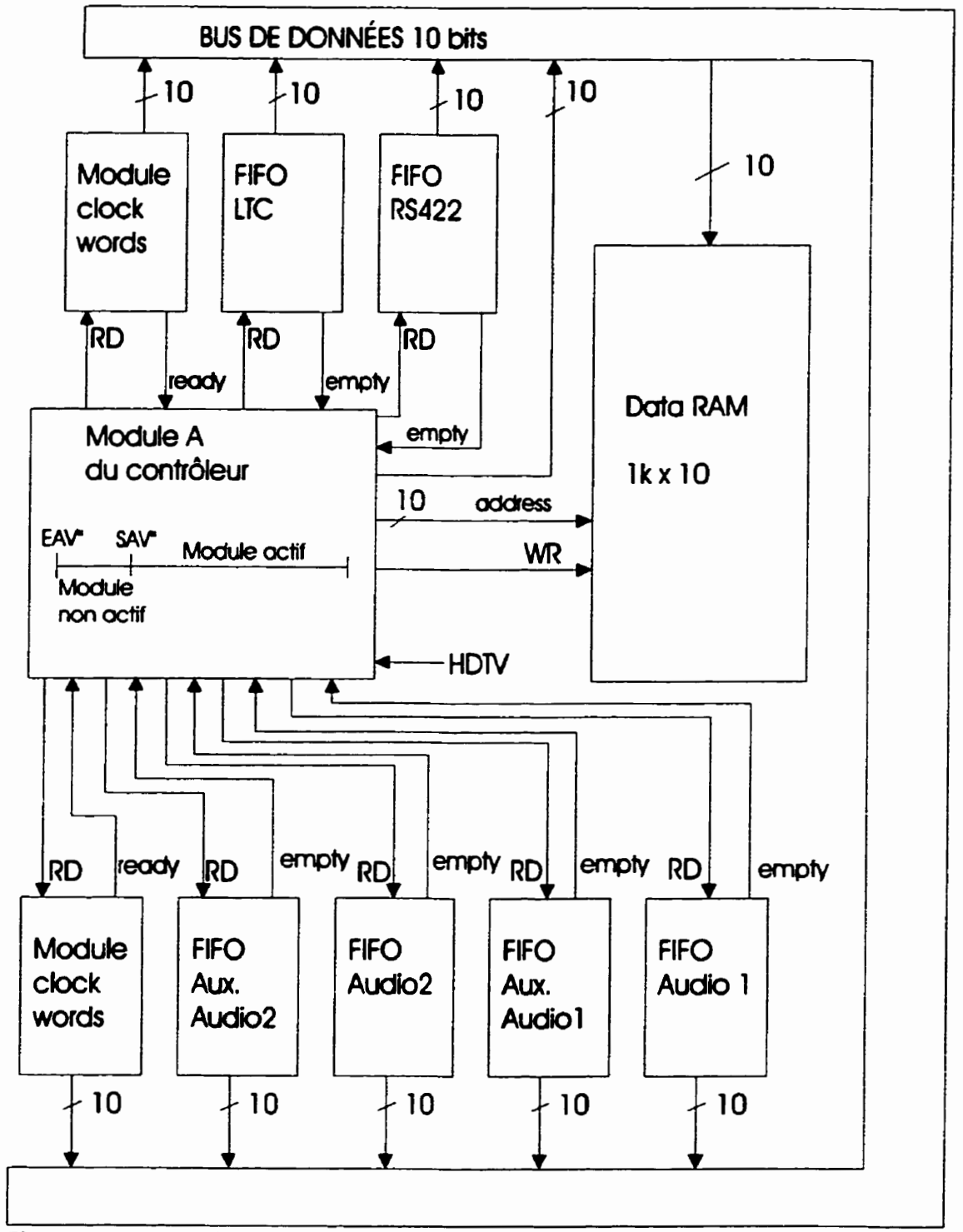


Figure 3.14 Module A du contrôleur en interaction avec le reste du processeur

Les machines bit-sérielles continuent d'assembler les bits en mots de 10 bits (UDWs) et ils sont écrits dans ces FIFOs.

La figure 3.14 résume la façon dont fonctionne le module A du contrôleur. Notons que même s'ils ne sont pas représentés sur cette figure, ce module reçoit plusieurs signaux de contrôle et plusieurs paramètres. Du module d'entrée du processeur, le module A reçoit les différents signaux l'informant du format vidéo en présence et sur l'endroit où on se trouve à tout moment sur une ligne vidéo (espace HANC, active picture). Le module A reçoit également une série de paramètres contenus dans la banque de registres du processeur ou sur les différents plots d'entrée du processeur. Ainsi, le module A reçoit des informations sur le nombre de bits qui forment les échantillons audio AES/EBU (20 ou 24 bits), sur la façon dont sont utilisés les blocs de données LTC et RS422 (ils traitent de l'audio en format vidéo HDTV ou leurs données LTC et RS422 respectives), les différents mots DID (Data IDentification) et DC (Data Count) nécessaires à la fabrication des paquets LTC et RS422, le groupe audio sélectionné, le sous-mode du processeur (normal, bypass, tones, etc.). Avec ces différents paramètres, le module A a pour responsabilité d'assembler correctement les différents paquets ancillaires dans les différents espaces mémoires prévus (selon le type du paquet) dans la RAM de données.

Plus précisément, ce module écrit l'entête d'un paquet ancillaire (000h, 3FFh, 3FFh), écrit le mot DID, procède à l'incrémentation et l'écriture du mot DBN et l'écriture du mot DC, dans la RAM de données. Par la suite, le module transfert les UDWs des différents FIFOs et modules « clock words » dans la RAM de données. Pour ce faire, le module A lit dans un premier temps le contenu du FIFO en question tout en activant le signal « enable » du bus de données principal auquel est connecté le FIFO. Puis, dans un deuxième temps, le module A active le signal d'écriture de la Data RAM, ayant préalablement calculé et envoyé sur le bus d'adresses, la bonne adresse où doit être écrit le mot (à ce moment, le mot à écrire est sur le bus de données principal).

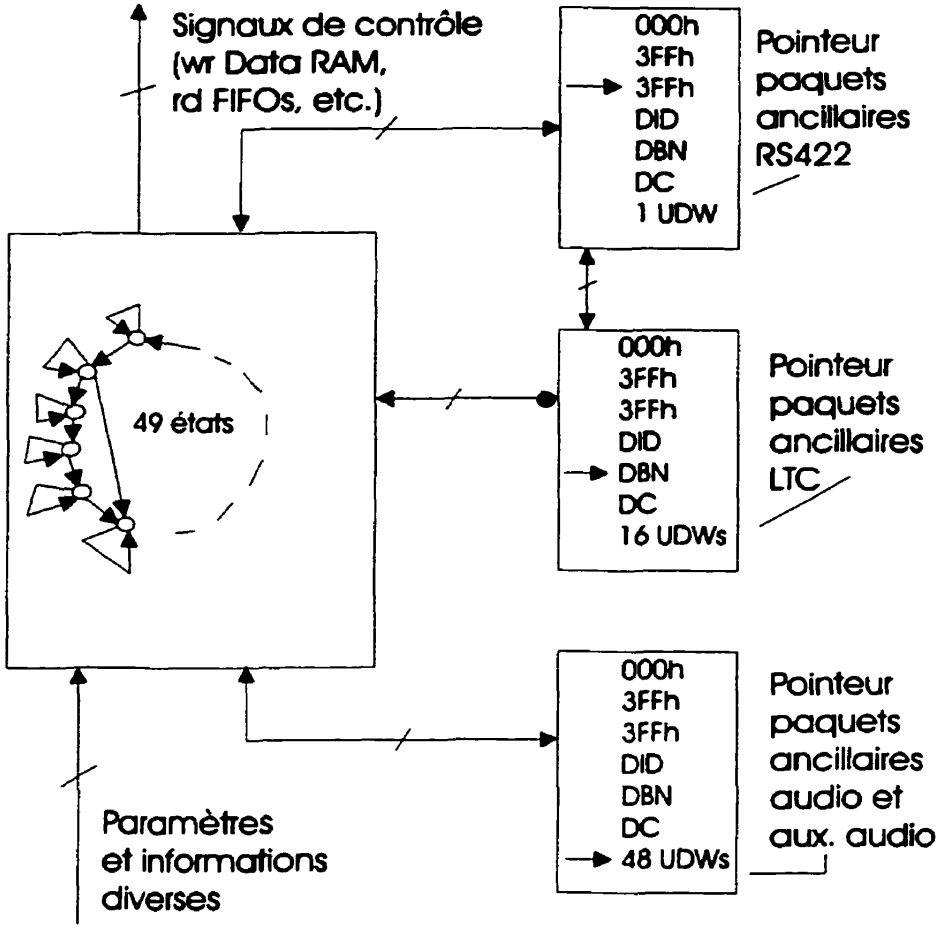


Figure 3.15 Le Module A en détail

La figure 3.15 montre les détails du module A du contrôleur. La pièce maîtresse de ce module est une machine à états finis de 49 états. La façon dont fonctionne cette machine est qu'elle contient l'information à envoyer aux autres modules pertinents du processeur en fonction de la tâche à accomplir. Par exemple, six états sont réservés pour l'inscription d'un entête (000h, 3FFh, 3FFh, DID, DBN, DC) de paquet ancillaire audio dans la RAM de données. Notons que lorsque c'est le module A qui inscrit les mots dans la RAM de données, un seul état est nécessaire par mot. Par contre, lorsque le mot à écrire est fabriqué par une unité fonctionnelle autre que le contrôleur du processeur, alors la tâche s'exécute en deux états (deux cycles d'horloge) : une pour mettre le mot

d'où il est contenu sur le bus de données, une pour écrire ce mot du bus de données dans la Data RAM. Nous avons représenté sur la figure 3.15, des états qui possèdent deux ou trois destinations. Nous n'entrerons pas dans les détails ici, mais le fait qu'un état puisse avoir trois destinations est une question de rapidité de la machine. Elle doit être suffisamment rapide pour que les différents FIFOs qu'elle gère se vident en moyenne plus rapidement qu'ils ne se remplissent. De plus, chaque état qui représente une tâche ou le premier état d'une tâche s'exécutant en deux états doit être capable de boucler sur lui-même pour toute la durée de l'espace HANC.

Les autres parties du module sont également des machines à états finis. Leur rôle est de pointer à travers les différents éléments formant un paquet ancillaire. Une série de signaux sont échangés entre la pièce maîtresse et les différents pointeurs de manière à ce que la pièce maîtresse sache quoi aller inscrire dans la RAM de données et les pointeurs conservent l'information à savoir où on est rendu dans la fabrication des paquets. Comme on peut le voir, trois pointeurs sont nécessaires : un pour les paquets audio (et auxiliaire audio), un pour les paquets LTC et un pour les paquets RS422. Ainsi, sur l'exemple de la figure, on est rendu à inscrire les UDWs d'un paquet ancillaire audio qui en comporte 48, on est rendu à inscrire le mot DBN d'un paquet ancillaire LTC (ATC) et on est rendu à inscrire le deuxième mot 3FFh d'un paquet ancillaire RS422. Nous devons également dire que les pointeurs LTC et RS422 unissent leurs efforts lorsque les deux blocs de données RS422 et LTC traitent des données audio en format vidéo HDTV. Nous devons également préciser que le mot CS, qui termine chaque paquet ancillaire, et les mots ECC d'un paquet ancillaire audio en format vidéo HDTV sont calculés et insérés par le module de sortie du processeur comme déjà mentionné dans une sous-section précédente.

L'implantation du module A est encore une fois composée de machines à états synthétisées en bascules D, parties IFL et OFL. Elles sont donc « coulées dans le béton ». La raison est simple, ce module fait des tâches répétitives et grâce à certains

paramètres qu'il reçoit, il est en mesure de fabriquer n'importe quel paquet ancillaire ayant n'importe quel DID et ayant un nombre de UDWs (DC) variable.

Un mot aussi sur la fabrication des paquets ancillaires audio en format vidéo DTV. Le nombre de UDWs d'un paquet n'est pas constant et est soit de 36, soit de 48. Dépendamment du format vidéo en présence, il faut insérer un certain nombre d'échantillons audio dans une fenêtre précise de trames vidéo. Nous n'entrerons pas dans les détails ici, mais mentionnons seulement qu'un circuit basé sur un accumulateur de phase et une division de deux nombres renseigne le module A, à chaque début de paquet ancillaire audio et ancillaire auxiliaire audio, sur le nombre de UDWs que doivent contenir ces paquets.

3.10.2 Le module D du contrôleur

Ce module est le plus important du contrôleur et donc du processeur en entier. C'est le module chargé du multiplexage des nouveaux paquets ancillaires et des anciens présents dans le signal vidéo d'entrée en mode MUX. Cependant, il est également actif en mode DEMUX. À haut niveau, on peut décrire la fonctionnalité de ce module comme suit. Tout d'abord, lorsqu'il reçoit le signal du début du traitement du signal vidéo, ce module initialise le délai de 74 mots nécessaire dans la RAM vidéo. Ensuite, il laisse passer le contenu de la région active picture d'une ligne vidéo jusqu'à ce qu'il rencontre la séquence EAV. Ensuite, il insère les nouveaux paquets ancillaires audio lorsque présents dans la RAM de données, retransmet ou non les paquets ancillaires présents dans l'espace HANC du signal vidéo d'entrée selon les spécifications de l'usager, insère un nouveau paquet ancillaire LTC (ATC) moyennant quelques conditions satisfaites et insère un nouveau paquet RS422 prêt à l'être et présent dans la RAM de données.

