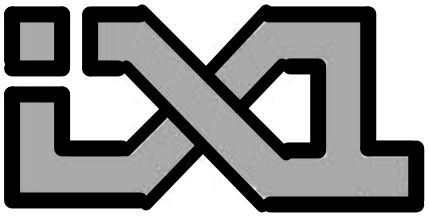


LABORATOIRE D'ETUDES DE L'INTEGRATION
DES COMPOSANTS ET SYSTEMES ELECTRONIQUES

Universite de Bordeaux I - E.N.S.E.R.B.

C.N.R.S. URA 846

CARTE D'ACQUISITION PCI: CONCEPTION ET REALISATION

Antonio PEREZ-BERDUD
Vincent CHEVALIER

Juin 1996

Projet dirigé par: Mr M. BENKAIS
Mr P. MARCHEGAY

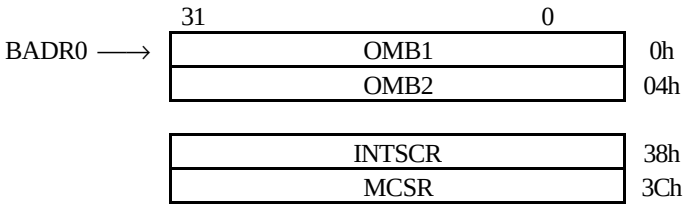
REMERCIEMENTS

Nous tenons à remercier tout particulièrement Monsieur MARCHEGAY et Monsieur BENKAIS responsables du projet, ainsi que toutes les personnes qui nous ont aidé pendant notre projet.

TABLE DES MATIERES

ANNEXES

MAPPING DES REGISTRES



REGISTRES DE L'INTERFACE

	31	30	29		7	6	5	4	3	2	1	0
BADR1 ⟶	X	FRONT	ON/OFF	XXX	CE7	CE6	CE5	CE4	CE3	CE2	CE1	CE0
	X	R/W	R/W	XXX	W	W	W	W	W	W	W	W

REGISTRE DE CONTROLE

	31	20	19	18		1	0
BADR2 ⟶	X X X X X X X X X X X X		C19	C18	C17 à C2	C1	C0
	X X X X X X X X X X X X		W	W		W	W

REGISTRE DE CHARGEMENT

	31	24	23	22		1	0
BADR3 ou BADR4 ⟶	X X X X X X X X		D23	D22	D21 à D2	D1	D0
	X X X X X X X X		R	R		R	R

REGISTRE DE LA RAM

I. INTRODUCTION

I.1 Historique

Dans le domaine du traitement de l'information, les convertisseurs Analogique/ Numérique ont une grande utilité. Les tests de ces composants toujours plus performants deviennent de plus en plus sophistiqués. C'est pourquoi l'utilisation d'outils puissants et conviviaux, tel que l'informatique, est devenu nécessaire.

L'évolution rapide des ordinateurs a amené l'IXL à développer des cartes d'acquisition de plus en plus élaborées.

Les différentes versions qui ont été développées à ce jour sont indiquées dans le tableau suivant :

Année	Développeur	BUS	Ordinateur employé	Fréquence Max	Taille mémoire
1989	IXL	NuBus	Macintosh	25 MHz x 24 bits	8k x 24 bits
1995	IXL	NuBus	MAC et PowerMAC	50 MHz x 24 bits	64k x 24 bits
1993	Thomson	ISA	PC	75 MHz x 24 bits	256k x 16 bits

Tableau I-1: les anciennes versions de carte d'acquisition

L'objectif de notre projet est la réalisation d'une nouvelle version de carte d'acquisition, plus rapide, qui puisse fonctionner à la fois sur PC et sur PowerMAC.

I.2 Cahier des charges

Notre travail va se décomposer en plusieurs étapes:

- Etude du bus PCI et choix de l'interface.

- Conception de la carte devant répondre aux caractéristiques suivantes:
  - carte au format du bus PCI version 2.0 (33 MHz).
  - acquisition d'échantillons 24 bits à la fréquence maximale de 100 MHz.

- stockage de l'acquisition dans une mémoire RAM de 1M x 24 bits (3 Mo).
- compatibilité PC et PowerMAC.

- Développement de l'interface logicielle avec le matériel pour PC et PowerMAC.