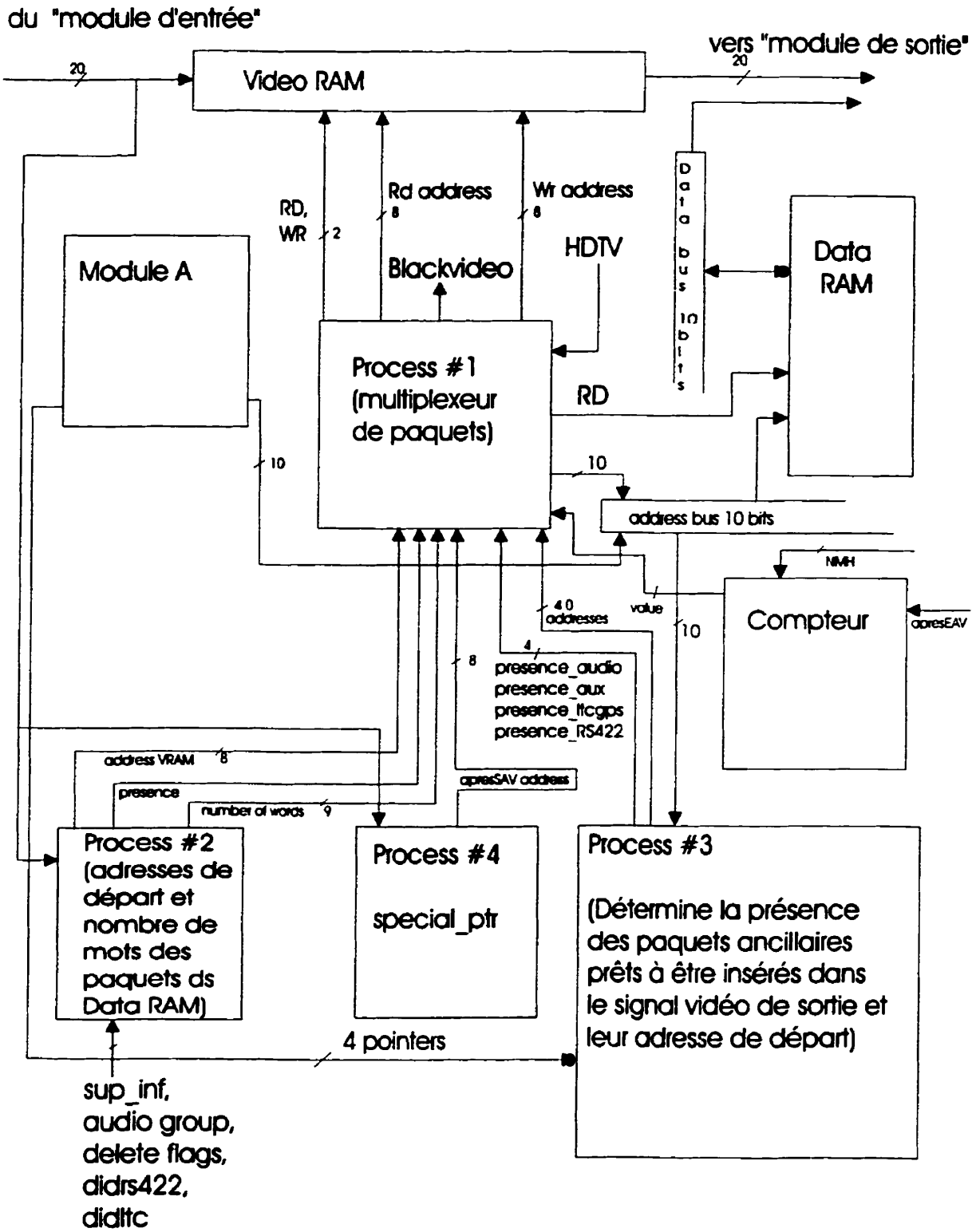


Figure 3.16 Architecture du module D en interaction avec quelques éléments de l'ADP

La figure 3.16 montre l'architecture du module D du contrôleur. Il est constitué de quatre processus et d'un compteur.

Le processus numéro 1 a pour fonction la gérance de l'espace HANC de chaque ligne vidéo. À chaque espace HANC, le processeur vérifie qu'il ne se trouve pas dans une ligne de changement de champ d'une trame vidéo ou dans la ligne suivante. Si ce n'est pas le cas, ce processus insère un nouveau paquet audio qui est présent et complété dans la RAM de données. Lorsque le paquet se transmet vers le module de sortie, le processus est « à l'écoute » du bus de données. Ce faisant, il est à la recherche du mot DC qui lui dit combien il reste de mots à lire de la RAM de données. Notons ici que bien que le mot CS terminant tout paquet ancillaire n'est pas présent dans la RAM de données, le processus numéro 1 laisse toujours un cycle d'horloge de temps mort entre le dernier UDW qu'il a lu et la prochaine décision qu'il prend. Ceci est fait pour permettre l'insertion du mot CS par le module de sortie du processeur sans écraser un mot. Ensuite, dépendamment des formats vidéo et audio (20 ou 24 bits) en présence, le processus #1 prend une des décisions suivantes. Il insère soit un paquet ancillaire auxiliaire audio si le format vidéo en est un de DTV et que les échantillons audio sont codés sur 24 bits, ou bien il insère un second paquet ancillaire audio s'il y en a un autre de complété dans la RAM de données et que le format vidéo en présence est un format HDTV. De plus, si les blocs de données LTC et RS422 ont pour mandat le traitement de données audio en présence d'un format vidéo HDTV, le même traitement est appliqué à la suite, i.e. on insère un nouveau paquet ancillaire audio présent dans la RAM de données ou deux s'il y en a plus d'un présent. Mentionnons également que durant toute la durée de l'insertion de nouveaux paquets audio, le processus active le signal « ChoiceForData » à l'intention du module de sortie du processeur.

Après l'insertion des nouveaux paquets audio, dépendamment du contenu de l'espace HANC du signal de vidéo d'entrée, le processus numéro 1 prend les décisions suivantes. Ce processus reçoit d'un autre processus du module D l'information qui le renseigne sur

la présence de paquets ancillaires présents dans la RAM vidéo. Si l'information lui dit qu'il y en a, le processus retransmet ce ou ces paquet(s) en s'aidant de certaines autres informations transmises par le processus numéro 2. Ce faisant, le processus 1 désactive le signal « ChoiceForData ». Il est à noter que le compteur, ainsi que le processus 2 du module D, transmettent des informations qui font qu'en aucun cas le module D ne retransmet des paquets ancillaires de la RAM vidéo vers la sortie, si la longueur de ces paquets fait déborder l'espace HANC. En effet, le nombre de mots d'une ligne vidéo propre à un certain format est critique. Il doit toujours y avoir le même nombre de mots réservés pour l'active picture, l'espace HANC, la séquence EAV (4) et la séquence SAV (4). En cas de non-retransmission de paquet pour cause de débordement de l'espace HANC, le ou les paquet(s) sera(ont) tout simplement écrasé(s) dans la RAM vidéo.

Le numéro de la ligne présentement traitée, le sous mode de fonctionnement du processeur et le fait que les blocs de données LTC et RS422 traitent de l'information audio influence l'action suivante. Il est certain que lorsque ces blocs de données traitent de l'information audio, ils ne peuvent pas en plus fabriquer des paquets ancillaires LTC et RS422. Cependant, lorsque ces blocs traitent du LTC ou du RS422, l'insertion de nouveaux paquets ancillaires LTC présents dans la RAM de données et suivant la retransmission des anciens paquets dans la RAM vidéo peut se faire. Elle se fait seulement si on est à la bonne ligne d'insertion, si on est dans le sous mode de fonctionnement « normal » et qu'il y a assez de place pour l'insertion du paquet dans l'espace HANC.

Suivant la décision expliquée au paragraphe précédent, le processus numéro 1 insère un nouveau paquet RS422 si il y en a un de prêt dans la RAM de données et que le bloc de données RS422 traite des données RS422 et qu'il y a suffisamment de place dans l'espace HANC.

Lorsqu'il reste de la place dans l'espace HANC, que les nouveaux paquets ancillaires qui pouvaient l'être ont été insérés et qu'il n'y a plus de paquets à retransmettre, le processus 1 active le signal « BlackVideo ». Ce signal fait en sorte que le module de sortie du processeur remplit le vide par un signal vidéo noir.

Évidemment, certains événements peuvent arriver pour faire en sorte qu'on ne retrouvera pas en sortie un espace HANC tel que décris ci-haut. Ainsi, il est possible que les paquets ancillaires du signal vidéo d'entrée ne soient pas concaténés au début de l'espace HANC d'une ligne. Cela ne change rien pour les nouveaux paquets ancillaires audio toujours insérés au début de l'espace, mais il se pourrait par exemple qu'après avoir retransmis deux paquets ancillaires présents dans le signal vidéo d'entrée, le processus reçoive à un moment ultérieur, l'information exacte qui lui dit qu'il n'y a pas d'autres paquets ancillaires présents dans la RAM vidéo. Le processus prend donc la décision d'insérer un nouveau paquet RS422 présent dans la RAM de données puisque la ligne en présence n'est pas la ligne d'insertion des nouveaux paquets LTC. Or à la suite de cette insertion, il est possible qu'un paquet ancillaire présent dans l'espace HANC du signal vidéo d'entrée se trouve maintenant dans la RAM vidéo puisque tous les paquets ancillaires de ce signal n'étaient pas concaténés au début de l'espace HANC. Le processus numéro 1 le retransmettra, mais l'espace HANC résultant contiendra alors une alternance pèle mêle de nouveaux paquets et d'anciens paquets. Cette situation est acceptable bien que non souhaitée. Également, le processus 1 et le module D dans l'ensemble est capable d'éliminer au plus 74 « trous » de l'espace HANC reçu en entrée. Nous entendons par trou tout paquet ancillaire qu'il ne faut pas retransmettre sans avoir inséré de nouveaux paquets ancillaires de même DID à leur place, ou tout simplement certains mots présents entre deux paquets ancillaires non concaténés et n'appartenant à aucun paquet ancillaire. On peut donc dire que le module D possède une méthode efficace pour la correction de certaines erreurs de l'espace HANC reçu en entrée.

Nous allons maintenant discuter du processus numéro 2 du module D. Tout d'abord, mentionnons que ce processus reçoit divers paramètres tels que les DID (et groupe audio) des paquets audio qu'on désire traiter ainsi que des drapeaux (flags) indiquant si on doit retransmettre ou non les paquets ayant ces DID et présents dans le signal vidéo d'entrée. Ces drapeaux sont spécifiés par l'usager et peuvent être en tout temps modifiés à l'aide de l'interface de communication I^2C que nous verrons un peu plus loin dans cette sous-section. Le processus numéro 2 est réalisé à l'aide de deux tableaux et une machine à états simple. Les tableaux contiennent respectivement l'adresse de départ des paquets à retransmettre dans la RAM vidéo, ainsi que leur longueur en nombre de mots. La machine à états détecte tous les paquets ancillaires présents dans le signal vidéo d'entrée et inscrit les différentes coordonnées de ces paquets dans les tableaux lorsqu'elle rencontre un paquet que l'usager veut traiter et que le drapeau associé à ce type de paquet n'est pas « levé ». Ces tableaux sont accédés par le processus numéro 1 qui examine deux choses. D'une part il vérifie si le pointeur de lecture du tableau contenant les adresses de départ des paquets ancillaires à retransmettre dans la RAM vidéo ne pointe pas au même endroit que le pointeur d'écriture de ce même tableau (implique qu'il y a des paquets de présent dans la RAM vidéo qu'on désire retransmettre dans le signal vidéo de sortie). D'autre part, le processus numéro 1 lit le nombre de mots du premier paquet à retransmettre dans la RAM vidéo et vérifie s'il y a de la place pour le retransmettre en sortie en comparant ce nombre de mots avec la valeur du compteur du module D qui indique le nombre de mots restant dans l'espace HANC. Notons que si le premier paquet à être retransmis ne peut l'être en raison de sa trop grande taille, le processus numéro 1 va voir, tant qu'il y en a, la longueur et l'adresse de départ du prochain paquet à retransmettre.

Le processus numéro 3 est en continuelle communication avec le module A comme on peut le voir sur la figure. Sa fonctionnalité est de renseigner le processus numéro 1 sur la présence et les adresses de départ des nouveaux paquets ancillaires prêts à être insérés dans le signal vidéo de sortie.

Comme on a déjà mentionné, il est essentiel de voir à ce que l'espace HANC de chaque ligne vidéo contienne le nombre de mots prescrit par le standard vidéo en présence. Hors, lorsque l'on modifie l'espace HANC d'une ligne en insérant de nouveaux paquets et en retirant certains de ceux qui y étaient déjà présents, on s'expose à certains risques de retransmettre des espaces HANC n'ayant pas le nombre de mots voulu. Le processus numéro 4 a en ce sens, une fonctionnalité critique. Il doit à chaque ligne détecter la séquence SAV indiquant la fin de l'espace HANC et donc le début de la zone active picture. La détection de la séquence s'effectue sur la ligne vidéo qui entre dans la RAM vidéo (elle y est inscrite). À la suite de la détection de la séquence, le processus 4 doit mémoriser l'adresse du début de la séquence (4 mots) dans la RAM vidéo. L'adresse est communiquée au processus numéro 1, qui lorsqu'il reçoit le signal de terminaison de l'espace HANC par le module d'entrée du processeur, va immédiatement lire l'adresse pour faire passer la séquence SAV en sortie.

Un dernier élément vient compléter le module D. Il s'agit du compteur. Au début de la région de l'active picture de chaque ligne, ce compteur est chargé avec la valeur du nombre de mots que contient le standard vidéo en présence. Rappelons ici au lecteur que c'est le module d'entrée du processeur qui, par son module de détection automatique du standard vidéo en présence, charge le compteur avec le signal NMH (Nombre de Mots dans l'espace HANC). Lorsque l'on entre dans l'espace HANC, le module d'entrée du processeur lance un signal de départ au compteur qui à partir de ce moment, commence à décrémenter à chaque coup d'horloge. La valeur qu'il envoie au processus 1, NMRH (Nombre de Mots Restant dans l'espace HANC), permet à ce dernier de faire des comparaisons pour juger de la retransmission ou de l'insertion de nouveaux paquets ancillaires, selon la place disponible dans l'espace HANC.

INSTRUCTION A : Écrire dans la VIDÉO RAM
INSTRUCTION B : Écrire et lire dans la VIDÉO RAM
INSTRUCTION C: Écrire dans la VIDÉO RAM, lire dans la DATA RAM (nouveaux paquets)
INSTRUCTION D: Écrire et lire dans la VIDÉO RAM (anciens paquets)
INSTRUCTION E : Écrire et lire dans la VIDÉO RAM + activer le signal "BlackVideo"

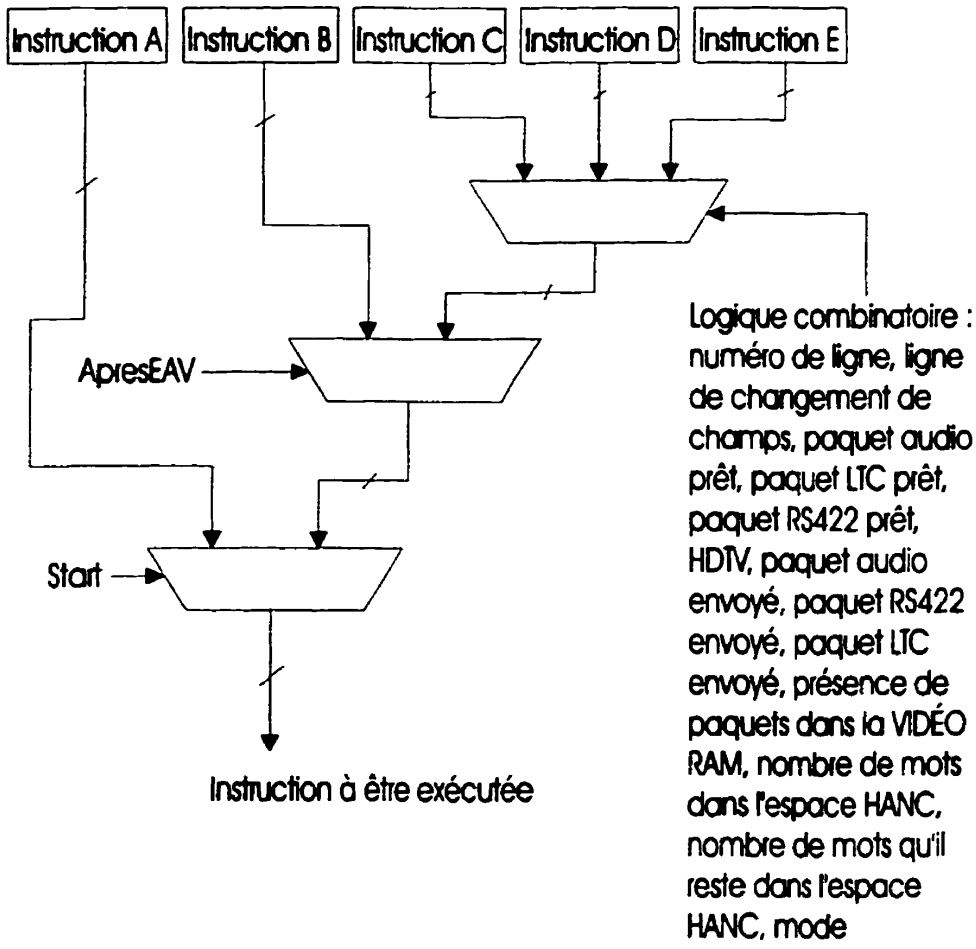


Figure 3.17 Une autre façon de voir le module D

La figure 3.17 résume le fonctionnement du module D du contrôleur. On peut voir le module D comme un ensemble de cinq instructions. La décision d'exécuter une instruction ou une autre dépend d'un réseau combinatoire de plusieurs signaux tels le mode de fonctionnement du processeur, le numéro de la ligne, le sous mode de fonctionnement, l'endroit où on se trouve dans la ligne vidéo, le nombre de mot de l'espace HANC, etc. L'instruction A est utilisée au départ pour créer le tampon de 74 mots dont on a besoin tout le long du fonctionnement du processeur. Ainsi, on écrit 74 mots dans la Vidéo RAM. L'instruction B est utilisée pendant la majeure partie des lignes vidéo. On ne fait qu'écrire un nouveau mot dans la RAM vidéo tout en lisant un mot. Cette instruction est utilisée dans la partie « active picture » de la ligne où aucun traitement en particulier n'est à exécuter sur les mots qui la composent. L'instruction C est utilisée pour insérer un nouveau paquet ancillaire présent dans la RAM de données. On insère le paquet en lisant la RAM de données tout en écrivant les mots que l'on reçoit du signal de vidéo d'entrée, dans la RAM vidéo. Cette instruction est évidemment exécutée pendant le transfert complet du nouveau paquet ancillaire. L'instruction D est utilisée pour retransmettre les anciens paquets présents dans la RAM vidéo. On lit et écrit simultanément un mot dans la RAM vidéo comme pour l'instruction B, à la différence qu'un saut d'adresse pour la lecture peut être nécessaire dans l'instruction D. En effet, le prochain paquet à retransmettre n'est pas nécessairement à l'adresse suivant immédiatement celle du dernier mot lu de la RAM vidéo. Finalement, l'instruction E ressemble aussi à l'instruction B. On écrit et lit un mot dans la RAM vidéo, mais en activant le signal « BlackVideo ». Cela informe le module de sortie qu'il ne doit pas retransmettre le mot provenant de la RAM vidéo, mais bien un mot de vidéo noir.

Notons en terminant que le module D du contrôleur est également actif en mode DEMUX. En fait, il réalise essentiellement les mêmes tâches dans la mesure où l'usager peut décider de ne pas retransmettre certains types (DID) de paquets dans le signal vidéo de sortie. La différence majeure entre la fonctionnalité du module dépendamment du

mode de fonctionnement en présence est qu'il est évidemment impossible au module D d'insérer de nouveaux paquets dans le signal vidéo. Autrement dit, l'instruction C est inutilisée dans le mode DEMUX.

Encore une fois, nous avons implanté le module D en plusieurs machines à états dédiées et en compteur. La raison est simple, dans les machines à états, on retrouve à la fois l'instruction qui doit être exécutée ainsi que tous les séquencements possibles d'une instruction à une autre. De plus, la façon de voir le module D en cinq instructions résume bien ce dont on a besoin pour traiter l'espace HANC d'un signal vidéo. Avec les paramètres fournis par les registres internes du processeur, une grande flexibilité est permise dans le traitement de l'espace HANC. Tous les autres signaux viennent d'autres modules du processeur ou carrément du monde extérieur. En ce sens, les machines à états finis sont très efficaces et assurent un degré de flexibilité non négligeable. De plus, rien ne porte à croire pour le moment que certaines autres instructions devront être définies pour mieux traiter l'espace HANC.

3.10.3 Le module B du contrôleur

Un autre module du contrôleur extrêmement important est le module B. Il est actif uniquement en mode DEMUX. La figure 3.18 nous montre son schéma bloc.

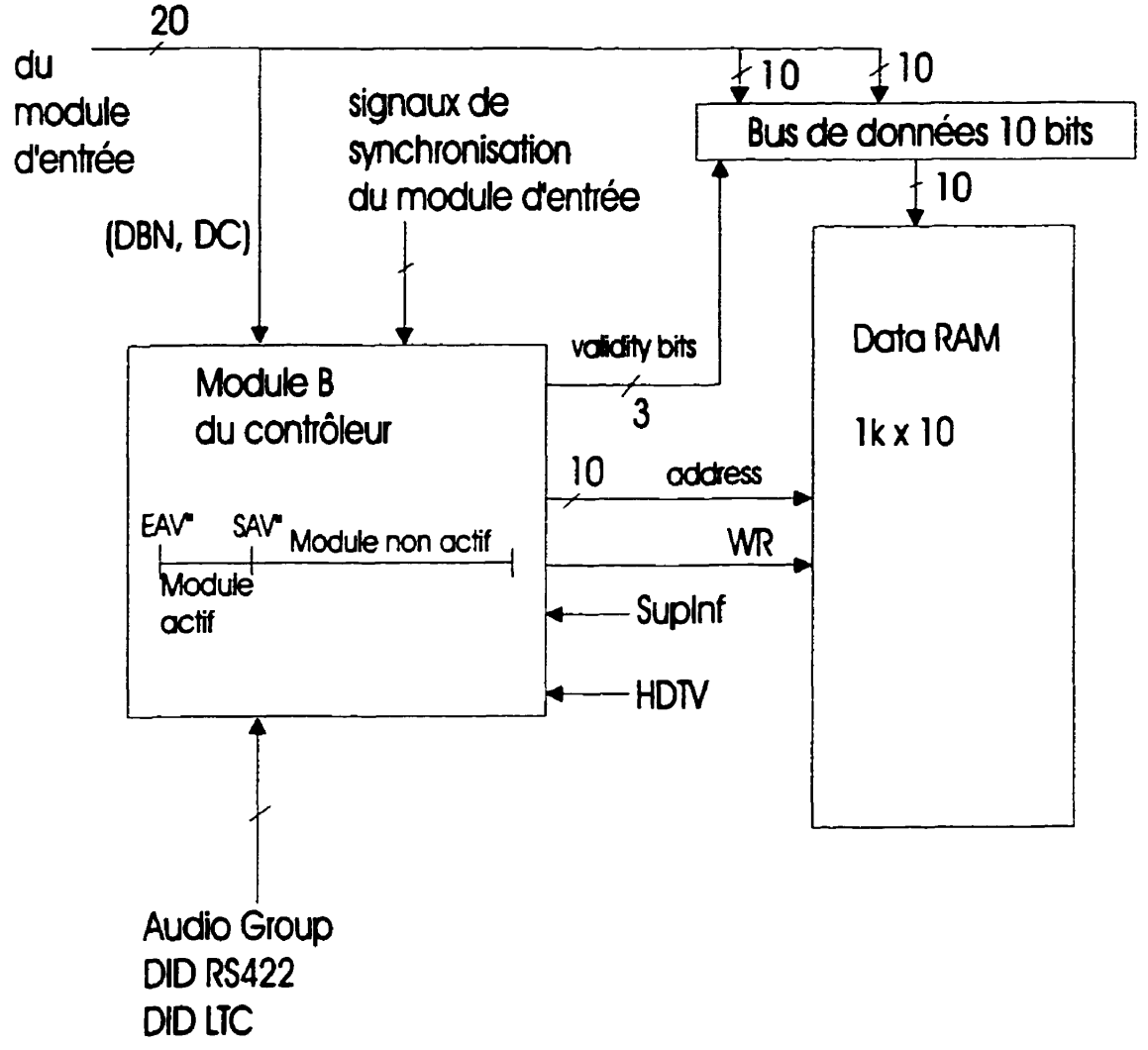


Figure 3.18 Module B du contrôleur

Tout d'abord, nous devons mentionner que les différents paramètres qui servaient à fabriquer les différents paquets que l'usager voulait insérer dans le signal vidéo de sortie en mode MUX, (DID, DC, audio group, etc.) sont les mêmes qui servent au module B.

Dans le mode DEMUX, la fonctionnalité de ce module du contrôleur est de détecter les paquets qui intéressent l'usager et d'aller inscrire les UDWs de ces paquets dans les endroits réservés à cette fin dans la RAM de données. Comme on peut le voir sur la figure 3.18, le module est évidemment actif lors de l'espace HANC de la ligne vidéo et est inactif lors de l'active picture.

Les mots DBN et DC ont été inscrits sur cette même figure. Le module B pratique un traitement plus spécifique sur les paquets ancillaires audio. Tout d'abord, le module s'assure que chaque mot DBN d'un paquet ancillaire audio reçu est soit égal ou soit égal à un de plus que le DBN du paquet ancillaire audio reçu précédemment. Lorsque ce n'est pas le cas, le module B doit mettre à '1', tous les « validity » bits des échantillons audio présents dans le paquet (voir chapitre 1 pour les détails des UDWs des échantillons audio). Évidemment, ces mots seront également inscrits dans la Data RAM, mais le fait que leur validity bit soit à '1', indique au monde extérieur que ces échantillons audio ne sont pas valides.

L'autre traitement est relié au mot DC des paquets ancillaires audio. On a vu au chapitre 1 que les échantillons audio insérés ou extraits d'un signal vidéo DTV sont codés en trois UDWs de 10 bits chacun. Il est très important que les échantillons soient transmis vers le monde extérieur en entier et non à demi ou au trois quart. Ce genre de situation pourrait arriver si une erreur était présente sur le mot DC d'un paquet ancillaire audio. Pour palier à la situation, le traitement suivant est appliqué aux paquets ancillaires audio avant que leurs UDWs soient inscrits dans la RAM de données. Tout d'abord, le module lit et mémorise la valeur DC initiale qui se trouve dans le paquet. Le module soustrait trois de cette valeur et inscrit les trois premiers UDWs seulement si le résultat de la soustraction est plus grand ou égal à zéro. L'opération de soustraction se répète et les UDWs sont inscrits dans la RAM de données jusqu'à ce que le résultat soit plus petit que zéro. Bien sur, ce traitement n'est pas infaillible. Un paquet qui aurait une valeur DC erronée indiquant 72 UDWs alors qu'il en contiendrait en réalité 36 n'empêcherait

pas d'inscrire dans la RAM de données des UDWs qui n'en sont pas. On ne peut rien faire dans cette situation. Tout ce qu'on peut faire est de s'assurer que les UDWs inscrits dans la RAM de données le sont par multiple de trois. Ce traitement n'est pas nécessaire pour les paquets ancillaires audio d'un signal vidéo HDTV. En effet, nous avons également vu dans le chapitre 1 que les paquets audio pour l'HDTV sont fabriqués différemment de ceux pour les formats DTV. Nous rappelons ici au lecteur que les paquets ancillaires audio pour l'HDTV sont faits de 24 UDWs dont 16 contiennent les échantillons codés. Cette valeur étant constante et les UDWs pertinents étant toujours aux mêmes positions pour tous les paquets, le module B court-circuite les erreurs possibles du DC et inscrit toujours 16 UDWs pour chaque paquet ancillaire audio rencontré, dans la RAM de données.

Ce genre de traitement est critique uniquement pour les paquets ancillaires audio. Il se peut que des erreurs sur le mot DC de d'autres types de paquet surviennent, mais comme nous l'avons dit plusieurs fois dans cet ouvrage, la priorité du processeur à données ancillaires est le traitement des données audio.

3.10.4 Le module C du contrôleur

Le module C est la contre partie du module B en ce sens qu'également actif en mode DEMUX, il est en action dans la partie active picture des lignes vidéo et est inactif dans les parties HANC de ces mêmes lignes. La fonctionnalité de ce module est de transférer les UDWs des différents paquets ancillaires qui intéressent l'usager aux différents FIFO des blocs de données appropriés. Tout comme les modules A et D en mode MUX, la décomposition de la fonctionnalité du mode DEMUX en deux principaux modules fait en sorte qu'on n'a pas besoin d'avoir un mécanisme de lecture et d'écriture simultanée, ce qui serait plus complexe et exigerait une RAM de données dotée de deux ports d'adresse (permettant une lecture sur l'un des ports d'adresse et une écriture sur l'autre.).