- Routage de la carte.

II. LE BUS PCI

II.1 Généralités

Le bus PCI (Peripheral Component Interconnect) est un bus utilisé pour la connexion de tout type de périphérique. Quasiment devenu une norme, on le rencontre désormais dans tous les PC et dans les plus récents PowerMacintosh. Cette caractéristique nous assure la compatibilité recherchée et justifie le choix de ce bus.

Le bus PCI est un bus local à performances élevées, proche du microprocesseur. La figure ci-dessous montre l'architecture globale que l'on peut trouver dans les ordinateurs possédant un bus PCI:

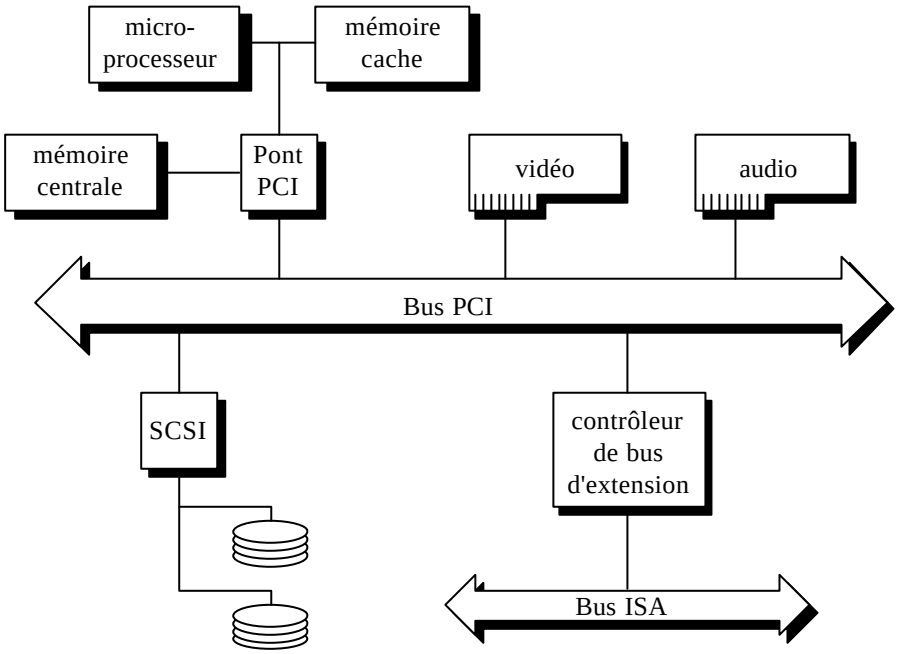


Figure II-1: architecture du bus PCI

Les performances du bus PCI se caractérisent surtout par une grande vitesse de transfert (jusqu'à 528 Mo/s en débit crête pour les plus récentes spécifications) et par la configuration automatique de ses cartes. Plusieurs raisons sont à l'origine de sa vitesse élevée:

- le bus PCI dispose du mode de transfert appelé mode rafale linéaire ou mode BURST. Pendant une opération de lecture ou d'écriture, le bus d'adresse et le bus de donnée sont multiplexés: deux phases sont donc nécessaires: une phase d'adressage et une phase de donnée. Le BURST est un transfert composé d'une phase d'adresse et de plusieurs phases de données ce qui permet d'augmenter le débit. Pour cela, chaque périphérique doit implicitement incrémenter l'adresse à chaque nouvelle phase de donnée en partant de l'adresse de base précisée au début du transfert. Cette technique d'adressage linéaire se révèle particulièrement efficace pour la lecture ou l'écriture d'une grande quantité d'informations.

- la largeur du bus est de 32 bits et peut être étendue à 64 bits.

- la vitesse de fonctionnement: cadencé à une fréquence constante de 33 MHz (66 MHz pour la version 2.1), quelle que soit celle du microprocesseur employé, le bus PCI est nettement plus rapide que le bus ISA et EISA d'un PC qui ne fonctionnent qu'à 8 MHz ou le Nubus à 10 MHz d'un Macintosh. D'autre part, il est complètement indépendant du CPU contrairement à d'autres bus rapides comme le Vesa Local Bus de certains PC.