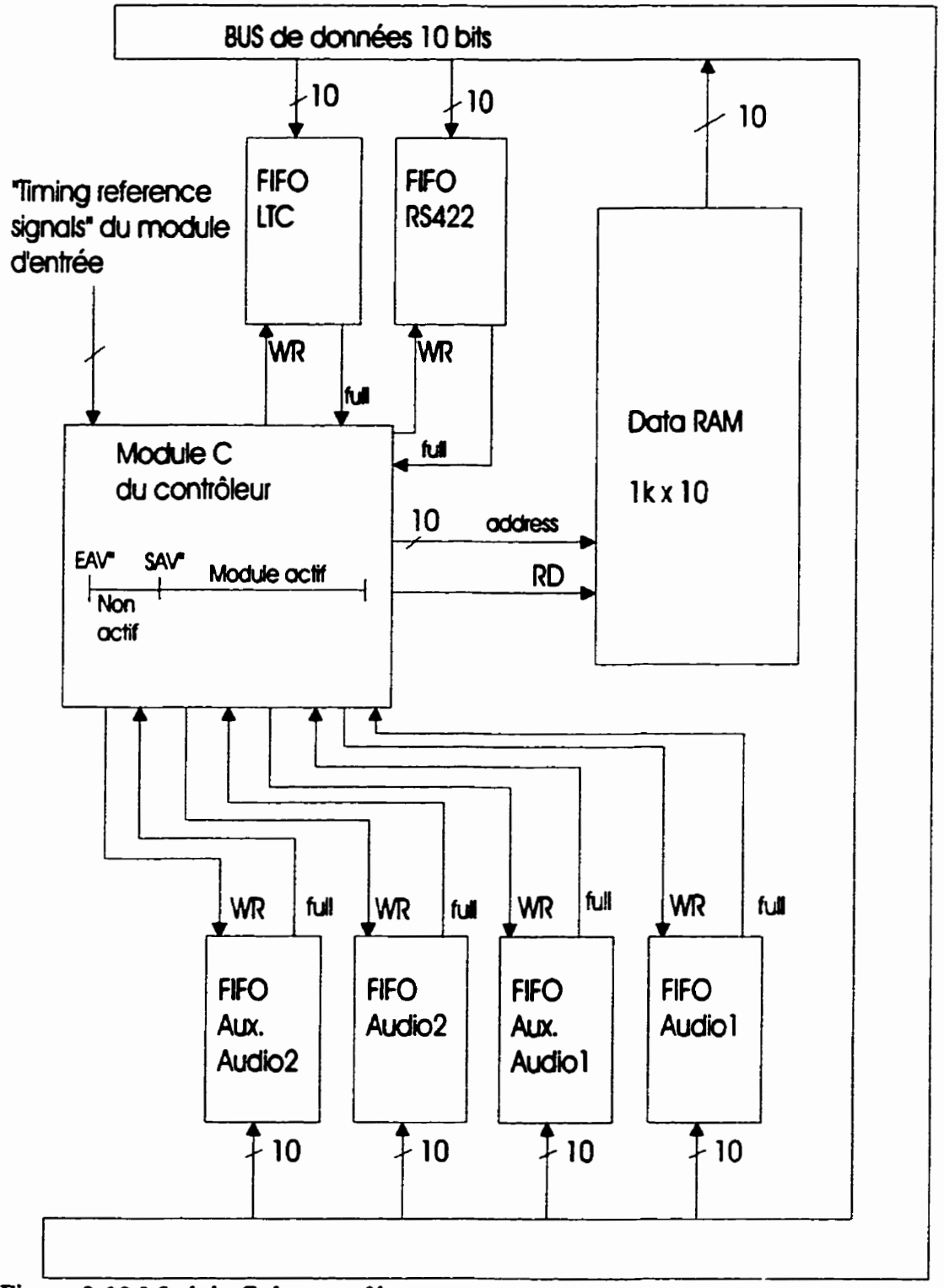


Figure 3.19 Module C du contrôleur

La figure 3.19 montre le schéma bloc du module C. Pour bien suivre la façon dont il fonctionne, nous allons décrire un exemple typique. Tout d'abord, établissons la situation d'un signal vidéo de DTV. Donc, les blocs de données LTC et RS422 traitent respectivement des données LTC et RS422. Nous sommes évidemment dans le mode DEMUX et le sous mode normal. On retrouve des paquets ancillaires auxiliaires audio marqués du DID correspondant au groupe audio traité dans le signal vidéo d'entrée, ce qui signifie que les échantillons audio sont codés sur 24 bits.

Nous atteignons la séquence SAV, ce qui fait que le module B a préalablement inscrit les UDWs des différents paquets ancillaires audio, auxiliaires audio, LTC et RS422 qu'il a trouvé dans l'espace HANC de cette ligne. Mentionnons que le processeur fonctionne depuis un bon moment et donc que dans la Data RAM, à ce moment, on retrouve des UDWs de paquets ancillaires provenant d'autres lignes précédentes.

La première tâche qu'effectue le module C en entrant dans la région « active picture » est de s'occuper des mots UDWs audio 1. Ainsi, le module C transfère les UDWs de la RAM de données dans le FIFO audio 1 en deux cycles pour chacun (un pour mettre le UDW sur le bus de données, un pour inscrire le mot dans le FIFO). Nous avons représenté un signal « full » pour chaque FIFO, sur la figure 3.19. En réalité, il existe plusieurs signaux sur le statu du FIFO. Le module C écrit les UDWs dans le FIFO jusqu'à ce qu'il n'y en ait plus dans la RAM de données ou jusqu'à ce qu'il ait atteint un seuil critique du FIFO. Ici, le module C fait la même chose que le module B, i.e. il s'assure que le nombre de mots écrits dans le FIFO audio est un multiple de trois. Un traitement similaire est appliqué pour les UDWs audio d'un signal vidéo HDTV. Le seuil critique correspond à un signal qui est actif lorsqu'il y a moins de trois espaces libres disponibles dans le FIFO (quatre pour un signal HDTV, voir chapitre 1).

Ici, une parenthèse sur la profondeur des FIFOs s'impose. On se rappelle que pour un signal vidéo DTV, aucun paquet audio n'est supposé être présent dans l'espace HANC

d'une ligne de transition de champ ou de la ligne suivante pour un signal DTV. Dans le cas d'un signal HDTV, seule la ligne de transition de champ ne doit pas contenir de paquet ancillaire audio. Également, les données audio doivent être transmises de manière continue vers le monde extérieur. Nous n'entrerons pas ici dans les détails, car il suffit de dire ici que le FIFO doit contenir un volume suffisant de mots pour fournir la machine bit-sérielle audio pendant environ trois lignes vidéo. Le calcul s'est fait en considérant la fréquence de transmission des données audio ainsi que toutes les fréquences des différents formats vidéo traités par le processeur. La profondeur a évidemment été déterminée par le besoin le plus grand.

Revenons maintenant à l'exemple, si on atteint le seuil critique et qu'il reste des UDWs audio1 dans la Data RAM, cela signifie qu'il y avait trop de UDWs dans le signal vidéo d'entrée. En effet, la distribution d'échantillons audio dans un signal vidéo doit respecter un nombre précis d'échantillons par trame vidéo pour avoir un synchronisme entre l'image et le son. Dans le cas du seuil critique atteint, il y a peut-être eu des erreurs sur les mots DC des paquets ou tout simplement trop de paquets audio de présent. Quoi qu'il en soit, cette situation en est une d'asynchronisme en surplus. L'asynchronisme en moins sera décrit dans la sous-section suivante. Lorsque cette situation survient, le module C lit le reste de la Data RAM (le pointeur de lecture incrémente), mais les UDWs ne sont pas écris dans le FIFO audio 1. Les UDWs excédentaires sont tout simplement perdus. Suivant les mots audio 1, un traitement similaire est appliqué pour les UDWs auxiliaires audio 1. Ces paquets sont complémentaires des paquets audio et le traitement des paquets ancillaires auxiliaires audio ne peut être fait indépendamment de celui des paquets audio. Ensuite, c'est le tour des UDWs audio 2 et auxiliaires audio 2. Un traitement identique à l'audio 1 est effectué à l'audio 2. Seuls les FIFO de destination changent. Les données LTC et RS422 sont ensuite traitées. Pour ces types de données, la transmission est discontinue, i.e. on transmet des données quand il y en a de présentes dans le signal vidéo d'entrée et aucune règle précise ne gère la fréquence d'insertion de ces paquets. Dans ce cas, on se

base sur le signal « full » pour suspendre l'écriture de UDWs dans les FIFO en question. Contrairement aux données audio, l'atteinte du seuil plein des FIFO ne provoque pas la suppression des données restantes dans la RAM de données. Ces données seront conservées dans la Data RAM et transférées dans leur FIFO respectif lors de l'active picture de la prochaine ligne vidéo.

Encore une fois, les modules B et C du contrôleur sont implantés en machines à états finies dédiées. Leur traitement est efficace, répétitif et les quelques paramètres de ces modules permettent une flexibilité. Cette implantation est simple et permet d'occuper une surface réduite.

3.10.5 Le module d'asynchronisme audio

Nous avons vu dans la sous-section précédente le traitement effectué par le module C sur les données audio lorsque ces dernières, pour quelques raisons que se soit, sont trop nombreuses dans le signal vidéo d'entrée. Nous avons également dit que les données audio, en mode DEMUX, doivent être transmises de façon continue au monde extérieur. Le module présenté ici est le module chargé de voir à ce qu'il y ait toujours un nombre suffisant de UDWs dans les FIFOs audio en mode DEMUX, pour que les machines bit-sérielles puissent transmettre les données sans être interrompues.

Nous avons également vu dans le chapitre 1 qu'il y avait quatre groupes audio qui pouvaient se retrouver dans un signal vidéo et que toutes les combinaisons sont possibles, y compris celle où aucun paquet audio n'est présent dans le signal vidéo d'entrée.

La figure 3.20 montre le schéma bloc du module d'asynchronisme audio.

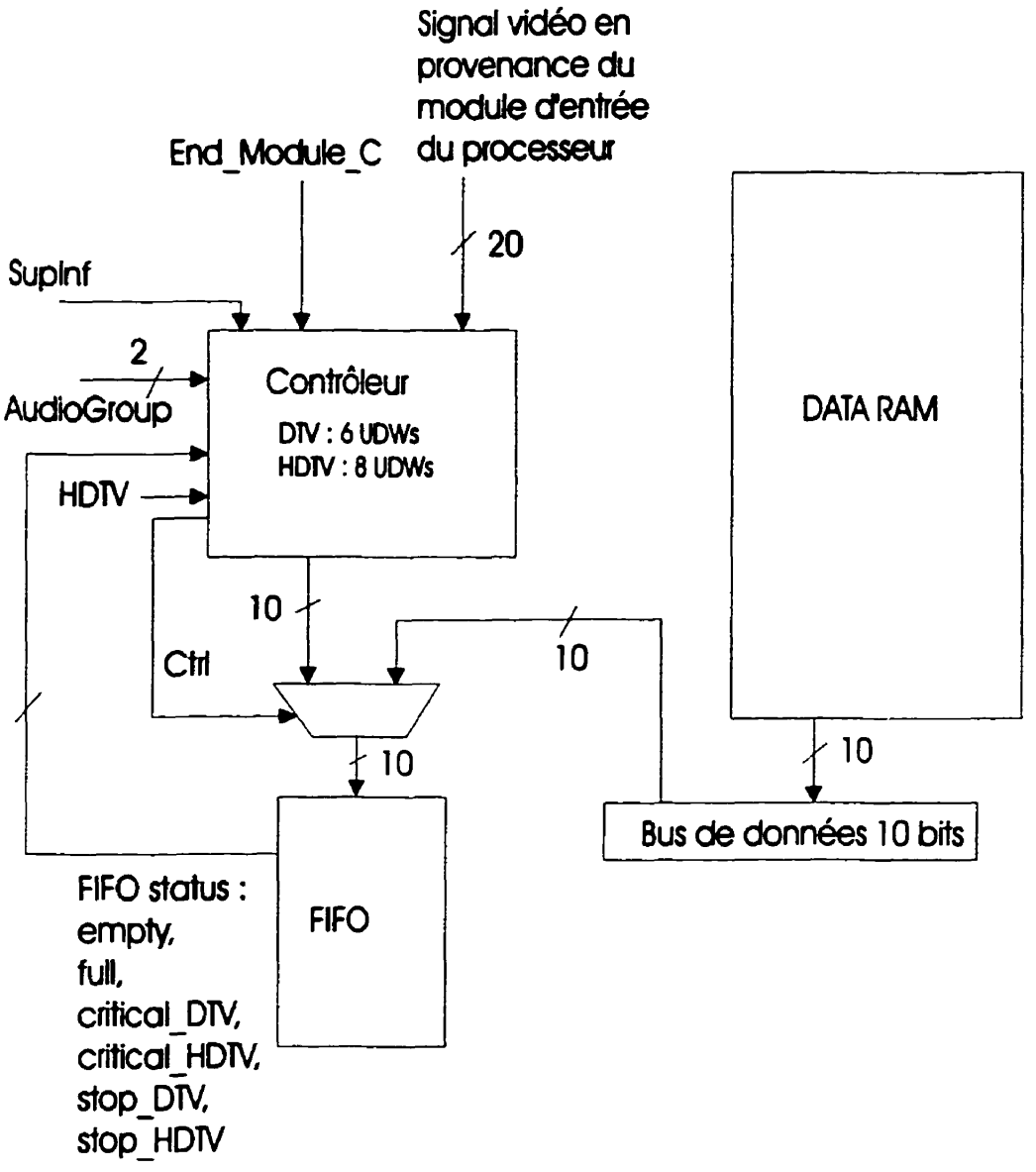


Figure 3.20 Module d'asynchronisme du contrôleur

La fonctionnalité de ce module se décompose en deux. Tout d'abord, dépendamment du groupe audio sélectionné et du format vidéo en présence, le module garde une copie (dans des registres) des UDWs qui forment deux échantillons audio et qui sont présents dans les paquets ancillaires audio de l'espace HANC. En format DTV, un échantillon audio est codé sur trois mots de 10 bits. Un échantillon du canal 1 d'une paire AES est

codé sur trois mots et un échantillon du canal 2 de cette même paire est également codé sur trois mots. On mémorise donc six mots. En format vidéo HDTV, les mêmes canaux de la même paire AES sont codés sur quatre mots chacun. Dans ce cas, huit mots sont mémorisés. Il est à noter ici qu'une partie du module d'asynchronisme s'occupe des UDWs reliés au bloc de données audio 1 alors qu'une autre partie s'occupe des UDWs reliés au bloc audio 2. En effet, on se rappelle que peu importe le format vidéo en présence, un paquet ancillaire audio contient des UDWs d'une paire AES 1 et des UDWs d'une paire AES 2. En tout, le module mémorise douze ou seize mots. Notons également que le même traitement est effectué pour les blocs RS422 et LTC lorsque ces derniers traitent des données audio en format vidéo HDTV.

Dans les paquets ancillaires audio d'un format vidéo HDTV, il n'y a que quatre échantillons. Ces quatre échantillons sont copiés et chaque fois qu'on rencontre un nouveau paquet, on remplace les valeurs de ces échantillons par les nouvelles contenues dans le paquet pour s'assurer d'avoir les données les plus récentes. Dans un paquet ancillaire audio d'un format vidéo DTV, il y a normalement 12 ou 16 échantillons. Un peu comme le module C, le module d'asynchronisme prend une copie du premier échantillon du paquet, s'assure qu'il est suivi d'un autre échantillon complet (opération de soustraction sur le DC), puis refait le même manège jusqu'à la fin du paquet audio.

La deuxième partie de la fonctionnalité de ce module s'opère à la suite de celle du module C. On a déjà dit qu'il y a plusieurs signaux rattachés aux différents FIFO. Le rôle du module est de s'assurer à ce que les FIFO audio contiennent toujours un nombre suffisamment élevé de mots pour permettre aux machines bit-sérielles de transmettre les données pendant le temps d'une ligne vidéo complète plus une marge de sécurité. Le seuil a été déterminé par le temps de ligne le plus lent, temps de ligne dépendant du format vidéo en présence. Lorsque le module C a fini de transférer les UDWs de la Data RAM aux différents FIFO audio et que le seuil minimum n'est pas respecté, le module d'asynchronisme écrit dans les différents FIFO les copies des derniers échantillons audio

qu'il a mémorisés, jusqu'à ce que les seuils minimaux de ces FIFO soient atteints. On peut voir que le traitement se fait lors de chaque portion active picture d'une ligne vidéo, au besoin.

Une question qu'on peut se poser est : qu'arrive-t'il si on ne reçoit jamais de paquets ancillaires audio ayant le groupe audio identique à celui sélectionné par l'usager? Lors du démarrage du processeur, les copies des échantillons audio de ce module sont initialisées avec des zéros. Le processeur transmettra donc des échantillons audio ne contenant que des zéros ce qui est généralement appelé « audio black » dans le jargon de la télévision.

3.10.6 L'interface I²C

La méthode de communication des différents paramètres du processeur qui a été retenue est un port sériel de communication tel que définit par la compagnie Philips : l'interface I²C. Dans cette sous-section, nous présentons les généralités de ce genre d'interface. L'annexe C contient des informations détaillées sur le protocole de communication I²C ainsi que la banque de registres internes du processeur.

Un protocole de communication I²C est défini sur deux ports d'entrée/sortie : un port de données et un port d'horloge. Ce protocole s'établit entre un ou plusieurs maître(s) et un ou plusieurs esclave(s). Le master est un micro contrôleur qui fournit l'horloge nécessaire à la transmission de données sérielles. L'esclave est un micro contrôleur qui ne génère pas sa propre horloge pour communiquer avec les autres micro contrôleurs. Le processeur à données ancillaires agit toujours en mode esclave. Comme on peut le voir en annexe C, un protocole de communication I²C se fait à partir d'adresses. Chaque micro contrôleur du réseau possède sa propre adresse. Les adresses sont codées sur 7 ou 10 bits, dépendamment du protocole retenu. Une procédure d'écriture et une procédure de lecture forment le protocole. Ici, ajoutons que ce qui est défini ci-haut n'est qu'une

partie du protocole. S'y ajoute, une adresse de communication à tous ainsi que d'autres fonctionnalités que nous ne traiterons pas.

L'utilisation d'un tel type de protocole se justifie par la nécessité de communiquer au monde extérieur des renseignements sur les différents paramètres du processeur. Ces renseignements se trouvent dans les registres internes du processeur. De plus, il est essentiel en cours de traitement, d'offrir la possibilité à l'usager de modifier certains paramètres pour appliquer le traitement désiré au signal vidéo.

Encore une fois, l'annexe C contient des informations décrivant entre autres, certains registres internes du processeur. Nous y avons montré seulement ceux qui sont accessibles par l'interface I^2C. Mentionnons seulement ici, que d'autres registres contenant entre autres les valeurs des DID et DC des paquets dont on désire faire le traitement font partie de la banque de registres centrale du processeur. Comme on peut le voir dans l'annexe, certains registres sont accessibles en lecture seulement alors que d'autres sont accessibles en lecture et écriture. Ainsi, les registres de sélection de groupe audio à traiter, de sous mode de fonctionnement (aussi appelé testmode), de format vidéo en présence et de non-retransmission de certains types de paquets (registre « delete ») sont parmi les plus importants. En effet, avec l'information transmise sur le contenu du signal vidéo d'entrée (tel que la présence des groupes audio dans l'espace HANC) offre plusieurs possibilités à l'usager qui peut traiter les signaux vidéo selon ses besoins.

Ici se termine la description du contrôleur du processeur. Évidemment, les quelques modules décrits sont les plus importants, mais ne constituent pas la totalité du contrôleur. Ainsi, certains autres modules sont nécessaires pour la programmation des mémoires RAM associées aux blocs de données du processeur, pour le traitement des sous-modes de fonctionnement « colour bar » et « tones » ainsi que pour le démarrage

progressif de tous les modules du processeur lors d'une mise en marche ou d'un changement de sous-mode de fonctionnement. Nous devons dire également que les modules du contrôleur présentés dans cette section l'ont été d'un assez haut niveau. Pour décrire le contrôleur sous toutes ses ramifications, il aurait fallu élargir de beaucoup le nombre de pages de cet ouvrage. Même le code VHDL dans lequel il a été décrit ne saurait être ajouté en annexe sans ajouter un nombre de pages considérable à l'ouvrage. C'est pourquoi nous avons choisi de présenter dans cette section une description sommaire, mais qui donne une bonne idée du fonctionnement des principaux modules formant le contrôleur du processeur à données ancillaires.

CHAPITRE 4

LES RÉSULTATS OBTENUS AVEC LE PROCESSEUR À DONNÉES ANCILLAIRES INTÉGRÉ

Dans ce chapitre, nous allons discuter des résultats obtenus avec la nouvelle architecture du processeur à données ancillaires qui a été traitée tout le long de cet ouvrage. Le chapitre se divise en trois. Nous traiterons premièrement des résultats de simulation obtenus avec la nouvelle architecture. Deuxièmement, il sera question d'une brève analyse de complexité du processeur en nombre de portes logiques. Troisièmement, nous traiterons de quelques point de l'architecture qui auraient pu être réalisés autrement.

4.1 Les résultats de simulation

On l'a vu tout au long de l'ouvrage, le processeur à données ancillaires est une architecture complexe. Cette architecture a été définie en langage VHDL pour pouvoir être simulée et synthétiser au niveau portes logiques. Une centaine de fichiers VHDL dont certains ont deux milles lignes, ont servi à définir l'architecture en entier. Au total, environ 30 000 lignes de VHDL ont été écrites pour définir l'architecture du processeur (sans compter quelques 50 000 lignes VHDL nécessaires pour les tests fonctionnels, bancs d'essai « test benchs »). La façon dont les simulations fonctionnelles ont été réalisées est la suivante. Tous les fichiers de l'architecture ont été simulés individuellement à l'aide de bancs d'essai. Ces fichiers ont été intégrés au sein d'une hiérarchie où encore une fois ils ont été simulés, mais cette fois en interaction avec d'autres fichiers. Plusieurs niveaux de hiérarchie ont été définis, dans le but de faciliter la détection et le traitement des erreurs. Il serait ici fastidieux de présenter ces résultats de simulation en détails tels que réalisés avec l'outil VSS de Synopsys. Même le report

de ces résultats en annexe est farfelu, compte tenu du nombre de résultats de simulation et de la difficulté de les analyser pour le lecteur. C'est pourquoi nous préférons résumer ici les principaux résultats.

Le résultat le plus important est la latence de traitement du processeur intégré. On se rappelle qu'il était une des raisons majeures à la définition d'une nouvelle architecture du processeur, la première étant trop lente. Autant en mode MUX qu'en mode DEMUX, la plus longue latence de traitement est de 78 * $1/(27*10^6)$ = 2,9 us. Cette latance provient des 4 mots de retard destiné à déterminer le paramètre « SupInf » du module d'entrée du processeur ainsi que des 74 mots que l'on accumule dans la RAM vidéo. Comme nous pouvons le voir, c'est une nette amélioration en comparaison des 40 us de la première version. Outre ce résultat, l'architecture du processeur à données ancillaires réalise très efficacement les différentes tâches qui lui sont attribuées dont celle de traiter les signaux vidéo HDTV. En effet, le processeur assemble correctement les différents paquets ancillaires en fonction des données qu'il reçoit. Au cours des simulations, nous avons scruté à la loupe les différents paquets ancillaires fabriqués par le processeur. Tous les UDWs sont fabriqués comme indiqué dans les normes SMPTE, tous les mots DID, DBN et DC contiennent les valeurs qu'ils devraient avoir en fonction du paquet dans lequel ils se trouvent. De même, tous les mots CS (et ECC lorsque nécessaire) contiennent le résultat attendu selon la définition SMPTE. Également, tous ces paquets sont insérés aux bonnes lignes, lorsque dans le bon sous-mode de fonctionnement et selon la stratégie de traitement adoptée. Le processeur répare aussi certaines erreurs qui peuvent se retrouver dans un signal vidéo. Ainsi, il est capable de concaténer au début de l'espace HANC des paquets qui ne l'étaient pas (jusqu'à une certaine limite).

En mode DEMUX, le processeur ajoute les bits de préambule et de parité nécessaires à la transmission des données selon les différentes normes. Pour le cas des données audio, le processeur traite adéquatement les cas d'asynchronisme des données audio par rapport

au signal vidéo lorsque ces dernières sont trop peu ou trop nombreuses dans le signal vidéo reçu. De plus, le processeur est doté d'un mécanisme de réduction des erreurs efficace sur le mot DC de tout paquet ancillaire audio.

Finalement, ajoutons que les différents mécanismes de programmation initiale du processeur à l'aide d'une EEPROM ainsi que ceux pour la communication des différents paramètres du processeur à l'aide de l'interface I^2C ont été implantés et validés par simulation.

Ceci dit, un bon nombre de simulations après synthèse ont également été réalisées sur certains modules du processeur. Ces simulations donnaient également les résultats escomptés. La latence de traitement totale du circuit, bien que simulée au niveau fonctionnel, est la même au niveau portes logiques, puisqu'elle est définie en tenant compte de quatre étages de registre à décalage à franchir dans le module d'entrée, ainsi que des 74 étages de la RAM vidéo. En ce sens, après la synthèse du circuit, aucun élément séquentiel (bascules) ne saurait être inséré dans ce chemin critique et c'est pourquoi nous pouvons déjà chiffrer ce chemin à l'étape de simulation fonctionnelle. Nous supposons évidemment que les délais dans la logique combinatoire et les interconnexions permettront de rencontrer la plus haute fréquence d'opération ciblée avec une technologie CMOS 0,35 um.

4.2 Analyse de complexité

Plusieurs modules du processeur ont été synthétisés à l'aide de l'outil Synopsys. Le résultat de l'estimation de la complexité du circuit nous donne 130 000 portes logiques. De ce nombre, nous estimons les parties logiques et les modules de mémoires RAM à 90 000 et 40 000 portes respectivement. La porte de référence pour mesurer la complexité est une porte « nand » à deux entrées. De même, une cellule de mémoire RAM à 6

transistors a été considérée et l'ensemble des cellules a été ramené à une porte « nand » par une simple règle de trois.

4.3 Discussion

Un point important de la conception du processeur à données ancillaires intégré est qu'il est flexible. La flexibilité d'un circuit intégré est grandement appréciée, car par définition un « ASIC » (Application Specific Integrated Circuit) est un circuit dédié à réaliser une tâche spécifique. Comme nous avons vu, la machine bit-sérielle transformant les données d'un format à un autre est très flexible en raison de l'architecture la définissant De plus, le contrôleur réalise des actions selon une série de paramètres programmables contenus dans des registres internes. Ces registres internes sont programmés soit à l'initialisation du circuit à l'aide de l 'EEPROM, soit pendant le fonctionnement, à l'aide de l'interface I2C. Les registres internes sont prévus pour assurer la plus grande flexibilité possible au contrôleur. Ainsi, les valeurs de DID, de DBN, de DC de chaque paquet ancillaire que l'usager désire traiter sont inscrites dans des registres et sont modifiables. La flexibilité, on ne saurait trop le rappeler, est très importante en sens qu'elle permettra au processeur à données ancillaires intégré de traiter des données qui ne sont pas encore définies dans des standards.

Un autre point de discussion est que bien qu'elle soit efficace, l'architecture du processeur aurait pu être réalisée de différentes façons. Un des points qu'il aurait été possible de réaliser autrement est le contrôleur (FSM) de la machine bit-sérielle. En effet, recevant/transmettant des bits de données à une fréquence différente de celle des données vidéo, il aurait été possible de la faire fonctionner à la fréquence d'émission ou de réception des données sérielles. Il aurait alors fallu pipeliner la machine bit-sérielle de façon à lui faire réaliser le traitement voulu. Bien que cette réalisation ait peut-être pu faire gagner de l'espace, le problème de synchronisation entre deux horloges se serait présenté aussi. En effet, le FIFO étant l'interface entre la machine bit-sérielle

fonctionnant à la cadence de l'horloge audio (ou autre type de donnée) et le contrôleur global du processeur fonctionnant à la cadence de l'horloge vidéo, le problème se serait simplement déplacé.

Un dernier point de discussion, est la présence de la RAM de données dans le processeur. Cette question a uniquement trait à réduire l'aire du circuit. Sans la RAM de données, les divers UDWs auraient tous été contenus dans les différents FIFO attachés aux blocs de données du processeur. La question est de savoir si on est gagnant en enlevant la RAM de données, en agrandissant la taille de tous les FIFO d'un certain nombre de mots et en ajoutant de la logique combinatoire pour traiter adéquatement l'insertion des nouveaux paquets ancillaires fabriqués par le processeur. Ceci est possible, mais l'assemblage en temps réel des paquets sans la RAM de données aurait mis beaucoup de pression sur le design du contrôleur.

Retenons simplement de cette sous-section qu'il aurait été possible de réaliser l'architecture du processeur à données ancillaires autrement pour des raisons de surface , bien que la solution adoptée soit très efficace

CONCLUSION

Dans ce mémoire, nous avons d'abord introduit les différentes normes de télévision numérique ainsi que certaines normes pour des données telles que les données audio, LTC (linear time code) et RS422. Toutes ces notions étaient essentielles au lecteur pour la suite de l'ouvrage. Nous avons ensuite examiné une version carte d'un processeur à données ancillaires réalisée par la compagnie Miranda Technologies Inc. Cette société désirait réaliser une seconde version de ce processeur beaucoup plus performante et plus compacte. Après avoir présenté cette première version, nous avons établi de nouvelles spécifications fonctionnelles. L'architecture de la nouvelle version du processeur a ensuite été présentée en détail.

Le premier chapitre de ce mémoire présentait l'environnement dans lequel s'inscrit un processeur à données ancillaires. Tout d'abord, les signaux de télévision numérique ont été présentés, permettant au lecteur de se familiariser avec les différentes notions que sont les paquets ancillaires, les trames vidéo, les champs vidéo, les lignes vidéo, les séquences de synchronisation SAV (Start of Active Video) et EAV (End of Active Video), la région de vidéo active, l'espace HANC (Horizontal ANCillary), les différents formats de télévision numérique (DTV, HDTV), etc. Ensuite, le format AES/EBU, pour les données audio numériques a été présenté en détail et ce pour la bonne raison que ces données sont critiques au processeur. Nous avons vu en détail comment transformer les données audio AES/EBU sérielles en format ancillaire et vice versa. Finalement, les types de données LTC et RS422 ont également été présentées. Ce chapitre fait un bref survol de la télévision numérique et des types de données traitées par le processeur. En ce sens, il atteint l'objectif d'introduire le lecteur à la télévision numérique.

Le deuxième chapitre présentait un bref survol de la fonctionnalité de la première version du processeur à données ancillaires, dont la fonctionnalité de base est d'insérer/extraire dans/d'un signal vidéo numérique des données ancillaires. Les deux

modes de fonctionnement (MUX, DEMUX) du processeur ainsi que les différents sous-modes propres à chacun mode ont été présentés en terme de ce que le processeur devait faire dans chaque cas. Puis, nous avons ensuite présenté les nouvelles fonctionnalités du processeur qui devait, entre autres, être capable de traiter un signal vidéo HDTV. Dans ce second chapitre, le lecteur a pu comprendre la motivation ayant conduit à une version intégrée d'un processeur à données ancillaires intégré. Cet objectif a été atteint en décrivant brièvement la version carte du processeur à données ancillaires ainsi qu'en énonçant les nouvelles fonctionnalités que devait réaliser la version intégrée. La comparaison des fonctionnalités de la version carte et intégrée mettait clairement en lumière le besoin de la version intégrée.

Le troisième chapitre présentait en détail l'architecture de la deuxième version du processeur à données ancillaires qui répondait aux différentes fonctionnalités énoncées dans le chapitre 2. Ainsi, pour des raisons d'espace, de rapidité d'exécution et de besoin de composants analogiques, l'architecture a été conçue pour être implantée dans un ASIC (Application Specific Integrated Circuit).