Mais les performances de ce bus ne s'expliquent pas uniquement par sa largeur de bande élevée. Il permet également l'autoconfiguration logicielle des cartes qui lui sont connectées. Pour cela, chaque périphérique doit posséder une zone de mémoire particulière appelée zone de configuration qui contient les informations nécessaires pour que le système d'exploitation (le BIOS dans un PC, l'Open Firmware Code dans un MAC) puisse correctement configurer la carte. Cette zone comprend notamment les identifications du périphérique, la mémoire dont il a besoin, des temps de réponses, sa fonctionnalité, etc..

A l'issue de cette configuration, chaque périphérique présent sur le bus et reconnu en tant qu'agent du bus, se verra attribué un espace mémoire qui lui est réservé et qui n'entrera pas en

conflit avec celui des autres périphériques comme cela peut être le cas lorsqu'il s'agit d'une configuration manuelle.

Le bus PCI est d'autre part un bus évolutif: il a été réalisé dans une optique d'avenir. La version 2.0 du bus PCI propose classiquement un bus 32 bits à 33 MHz pour une vitesse de transfert atteignant 132 Mo/s en débit crête. Elle propose également une extension 64 bits permettant des vitesses de 264 Mo/s.

La version 2.1 du bus PCI améliore la vitesse d'horloge, celle-ci pouvant atteindre 66 MHz: avec une largeur de bus de 64 bits, la vitesse maximale passe donc à 528 Mo/s.

Le bus PCI accepte la technologie 3.3V qui, à terme, remplacera la technologie 5V.

La proximité du microprocesseur et la vitesse du bus limite la charge et donc le nombre de connecteurs (deux à trois selon les ordinateurs). Il peut cependant être augmenté grâce à l'utilisation d'un pont PCI/PCI, véritable passerelle vers un autre bus PCI qui peut à son tour posséder d'autres connecteurs et un autre pont PCI. C'est ainsi que l'on peut voir sur certains PowerMAC, six connecteurs PCI.

II.2 Spécifications du bus

Les spécifications indiquées dans ce chapitre se rapportent à la version 2.0 du bus PCI.

II.2.1 Principes de base

Chaque entité présente sur le bus PCI est appelée agent du bus. Cela désigne les cartes connectées, le microprocesseur ou encore la mémoire centrale bien que ces deux derniers doivent passer par l'intermédiaire du pont hôte qui joue le rôle d'interface entre le bus interne du CPU et le bus PCI.

Chaque opération effectuée sur le bus est une transaction. Elle concerne deux agents du bus PCI, l'un étant le maître c'est à dire celui qui réalise l'accès, et l'autre étant la cible ou

l'esclave. N'importe quel agent peut, à des moments différents, agir aussi bien en maître qu'en esclave.

Toutes les transactions du bus sont gérées par un circuit spécialisé appelé arbitre des transactions. C'est lui qui accorde la possession du bus aux agents qui désirent l'utiliser (les maîtres) selon un algorithme complexe, de manière à minimiser les temps d'accès et à satisfaire tous les agents.

II.2.2 Description des principaux signaux

L'interface d'une carte PCI requiert un minimum de 47 broches (esclave). Le schéma ci-dessous indique tous les signaux définis par les spécifications du bus (version 2.0):

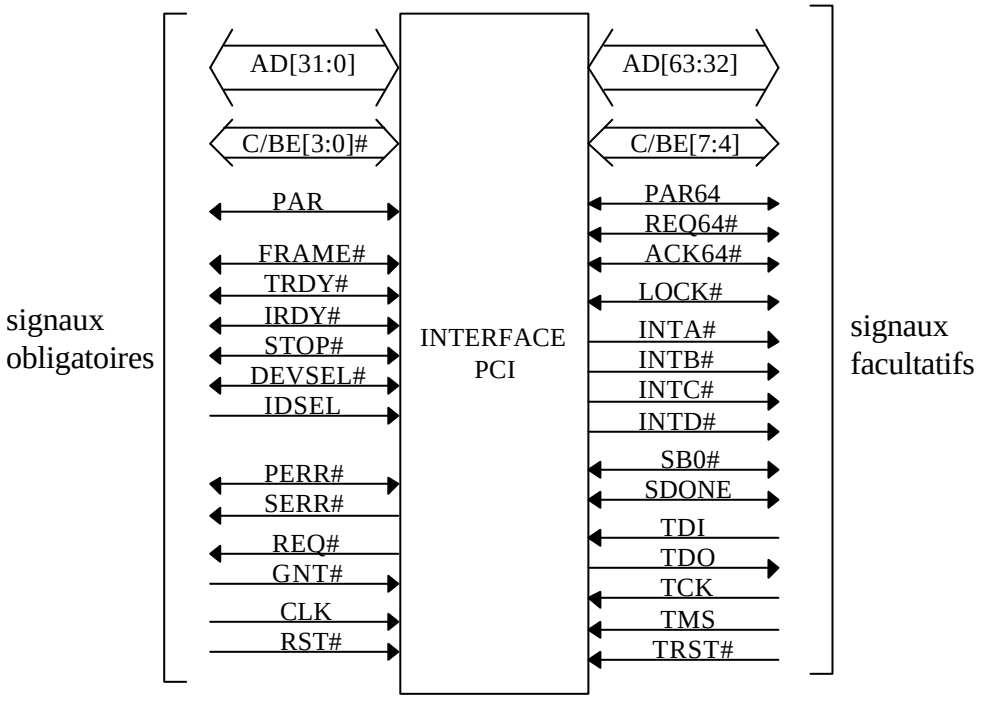


Figure II-2: les signaux d'une interface PCI

Système:

- CLK est l'horloge du bus PCI (à 33 MHz généralement)

- RST# est le reset utilisé pour l'initialisation de tous les composants du bus.

Adresse et données:

- AD[31:0]: les adresses et les données sont multiplexées. AD[31:0] véhiculent donc l'adresse pendant la phase d'adressage et les données pendant les phases de données.
- C/BE[3:0]#: ces quatre signaux sont également multiplexés. Durant la phase d'adressage, ils indiquent le type de l'opération effectuée sur le bus (Command) (voir tableau ci-dessous). Pendant les phases de données, ils indiquent quels octets du mot de 32 bits contiennent l'information utile (Byte Enable).

C/BE	Nom de la commande
0000	**Interrupt Acknowledge**
0001	**Special Cycle**
0010	**I/O Read**
0011	**I/O Write**
0100	**Reserved**
0101	**Reserved**
0110	**Memory Read**
0111	**Memory Write**
1000	**Reserved**
1001	**Reserved**
1010	**Configuration Read**
1011	**Configuration Write**
1100	**Memory Read Multiple**
1101	**Dual Address Cycle**
1110	**Memory Read Line**
1111	**Memory Write and Invalidate**

Tableau II-1: signification de COMMAND

- PAR est un signal de parité calculé sur AD[31:0] et C/BE[3:0]# pour la validité des données transmises.

Contrôle d'interface:

- FRAME# est un signal fourni par l'initiateur d'une transaction (le maître) pour indiquer le début d'une transaction ainsi que sa durée.
- IRDY# est le signal indiquant que le maître est prêt à passer à la phase de donnée suivante.
- TRDY# indique que la cible est prête pour la phase suivante.
- STOP# est généré par la cible pour interrompre la transaction en cours.
- LOCK# est utilisé pour réserver l'accès d'une cible à un maître particulier.
- DEVSEL# est activé par un esclave pour indiquer au maître qu'il répond à l'accès que celui réalise sur le bus.
- IDSEL joue le rôle de "Chip Select" de la zone de configuration, activé pendant les opérations d'accès en zone de configuration.

Arbitrage:

- REQ# indique à l'arbitre des transactions qu'un maître désire utiliser le bus.
- GNT# est la réponse à cette requête. Il est activé lorsque la demande du maître est acceptée.

Chaque maître doit posséder ces deux signaux.

Détection d'erreur:

- PERR# signale une erreur de parité pendant les phases de données.
- SERR# signale une erreur de parité pendant une opération "Special Cycle" et pendant les phases d'adressage.

Interruptions:

INTA#, INTB#, INTC# et INTD# sont quatre signaux d'interruptions connectés au routeur d'interruptions chargé de les relier au contrôleur d'interruption. Les trois derniers signaux ne sont utilisables que par une carte multifonctions.