Nous avons tout d'abord traité du module d'entrée du processeur dont la fonctionnalité était de faire la détection automatique du format vidéo en présence ainsi que la détection des signaux de synchronisation destinés à tout le reste du circuit. Nous avons ensuite traité de la nécessité d'inclure une mémoire RAM (la RAM vidéo) dans le processeur. Nous avons vu dans cette section que cette RAM était une bonne solution pour le multiplexage efficace des anciens paquets ancillaires avec les nouveaux fabriqués par le processeur, en mode MUX. Par la suite, nous avons traité de la présence d'un autre bloc de mémoire RAM (RAM de données) à l'intérieur du processeur. Nous avons vu que cette mémoire contenait les différents types de paquets ancillaires fabriqués par le processeur et prêts à être insérés, en mode MUX, alors qu'elle contenait les différents UDWs (User Data Words) des paquets qu'on voulait traiter en mode DEMUX. Après un bref survol des ROM spécialisées nécessaires au processeur pour les sous-modes de

fonctionnement « tones » et « colour bar », nous avons présenté le module de sortie du processeur dont la fonctionnalité est le calcul et l'insertion des mots CS (Check Sum) et ECC (Error Correcting Codes) des paquets ancillaires insérés et fabriqués par le processeur. Un des blocs extrêmement important du processeur a ensuite été traité : la machine bit-sérielle. Dans la sous-section la décrivant, nous avons vu que sa fonctionnalité est d'assembler les mots UDWs, formant les paquets ancillaires en mode MUX, à partir des données sérielles et de sérialiser ces mêmes UDWs vers le monde extérieur, en mode DEMUX. Un des points importants de cette machine est qu'elle contient une mémoire RAM permettant de programmer le séquencement des instructions que l'on veut exécuter sur les données. Nous avons vu que cette machine est grandement responsable de la flexibilité offerte par le processeur. En effet, grâce à cette machine, nous avons vu que le processeur sera éventuellement capable de traiter des données qui ne sont, pour le moment, pas encore définies dans des normes existantes. Suivant la description de cette machine, nous avons présenté la structure du bus de données principal du processeur, ainsi que le chemin de données des quatre blocs de données présents dans le processeur.

Le chapitre 3 s'achève avec la description détaillée du morceau le plus important du processeur : le contrôleur. Nous avons vu dans cette sous-section la partition du contrôleur en plusieurs modules. Le module A dont la fonctionnalité est d'inscrire dans la RAM de données les nouveaux paquets ancillaires formés par le processeur, a d'abord été présenté. Ensuite, le module D, dont la fonctionnalité est le multiplexage de l'espace HANC de chaque ligne vidéo, a été décrit. Nous avons ensuite traité des modules B et C qui ont pour fonction de sélectionner et transférer via les différents FIFOS, aux machines bit-sérielles, les UDWs des différents paquets ancillaires dont on désire faire le traitement. Dans ces modules, on a vu l'accent mis sur les paquets audio qui sont critiques au processeur. Le module chargé de gérer l'asynchronisme audio a ensuite été présenté. Nous avons vu que les données audio en format AES/EBU doivent être transmises continuellement en mode DEMUX et ne pouvaient souffrir d'un manque de

données à aucun moment. La stratégie de copier le dernier échantillon audio retrouvé dans le signal vidéo d'entrée pour palier à cette situation a été décrite dans cette sous-section. Finalement, nous avons présenté en toute dernière instance de ce chapitre, le module de communication basé sur le mécanisme I^2C (Inter Integrated Circuit). Nous y avons présenté les différents registres internes accessibles par le monde extérieur ainsi que la façon de modifier ceux qui sont accessibles en écriture dans le but de pouvoir exécuter le traitement vidéo le plus adéquat possible selon les besoins de l'usager.

Ce chapitre trois est très important. Il atteint tout d'abord l'objectif de présenter en détail au lecteur, la version intégrée du processeur à données ancillaires. Ceci est accompli en décrivant rigoureusement les modules clés définissant son architecture ainsi que les différentes interactions entre eux. Également dans ce chapitre, l'objectif de présentation de quelques contributions de ce mémoire est atteint. Des descriptions détaillées de la latence de traitement du processeur, de la machine bit-sérielle et de l'élément flexibilité du processeur atteignent cet objectif. La description détaillée de l'architecture du processeur explique également très bien la façon dont les objectifs de performance, spécialement au niveau latence de traitement, ont été atteints. La RAM vidéo et le module D du contrôleur sont conçus de manière à ce que la latence de traitement minimale soit obtenue en fonction de la tâche à réaliser. L'architecture de la machine bit-sérielle (séquence d'instructions programmable) ainsi que les registres internes du contrôleur (accessible par l'EEPROM ou l'interface I^2C) assurent la flexibilité du processeur.

Le quatrième chapitre du mémoire présentait les différents résultats obtenus avec l'architecture de la deuxième version du processeur. Après y avoir résumé les différentes simulations exécutées sur le processeur, nous avons décrit une analyse de complexité estimant le nombre de portes logiques du circuit formant le processeur à données ancillaires. Ce chapitre se terminait par une brève discussion sur la façon de réaliser la machine bit-sérielle ainsi que la présence de la mémoire Data RAM. Le

chapitre 4 est en sorte une comparaison entre les objectifs de performance et les résultats obtenus, prouvant que ces résultats ont été atteints.

Ce mémoire s'inscrit dans la liste des travaux destinés à l'implantation d'une fonctionnalité donnée. Plusieurs façons sont à considérer lors d'une implantation. Les technologies FPGA et ASIC permettent des implantations différentes. L'avenir rapproché sera très intéressant en matière de choix d'implantation. Les technologies FPGA sont de plus en plus rapides et permettent des circuits de complexité de plus en plus grande. Les technologies ASIC permettent également des vitesses et des niveaux de complexité de plus en plus grands en raison des tailles de transistor qui diminuent sans cesse. De plus, certains fabricants permette l'utilisation de leur processeur dans un circuit ASIC de plus grande taille, créant des « systèmes intégrés » (System on chip). Cette façon de faire permet à l'usager d'utiliser le processeur en tant que contrôleur flexible du circuit intégré, ce qui lui donne l'opportunité de se concentrer sur la conception du chemin de données (ou autres contrôleurs secondaires) entourant le processeur. N'oublions pas également les approches matérielle/logicielle « classiques » (processeur + éléments matériels qui ne sont pas intégrés dans un même composant physique) qui continuent toujours de se raffiner. Éventuellement, les travaux d'implantation conduiront au fait qu'une technologie d'implantation est meilleure que les autres ou de façon plus réaliste, que considérant les coûts, les performances au niveau rapidité, les niveaux d'intégration, etc., une technologie d'implantation est plus appropriée pour tel type de fonctionnalité alors qu'une autre technologie l'est pour un autre type d'application. Le processeur à données ancillaires intégré a déjà été réalisé en approche matérielle/logicielle classique et en technologie ASIC. Il sera sûrement réalisé de façon FPGA ou « système intégré » bientôt. Tout ceci pour dire qu'il existe plusieurs choix d'implantation et que la décision se prend à la suite de l'analyse de ce que l'on désire réaliser.

En conclusion, ce mémoire a décrit en détail l'architecture du processeur à données ancillaires intégré qui représente la deuxième version du produit original développé par la compagnie Miranda Technologies Inc. Comme nous l'avons vu tout au long de l'ouvrage, ce processeur est destiné au traitement des signaux de télévision numérique qui feront bientôt partie de tout téléviseur sur la planète. Une deuxième version était nécessaire, la première possédant une latence de traitement beaucoup trop élevé ainsi qu'une incapacité de traitement des signaux HDTV (High Definition TeleVision). Avec la version intégrée du processeur, la latence de traitement est passée de 40 us à moins de 3 us. Également, le processeur à données ancillaires intégré a été conçu pour traiter les signaux HDTV à la fréquence de 74,25 MHz. En plus de répondre efficacement en tous points aux différentes fonctionnalités exigées au départ, il offre une flexibilité qui fait qu'il pourra éventuellement traiter des types de données encore non définis dans les différentes normes en matière de télévision, ce qui est un avantage des plus apprécié.

RÉFÉRENCES

AES3-1992. AES Recommended practice for digital audio engineering – Serial transmission format for two-channel linearly represented digital audio data, New York (NY), Audio Engineering Society, 1992.

ANSI/SMPTE 267M-1995. Bit-parallel digital interface – component video signal 4 :2 :2 16x9 aspect ratio, White Plains (NY), The Society of Motion Picture and Television Engineers, 1995.

ANSI/SMPTE 272M-1994. Formatting AES/EBU audio and auxiliary data into digital video ancillary data space, White Plains (NY), The Society of Motion and Television Engineers, 1994.

ANSI/SMPTE 291M-1996. Television – Ancillary data packet and space formatting, White Plains (NY), The Society of Motion and Television Engineers, 1996.

ANSI/SMPTE 292M-1996. Bit-serial digital interface for high-definition television systems, White Plains (NY), The Society of Motion and Television Engineers, 1996.

ANSI/SMPTE 299M-1997. 24-bit digital audio format for HDTV bit-serial interface, White Plains (NY), The Society of Motion and Television Engineers, 1997.

ASHENDEN, P. The designer's guide to VHDL, San Francisco (CA), Morgan Kaufmann Publishers, 1996.

SMPTE RP 165-1994. Error detection checkwords and status flags for use in bit-serial digital interfaces for television, White Plains (NY), The Society of Motion and Television Engineers, 1994.

SMPTE RP 188, « Transmission of time and control code in the ancillary data space of a digital television data stream », SMPTE Journal, octobre1995, pp. 708-711.

SMPTE 12M, « Time and control code », SMPTE Journal, février 1995, pp. 100-108.

ANNEXE A

Détail des instructions de la machine bit-sérielle

Note : Les instructions sont codées sur 16 bits. La première rangée de chaque instruction est le numéro des bits dans le mot d'instruction. La deuxième rangée sont les différents champs de chaque instruction (0, 1, *). Chaque champ d'une instruction comportant le symbole * est défini sous les deux rangées définissant une instruction.

1. NOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0

Bit 11 : remise à zéro du bloc de parité numéro 1

2. NOP MUX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	*	*	*	*	*	*	*	*	*	*	*	0

Bits 11-9 : sélection de l'entrée du multiplexeur A du data path

Bit 8 : sélection de l'entrée du multiplexeur B du data path

Bit 7 : activation du bloc de parité numéro 1

Bit 6 : activation du signal d'écriture du registre à décalage A

Bit 5 : activation du signal d'écriture du registre à décalage B

Bit 4 : incrémentation du pointeur d'instruction

Bit 3 : saut d'adresse du pointeur d'instruction

Bit 2 : activation du bloc de parité numéro 2

Bit 1 : bit d'élaboration de l'instruction

3. MOVEBIT 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	*	*	*	*	*	*	*	*	*	*	0	0

Bits 11 à 9 : sélectionne l'entrée du multiplexeur A du data path

Bit 8 : sélectionne l'entrée du multiplexeur B du data path

Bit 7 : activation (enable) du bloc de parité numéro 1

Bit 6 : activation du signal d'écriture du registre à décalage A

Bit 5 : activation du signal d'écriture du registre à décalage B

Bit 4 : incrémentation du pointeur d'instruction

Bit 3 : saut d'adresse du pointeur d'instruction

Bit 2 : activation du bloc de parité numéro 2

4. MOVEBIT 2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	*	*	*	*	*	*	*	*	*	*	0	0

Bits 11 à 9 : sélectionne l'entrée du multiplexeur A du data path

Bit 8 : sélectionne l'entrée du multiplexeur B du data path

Bit 7 : activation (enable) du bloc de parité numéro 1

Bit 6 : activation du signal d'écriture du registre à décalage A

Bit 5 : activation du signal d'écriture du registre à décalage B

Bit 4 : incrémentation du pointeur d'instruction

Bit 3 : saut d'adresse du pointeur d'instruction

Bit 2 : activation du bloc de parité numéro 2

5. MOVEBIT 3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	*	*	0	0	0	0	0	0	0	0	0	0

Bit 11 : incrémentation du pointeur d'instruction

Bit 10 : saut d'adresse du pointeur d'instruction

6. MOVEBIT 4

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	*	*	*	*	*	*	0	0	0	0	0	0

Bits 11 à 9 : sélectionne l'entrée du multiplexeur A du data path

Bit 8 : activation (enable) du bloc de parité numéro 1

Bit 7 : incrémentation du pointeur d'instruction

Bit 6 : saut d'adresse du pointeur d'instruction

7. MOVEBIT 5

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	*	0	0	0	0	0	0	0	0	0	0	0

Bit 11 : sélectionne l'entrée du multiplexeur B du data path

8. MOVEBIT 6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*

Bits 11 à 8 : sélectionne l'entrée du multiplexeur C et du multiplexeur D du data path

Bits 7 à 5 : sélectionne l'entrée du multiplexeur A du data path

Bit 4 : incrémentation ou saut d'adresse du pointeur d'instruction

Bit 3 : sélectionne le prochain état du contrôleur de la machine bit-sérielle

Bit 2 : reset du bloc de parité numéro 1

Bit 1 : sélectionne un chemin précis et déterminé (une entrée du multiplexeur A et une entrée du multiplexeur B)

Bit 0 : activation du signal d'écriture du registre à décalage B

9. MOVEBIT 7

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	*	*	*	*	*	*	*	*	*	*	*	*

Bits 11 à 9 : sélectionne l'entrée du multiplexeur A du data path

Bit 8 : incrémentation ou saut d'adresse du pointeur d'instruction

Bits 7 à 4 : sélectionne l'entrée du multiplexeur C du data path

Bit 3 : sélectionne le prochain état du contrôleur de la machine bit-sérielle

Bit 2 : activation du signal d'écriture du FIFO principal

Bit 1 : activation du bloc de parité numéro 2

Bit 0 : activation du signal de lecture du FIFO principal

10. MOVEWORD 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

11. MOVEWORD 2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	*	*	0	0	0	0	0	0	0	0	0	0

Bit 11 : active le signal de lecture du FIFO principal

Bit 10 : active le signal de lecture du FIFO auxiliaire

12. EDGEDETECT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	*	0	0	0	0	0	0	0	0	0	0	0

Bit 11 : reset du bloc de parité numéro 1

Exemple commenté d'un segment de programme de la machine bit-sérielle

Dans cet exemple, nous allons voir comment assembler quelques bits de données AES/EBU tel que montré à la figure 1.4 b), en format ancillaire tel que montré au tableau 1.2. Cet exemple s'inscrit donc dans le mode MUX, où l'on traite des échantillons audio AES/EBU codés sur 20 bits. Le sous-mode normal est également de mise dans cet exemple. Notons également que le diagramme temporel d'arrivée des données audio au processeur est celui de la figure 3.8 a). Finalement, cet exemple prend en considération le bloc de traitement des données audio 1 du processeur.

Instruction #1 : NOP MUX, "1001000000010110"

Dans cette instruction, le processeur attend une transition montante sur le signal « fsync » (une transition montante sur sck, horloge d'arrivée des bits, arrive en même temps), indiquant le début d'un échantillon audio. Puisque le bit reçu est un bit de préambule qui ne sert pas au format ancillaire, la machine bit-sérielle ne fait rien avec ce bit et passe à l'instruction suivante.