Extension 64 bits:

- AD[63:32] représentent les 4 octets de poids forts du mot de 64 bits lorsque des accès 64 bits sont réalisés.
- C/BE[7:4]# indiquent quels octets sont valides.
- REQ64# est une demande d'accès 64 bits en direction de la cible.
- ACK64# est la réponse de la cible à cette demande. Si la cible ne fonctionne qu'en 32 bits, le maître doit également fonctionner en 32 bits.

Support Cache:

SB0# et SDONE sont des signaux utilisés lors d'accès à de la RAM PCI cachée.

JTAG:

TDI, TDO, TCK, TMS et TRST# sont utilisés pour le test de carte supportant le Boundary Scan.

II.2.3 Protocole en lecture et écriture

Les opérations les plus couramment utilisées sur le bus sont les accès mémoires. Les protocoles détaillés ci-dessous sont basés sur ces types d'accès mais décrivent également tous les autres.

Chaque agent esclave possède une ou plusieurs zones réservées en mémoire (jusqu'à six). Lorsqu'un maître désire accéder à un esclave, il doit préciser l'adresse correspondant à une de ses zones.

Dans un PC, deux plans mémoires sont disponibles: le champ d'entrées/sorties ou champ I/O limité à 64 Ko et le champ mémoire classique (RAM). Dans un PowerMAC, le champ I/O n'existe pas mais il est recréé dans le champ mémoire à des adresses ne contenant pas de RAM.

Lecture:

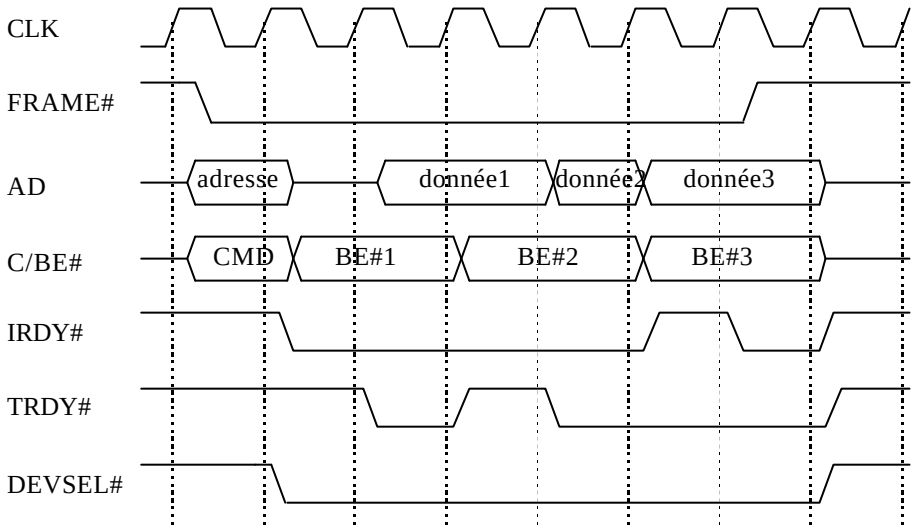


Figure II-3: lecture sur le bus PCI

Le transaction débute avec l'état bas de FRAME#. Simultanément, AD indique l'adresse et C/BE# la commande "lecture". L'esclave qui décode l'adresse doit répondre par l'activation du signal DEVSEL#.

On peut remarquer ensuite un cycle d'horloge pendant lequel TRDY# reste inactif et où aucune opération n'a lieu. Ce cycle appelé "turnaround cycle" est utilisé pour éviter toute contention sur le bus entre le maître et l'esclave qui utilisent tous les deux AD[31:0] (le maître fournit l'adresse et l'esclave les données).

Le transfert de données est réalisé lorsque les deux agents de la transaction en cours sont prêts c'est à dire quand IRDY# et TRDY# sont tous les deux à l'état bas. Une nouvelle phase de donnée peut alors débuter.

La fin de la transaction est indiquée par les états hauts de FRAME#, IRDY# et TRDY#. FRAME# se relève en effet lorsque la dernière phase de donnée va avoir lieu et quand IRDY# est actif (le maître est prêt à effectuer le dernier transfert).

On peut ainsi détecter un transfert BURST si l'on a FRAME# et IRDY# actifs dans un même cycle d'horloge.