Instruction #2 : MOVEBIT 2, "0010011011010000"

Dans cette instruction, la machine bit-sérielle attend (transition montante sur « sck ») le deuxième bit du format AES/EBU qui est également un bit de préambule ne servant pas au format ancillaire. Cette instruction permet d'aller inscrire le bit z qui est le bit numéro 0 du mot x du format ancillaire. Pour ce faire, l'entrée adéquate du multiplexeur A est sélectionnée, le bit est compté dans le calcul de parité et inscrit dans le registre à décalage A.

Instruction #3 : MOVEBIT 2, "00101000011010000"

Dans cette instruction, la machine bit-sérielle attend (transition montante sur « sck ») le troisième bit du format AES/EBU qui est également un bit de préambule ne servant pas au format ancillaire. Cette instruction permet d'aller inscrire le bit c0 qui est le bit numéro 1 du mot x du format ancillaire. Pour ce faire, l'entrée adéquate du multiplexeur A est sélectionnée, le bit est compté dans le calcul de parité et inscrit dans le registre à décalage A.

Instruction #4 : MOVEBIT 2, "00101011110100000"

Dans cette instruction, la machine bit-sérielle attend (transition montante sur « sck ») le quatrième bit du format AES/EBU qui est également un bit de préambule ne servant pas au format ancillaire. Cette instruction permet d'aller inscrire le bit c1 qui est le bit numéro 2 du mot x du format ancillaire. Pour ce faire, l'entrée adéquate du multiplexeur A est sélectionnée, le bit est compté dans le calcul de parité et inscrit dans le registre à décalage A.

Instructions #5 à #8 : NOP, "0000000000000000"

Dans ces instructions, la machine bit-sérielle reçoit les bits AUX du format AES/EBU qui, puisque les échantillons audio sont codés sur 20 bits, ne servent pas au format ancillaire. La machine bit-sérielle ne fait que les « laisser passer ».

Instructions #9 à #13 : MOVEBIT 2, "0010000011010000"

Dans ces instructions, la machine bit-sérielle attend (transition montante sur « sck ») les bits AES/EBU numéro 0,1,2,3 et 4. Cette instruction permet à chaque fois d'aller ajouter le bit qui arrive à ce qui est déjà assemblé du mot x du format ancillaire. Pour ce faire, l'entrée adéquate du multiplexeur A est sélectionnée, les bits sont comptés un à un dans le calcul de parité et inscrits dans le registre à décalage A.

Instruction #14 : MOVEBIT 4, "0110000110000000"

Cette instruction attend un nouveau bit de donnée AES/EBU (transition montante sur « SCK »). L'instruction inscrit le bit dans le registre à décalage A tout en le faisant compter dans le calcul de parité numéro 1. Ensuite, l'inverse du dernier bit reçu est également inscrit dans ce même registre à décalage. Finalement, le mot x du format ancillaire étant complet, il est inscrit dans le FIFO audio A du bloc de traitement des données audio 1.

ANNEXE B

Programmes des machines bit-sérielle

AES/EBU 20 bits, signal vidéo DTV en mode MUX

"10010000000010110"	9016h NOP MUX preamble 0
"00100110110100000"	26D0h MOVEBIT 2 preamble 1
"00101000110100000"	28D0h MOVEBIT 2 preamble 2
"00101011110100000"	2BD0h MOVEBIT 2 preamble 3
"00000000000000000"	0000h NOP aux 0
"00000000000000000"	0000h NOP aux 1
"00000000000000000"	0000h NOP aux 2
"00000000000000000"	0000h NOP aux 3
"00100000110100000"	20D0h MOVEBIT 2 audio 0
"00100000110100000"	20D0h MOVEBIT 2 audio 1
"00100000110100000"	20D0h MOVEBIT 2 audio 2
"00100000110100000"	20D0h MOVEBIT 2 audio 3
"00100000110100000"	20D0h MOVEBIT 2 audio 4
"01100001100000000"	6180h MOVEBIT 4 audio 5
"00100000110100000"	20D0h MOVEBIT 2 audio 6
"00100000110100000"	20D0h MOVEBIT 2 audio 7
"00100000110100000"	20D0h MOVEBIT 2 audio 8
"00100000110100000"	20D0h MOVEBIT 2 audio 9
"00100000110100000"	20D0h MOVEBIT 2 audio 10
"00100000110100000"	20D0h MOVEBIT 2 audio 11
"00100000110100000"	20D0h MOVEBIT 2 audio 12
"00100000110100000"	20D0h MOVEBIT 2 audio 13
"01100001100000000"	6180h MOVEBIT 4 audio 14
"00100000110100000"	20D0h MOVEBIT 2 audio 15

"0010000011010000"	20D0h MOVEBIT 2 audio 16
"0010000011010000"	20D0h MOVEBIT 2 audio 17
"0010000011010000"	20D0h MOVEBIT 2 audio 18
"0010000011010000"	20D0h MOVEBIT 2 audio 19
"0010000011010000"	20D0h MOVEBIT 2 bit V
"0010000011010000"	20D0h MOVEBIT 2 bit U
"0011100000000000"	3800h MOVEBIT 3 bit C
"0010000000001000"	2008h MOVEBIT 2 bit P

AES/EBU 24 bits, signal vidéo DTV en mode MUX

"1001000000010110"	9016h NOP MUX preamble 0
"0010011011010000"	26D0h MOVEBIT 2 preamble 1
"0010100011010000"	28D0h MOVEBIT 2 preamble 2
"0010101111010000"	2BD0h MOVEBIT 2 preamble 3
"0010000000110000"	2030h MOVEBIT 2 audio 0
"0010000000110000"	2030h MOVEBIT 2 audio 1
"0010000000110000"	2030h MOVEBIT 2 audio 2
"0111000000000000"	7000h MOVEBIT 5 audio 3
"0010000011010000"	20D0h MOVEBIT 2 audio 4
"0010000011010000"	20D0h MOVEBIT 2 audio 5
"0010000011010000"	20D0h MOVEBIT 2 audio 6
"0010000011010000"	20D0h MOVEBIT 2 audio 7
"0010000011010000"	20D0h MOVEBIT 2 audio 8
"0110000110000000"	6180h MOVEBIT 4 audio 9
"0010000011010000"	20D0h MOVEBIT 2 audio 10
"0010000011010000"	20D0h MOVEBIT 2 audio 11
"0010000011010000"	20D0h MOVEBIT 2 audio 12
"0010000011010000"	20D0h MOVEBIT 2 audio 13
"0010000011010000"	20D0h MOVEBIT 2 audio 14

"0010000011010000" 2000h MOVEBIT 2 audio 15
"0010000011010000" 2000h MOVEBIT 2 audio 16
"0010000011010000" 20D0h MOVEBIT 2 audio 17
"0110000110000000" 6180h MOVEBIT 4 audio 18
"0010000011010000" 20D0h MOVEBIT 2 audio 19
"0010000011010000" 20D0h MOVEBIT 2 audio 20
"0010000011010000" 20D0h MOVEBIT 2 audio 21
"0010000011010000" 20D0h MOVEBIT 2 audio 22
"0010000011010000" 20D0h MOVEBIT 2 audio 23
"0010000011010000" 20D0h MOVEBIT 2 bit V
"0010000011010000" 20D0h MOVEBIT 2 bit U
"0011100000000000" 3800h MOVEBIT 3 bit C
"0010000000001000" 2008h MOVEBIT 2 bit P

AES/EBU 24 bits, signal vidéo HDTV en mode MUX

"1001101111010110" 9CD6h NOP MUX preamble 0
"0010101111010000" 2BD0h MOVEBIT 2 preamble 1
"0010101111010000" 2BD0h MOVEBIT 2 preamble 2
"0010011011010000" 26D0h MOVEBIT 2 preamble 3
"0010000011010000" 20D0h MOVEBIT 2 audio 0
"0010000011010000" 20D0h MOVEBIT 2 audio 1
"0010000011010000" 20D0h MOVEBIT 2 audio 2
"0011100000000000" 3800h MOVEBIT 3 audio 3
"0010000011010000" 20D0h MOVEBIT 2 audio 4
"0010000011010000" 20D0h MOVEBIT 2 audio 5
"0010000011010000" 20D0h MOVEBIT 2 audio 6
"0010000011010000" 20D0h MOVEBIT 2 audio 7
"0010000011010000" 20D0h MOVEBIT 2 audio 8
"0010000011010000" 20D0h MOVEBIT 2 audio 9

"0010000011010000"	20D0h MOVEBIT 2 audio 10
"0011100000000000"	3800h MOVEBIT 3 audio 11
"0010000011010000"	20D0h MOVEBIT 2 audio 12
"0010000011010000"	20D0h MOVEBIT 2 audio 13
"0010000011010000"	20D0h MOVEBIT 2 audio 14
"0010000011010000"	20D0h MOVEBIT 2 audio 15
"0010000011010000"	20D0h MOVEBIT 2 audio 16
"0010000011010000"	20D0h MOVEBIT 2 audio 17
"0010000011010000"	20D0h MOVEBIT 2 audio 18
"0011100000000000"	3800h MOVEBIT 3 audio 19
"0010000011010000"	20D0h MOVEBIT 2 audio 20
"0010000011010000"	20D0h MOVEBIT 2 audio 21
"0010000011010000"	20D0h MOVEBIT 2 audio 22
"0010000011010000"	20D0h MOVEBIT 2 audio 23
"0010000011010000"	20D0h MOVEBIT 2 bit V
"0010000011010000"	20D0h MOVEBIT 2 bit U
"0010000011010000"	20D0h MOVEBIT 2 bit C
"0011010000000000"	3400h MOVEBIT 3 bit P

TONES, signal vidéo DTV mode MUX

"1001000000010110"	9016h NOP MUX preamble 0
"0010011011010000"	26D0h MOVEBIT 2 preamble 1
"0010100011010000"	28D0h MOVEBIT 2 preamble 2
"0010101111010000"	2BD0h MOVEBIT 2 preamble 3
"0000000000000000"	0000h NOP aux 0
"0000000000000000"	0000h NOP aux 1
"0000000000000000"	0000h NOP aux 2
"0000000000000000"	0000h NOP aux 3
"1010001100000010"	A302h MOVEBIT 7 tones 0

"1010001100010010" A312h MOVEBIT 7 tones 1
"1010001100100010" A322h MOVEBIT 7 tones 2
"1010001100110010" A332h MOVEBIT 7 tones 3
"1010001101000010" A342h MOVEBIT 7 tones 4
"1010001101011110" A35Eh MOVEBIT 7 tones 5
"1010001101100010" A362h MOVEBIT 7 tones 6
"1010001101110010" A372h MOVEBIT 7 tones 7
"1010001110000010" A382h MOVEBIT 7 tones 8
"1010001110010011" A393h MOVEBIT 7 tones 9
"1010001100000010" A302h MOVEBIT 7 tones 10
"1010001100010010" A312h MOVEBIT 7 tones 11
"1010001100100010" A322h MOVEBIT 7 tones 12
"1010001100110010" A332h MOVEBIT 7 tones 13
"1010001101001110" A34Eh MOVEBIT 7 tones 14
"1010001101010010" A352h MOVEBIT 7 tones 15
"1010001101100010" A362h MOVEBIT 7 tones 16
"1010001101110010" A372h MOVEBIT 7 tones 17
"1010001110000010" A382h MOVEBIT 7 tones 18
"1010001110010011" A383h MOVEBIT 7 tones 19
"0010101011010000" 2AD0h MOVEBIT 2 bit V
"0010101011010000" 2AD0h MOVEBIT 2 bit U
"0010101011010000" 2AD0h MOVEBIT 2 bit C
"0110110001000000" 6C40h MOVEBIT 4 bit P

TONES, signal vidéo HDTV, mode MUX

"1001101111010110" 9CD6h NOP MUX preamble 0
"0010101011010000" 2AD0h MOVEBIT 2 preamble 1
"0010101011010000" 2AD0h MOVEBIT 2 preamble 2
"0010011011010100" 26D4h MOVEBIT 2 preamble 3

"0010101011010000"	2AD0h MOVEBIT 2 aux 0
"0010101011010000"	2AD0h MOVEBIT 2 aux 1
"0010101011010000"	2AD0h MOVEBIT 2 aux 2
"1100000010110000"	C0B0h nouvelle instruction non commentée
"1010001100000010"	A302h MOVEBIT 7 tones 0
"1010001100010010"	A312h MOVEBIT 7 tones 1
"1010001100100010"	A322h MOVEBIT 7 tones 2
"1010001100110010"	A332h MOVEBIT 7 tones 3
"1010001101000010"	A342h MOVEBIT 7 tones 4
"1010001101010010"	A352h MOVEBIT 7 tones 5
"1010001101100010"	A362h MOVEBIT 7 tones 6
"1100011110100000"	C7A0h nouvelle instruction non commentée
"1010001110000010"	A382h MOVEBIT 7 tones 8
"1010001110010011"	A393h MOVEBIT 7 tones 9
"1010001100000010"	A302h MOVEBIT 7 tones 10
"1010001100010010"	A312h MOVEBIT 7 tones 11
"1010001100100010"	A322h MOVEBIT 7 tones 12
"1010001100110010"	A332h MOVEBIT 7 tones 13
"1010001101000010"	A342h MOVEBIT 7 tones 14
"1100010110100000"	C5A0h nouvelle instruction non commentée
"1010001101100010"	A362h MOVEBIT 7 tones 16
"1010001101110010"	A372h MOVEBIT 7 tones 17
"1010001110000010"	A382h MOVEBIT 7 tones 18
"1010001110010011"	A383h MOVEBIT 7 tones 19
"0010101011010000"	2AD0h MOVEBIT 2 bit V
"0010101011010000"	2AD0h MOVEBIT 2 bit U
"0010101011010000"	2AD0h MOVEBIT 2 bit C
"1100000001000000"	C040h nouvelle instruction non commentée

AES/EBU 20, 24 bits signal vidéo DTV en mode DEMUX

"1000000010111101"	80BDh MOVEBIT 6 preamble 0 canal 1
"1000000010110000"	80B0h MOVEBIT 6 preamble 1 canal 1
"1000000010110000"	80B0h MOVEBIT 6 preamble 2 canal 1
"1000000010110000"	80B0h MOVEBIT 6 preamble 3 canal 1
"1000000001010010"	8052h MOVEBIT 6 audio 0 canal 1
"1000000101010010"	8152h MOVEBIT 6 audio 1 canal 1
"1000001001010010"	8252h MOVEBIT 6 audio 2 canal 1
"1000001101010010"	8352h MOVEBIT 6 audio 3 canal 1
"1000001100110000"	8330h MOVEBIT 6 audio 4 canal 1
"1000010000110000"	8430h MOVEBIT 6 audio 5 canal 1
"1000010100110000"	8530h MOVEBIT 6 audio 6 canal 1
"1000011000110000"	8630h MOVEBIT 6 audio 7 canal 1
"1000011100110000"	8730h MOVEBIT 6 audio 8 canal 1
"1000100000111000"	8838h MOVEBIT 6 audio 9 canal 1
"1000000000110000"	8030h MOVEBIT 6 audio 10 canal 1
"1000000100110000"	8130h MOVEBIT 6 audio 11 canal 1
"1000001000110000"	8230h MOVEBIT 6 audio 12 canal 1
"1000001100110000"	8330h MOVEBIT 6 audio 13 canal 1
"1000010000110000"	8430h MOVEBIT 6 audio 14 canal 1
"1000010100110000"	8530h MOVEBIT 6 audio 15 canal 1
"1000011000110000"	8630h MOVEBIT 6 audio 16 canal 1
"1000011100110000"	8730h MOVEBIT 6 audio 17 canal 1
"1000100000111000"	8838h MOVEBIT 6 audio 18 canal 1
"1000000000110000"	8030h MOVEBIT 6 audio 19 canal 1
"1000000100110000"	8130h MOVEBIT 6 audio 20 canal 1
"1000001000110000"	8230h MOVEBIT 6 audio 21 canal 1
"1000001100110000"	8330h MOVEBIT 6 audio 22 canal 1
"1000010000110000"	8430h MOVEBIT 6 audio 23 canal 1

"1000010100110000"	8530h MOVEBIT 6 bit V canal 1
"1000011000110000"	8630h MOVEBIT 6 bit U canal 1
"1000011100110000"	8730h MOVEBIT 6 bit C canal 1
"1000000011010100"	80D4h MOVEBIT 6 bit P canal 1
"1000000010111100"	80BCh MOVEBIT 6 preamble 0 canal 2
"1000000010110000"	80B0h MOVEBIT 6 preamble 1 canal 2
"1000000010110000"	80B0h MOVEBIT 6 preamble 2 canal 2
"1000000010110000"	80B0h MOVEBIT 6 preamble 3 canal 2
"1000010001010010"	8452h MOVEBIT 6 audio 0 canal 2
"1000010101010010"	8552h MOVEBIT 6 audio 1 canal 2
"1000011001010010"	8652h MOVEBIT 6 audio 2 canal 2
"1000011101010010"	8752h MOVEBIT 6 audio 3 canal 2
"1000001100110000"	8330h MOVEBIT 6 audio 4 canal 2
"1000010000110000"	8430h MOVEBIT 6 audio 5 canal 2
"1000010100110000"	8530h MOVEBIT 6 audio 6 canal 2
"1000011000110000"	8630h MOVEBIT 6 audio 7 canal 2
"1000011100110000"	8730h MOVEBIT 6 audio 8 canal 2
"1000100000111000"	8838h MOVEBIT 6 audio 9 canal 2
"1000000000110000"	8030h MOVEBIT 6 audio 10 canal 2
"1000000100110000"	8130h MOVEBIT 6 audio 11 canal 2
"1000001000110000"	8230h MOVEBIT 6 audio 12 canal 2
"1000001100110000"	8330h MOVEBIT 6 audio 13 canal 2
"1000010000110000"	8430h MOVEBIT 6 audio 14 canal 2
"1000010100110000"	8530h MOVEBIT 6 audio 15 canal 2
"1000011000110000"	8630h MOVEBIT 6 audio 16 canal 2
"1000011100110000"	8730h MOVEBIT 6 audio 17 canal 2
"1000100000111000"	8838h MOVEBIT 6 audio 18 canal 2
"1000000000110000"	8030h MOVEBIT 6 audio 19 canal 2
"1000000100110000"	8130h MOVEBIT 6 audio 20 canal 2

"1000001000110000" 8230h MOVEBIT 6 audio 21 canal 2
"1000001100110000" 8330h MOVEBIT 6 audio 22 canal 2
"1000010000110000" 8430h MOVEBIT 6 audio 23 canal 2
"1000010100110000" 8530h MOVEBIT 6 bit V canal 2
"1000011000110000" 8630h MOVEBIT 6 bit U canal 2
"1000011100110000" 8730h MOVEBIT 6 bit C canal 2
"1000000011000000" 80C0h MOVEBIT 6 bit P canal 2

AES/EBU 24 bits signal vidéo HDTV en mode DEMUX

"1000000010111101" 80BDh MOVEBIT 6 preamble 0
"1000000010110000" 80B0h MOVEBIT 6 preamble 1
"1000000010110000" 80B0h MOVEBIT 6 preamble 2
"1000000010110000" 80B0h MOVEBIT 6 preamble 3
"1000010000110000" 8430h MOVEBIT 6 audio 0
"1000010100110000" 8530h MOVEBIT 6 audio 1
"1000011000110000" 8630h MOVEBIT 6 audio 2
"1000011100111000" 8730h MOVEBIT 6 audio 3
"1000000000110000" 8030h MOVEBIT 6 audio 4
"1000000100110000" 8138h MOVEBIT 6 audio 5
"1000001000110000" 8230h MOVEBIT 6 audio 6
"1000001100110000" 8330h MOVEBIT 6 audio70
"1000010000110000" 8430h MOVEBIT 6 audio 8
"1000010100110000" 8530h MOVEBIT 6 audio 9
"1000011000110000" 8630h MOVEBIT 6 audio 10
"1000011100111000" 8730h MOVEBIT 6 audio 11
"1000000000110000" 8030h MOVEBIT 6 audio 12
"1000000100110000" 8130h MOVEBIT 6 audio 13
"1000001000110000" 8238h MOVEBIT 6 audio 14
"1000001100110000" 8330h MOVEBIT 6 audio 15

"1000010000110000"	8430h MOVEBIT 6 audio 16
"1000010100110000"	8530h MOVEBIT 6 audio 17
"1000011000110000"	8630h MOVEBIT 6 audio 18
"1000011100111000"	8730h MOVEBIT 6 audio 19
"1000000000110000"	8030h MOVEBIT 6 audio 20
"1000000100110000"	8130h MOVEBIT 6 audio 21
"1000001000110000"	8230h MOVEBIT 6 audio 22
"1000001100110000"	8330h MOVEBIT 6 audio 23
"1000010000110000"	8430h MOVEBIT 6 bit V
"1000010100110000"	8530h MOVEBIT 6 bit U
"1000011000110000"	8630h MOVEBIT 6 bit C
"1000011100100000"	8720h MOVEBIT 6 bit P

ANNEXE C

Détails de l'interface I²C et des registres internes du processeur

Une interface I²C est défini à l'aide de deux lignes sérielles. Une ligne est réservée pour l'horloge des données. Une autre ligne est réservée pour les données. Plusieurs circuits intégrés peuvent être connectés sur un tel support de communication. On appelle « master » tout composant branché sur une interface I²C et qui génère l'horloge de communication sur la ligne SCL. On appelle « slave » tout composant branché sur une interface I²C qui ne fournit pas d'horloge sur la ligne SCL. Plusieurs « master » peuvent être connectés sur un même lien. Pour déterminer la priorité entre « master », ils doivent être à l'écoute de la ligne de données pour voir si les informations qu'ils envoient correspondent à ce qui se trouve sur la ligne de donnée. Ainsi, dès qu'un « master » constate que ce qui est sur la ligne de données ne correspond pas à ce qu'il envoie, il doit cesser d'émettre et attendre que la ligne se libère. La façon de communiquer entres composants sur une ligne est simple. Le « master » envoie tout d'abord une adresse sur la ligne de données. Cette adresse peut être constituée de 7 ou 10 bits, dépendamment du protocole en présence. Chaque composant à sa propre adresse qui est unique (elle est définie par la compagnie Philips, inventeur de l'interface I²C). Chaque composant doit être constamment à l'écoute de la ligne de données (SDA) pour savoir si on lui « parle ». Quand un composant détecte qu'on veut lui parler, il doit envoyer un signal sur SDA disant qu'il a reconnaît qu'on s'adresse à lui. Quand ce signal n'est pas émis, le « master » renverra l'adresse sur la ligne SDA un certain nombre de fois (si la ligne est libre) avant de juger que le composant auquel il désire s'adresser n'est pas connecté. Dans le cas d'une communication établie, le « master » voudra lire le contenu d'un registre du composant à qui il s'adresse ou écrire dans un de ces registres. La figure C.1 montre l'interface électrique de l'interface I²C. Divers renseignements sur ce mode de communication et sur le processeur à données ancillaires intégré sont ensuite fournis.

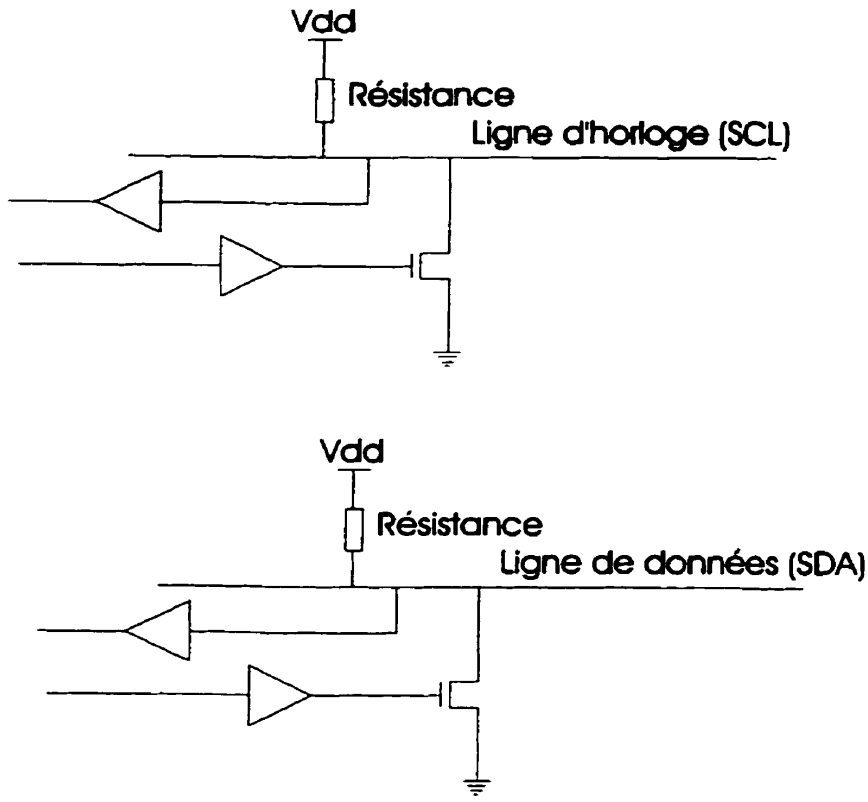


Figure C.1 Interface électrique I²C

Procédure d'écriture dans un protocole d'adresse à 7 bits

S	Slave address W	ACKs	Subaddress	ACKs	DATA (n bytes)	ACKs	P

Procédure de lecture dans un protocole d'adresse à 7 bits

S	Slave address W	ACKs	Subaddress	ACKs	
Sr	Slave address R	ACKs	DATA (1byte)	ACKm	P

Procédure d'écriture dans un protocole d'adresse à 10 bits

S	Slave address 1st 7 bits + W	ACKs	Slave address 2nd byte	ACKs	Subaddress
ACKs	DATA (n bytes)	ACKs	P		

Procédure de lecture dans un protocole d'adresse à 10 bits

S	Slave address 1st 7 bits + W	ACKs	Slave address 2nd byte	ACKs	Subaddress	ACKs
Sr	Slave address 1st 7 bits + R	ACKs	DATA (1 byte)	ACKm	P	

Légende :

S : Start condition

Sr : repeated Start condition

Slave address : l'adresse I²C du processeur; en 7 bits

ACKs : acknowledge généré par le slave (le processeur)

ACKm : acknowledge généré par le master

Subaddress : subaddress byte désignant le numéro d'un registre interne du processeur (voir tableau 3.5)

DATA : data byte, contenu du registre désigné par le subaddress (voir page suivante)

P : Stop condition

Note : En procédure d'écriture, si plus d'un byte de DATA est transmis au processeur, une auto-incrémentation du subaddress est appliquée.

Registres du processeur accédés par l'interface I²C

Fonction du registre	Subaddress	D7	D6	D5	D4	D3	D2	D1	D0
Standard Vidéo	00h	0	0	0	0	u	x	x	x
Testmode	01h	0	0	0	0	0	y	y	y
Audiogr A	02h	0	0	0	0	0	0	z	z
Audiogr B	03h	0	0	0	0	0	0	z	z
Delete	04h	0	0	0	0	Daudio	Daux	Dltc	Drs422
Auto Détection	05h	0	0	0	0	0	0	0	a
Error Flags m.b.-s. 1,2	06h	0	0	E10	E11	E12	E20	E21	E22
Error Flags m.b.-s. 3,4	07h	0	0	E30	E31	E32	E40	E41	E42
Async audio m.b.-s. 1,2	08h	0	0	0	0	0	0	0	b
Async audio m.b.-s. 3,4	09h	0	0	0	0	0	0	0	c
Audio group Presence	0Ah	0	0	0	0	a4	a3	a2	a1

Notes :

0) Standard vidéo :

u = format inconnu (Unknown)
xxx = « 000 » = > DTV, 525 lignes, 27 MHz
xxx = « 001 » => DTV, 625 lignes, 27 MHz
xxx = « 010 » => standard, 525 lignes, 36 MHz
xxx = « 011 » => standard, 625 lignes, 36 MHz
xxx = « 100 » => HDTV

1) Testmode (sous mode) :

yyy = « 000 » => normal, yyy = « 001 » => bypass, yyy = « 010 » => tones,
yyy = « 011 » = > colour bar, yyy = « 100 » => tones & colour bar

2,3) Audiogroup A, B

zz = « 00 » => audio group 1, zz = « 01 » => audio group 2,
zz = « 10 » => audio group 3, zz = « 11 » => audio group 4.

4) Delete

Delete audio : '1' : enlever les paquets audio ayant le groupe audio recherché
Delete aux : '1' : enlever les paquets auxiliaires audio ayant le groupe audio recherché.
Delete ltc : '1' : enlever les paquets LTC ayant le DID recherché
Delete rs422 : '1' enlever les paquets RS422 ayant le DID recherché

5) Autodetection

'1' : c'est le processeur qui fait la détection du standard vidéo, '0' : le standard est imposé au processeur via l'interface I^2C.

6,7) Drapeaux d'erreur audio venant du receveur audio CS8412 externe au processeur

8,9) Asynchronisme audio

a = '1' : asynchronisme audio relié aux machines bit-sérielles des blocs de données audio 1 et audio 2. S'applique en mode DEMUX seulement.
b = '1' : asynchronisme audio relié aux machines bit-sérielles des blocs de données LTC et RS422, lorsqu'elles traitent des données audio. S'applique en mode DEMUX seulement.

10) Présence des groupes audio dans le signal vidéo d'entrée

a4 = '1' => audio group 4 présent
a3 = '1' => audio group 3 présent
a2 = '1' => audio group 2 présent
a1 = '1' = > audio group 1 présent

Les registres 06h, 07h, 08h, 09h, 0Ah sont accessibles en lecture seulement alors que les autres sont accessibles à la fois en écriture et en lecture.