Ecriture:

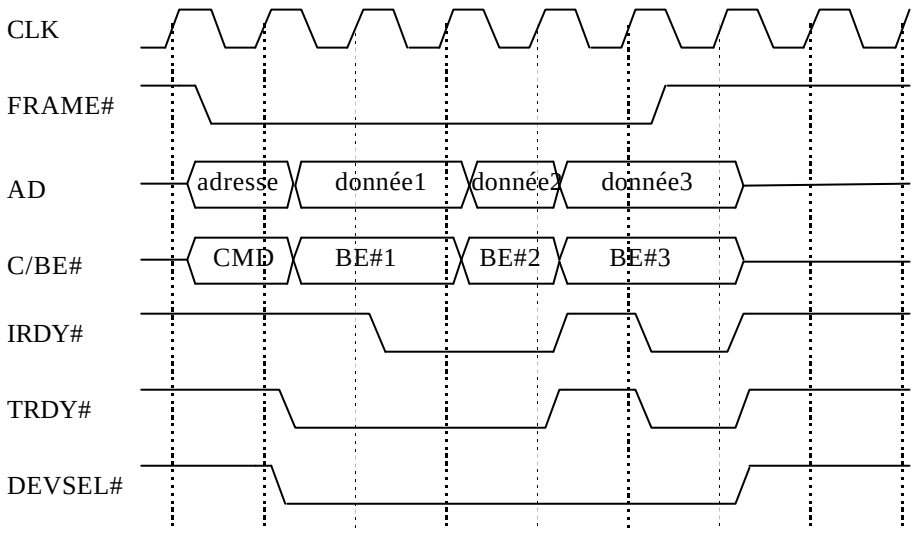


Figure II-4: écriture sur le bus PCI

L'écriture s'effectue de la même manière qu'une lecture mis à part la disparition du "turnaround cycle" qui n'est plus nécessaire puisque le maître fournit à la fois l'adresse et les données.

Remarques: afin de ne pas dégrader les performances du bus, certains temps de réponses doivent être respectés:

- la durée maximale d'une phase de donnée est de 8 cycles d'horloge.

- la réponse de la cible par DEVSEL# ne doit pas dépasser 6 cycles d'horloge.
- la longueur d'un BURST est limité par l'arbitre des transactions.

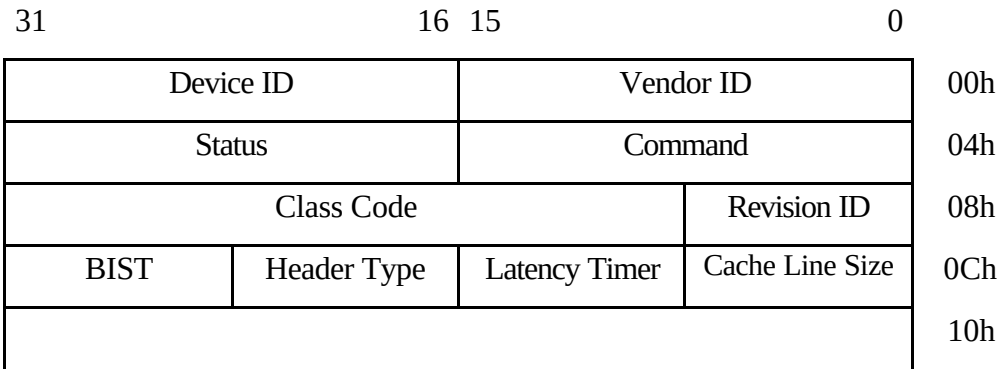
II.2.4 L'autoconfiguration

Pour permettre l'autoconfiguration, chaque périphérique doit posséder une zone de configuration organisée selon une structure spécifique. C'est une zone de 256 octets découpée en champs de taille variable contenant diverses informations nécessaires à l'initialisation et au fonctionnement de l'interface sur le bus PCI.

Dans le PC ou le PowerMAC, il n'existe pas de mémoire réservée pour les zones de configuration des cartes PCI. Dans un PC, l'accès à ces zones se fait par l'intermédiaire de 2 registres situés dans le champ I/O. Les opérations réalisées dans ces registres sont transformées par le pont hôte en opération d'accès en zone de configuration (Configuration Read ou Configuration Write) sur le bus PCI. Dans un PowerMAC, le mécanisme d'accès est très semblable et utilise également deux registres.

Il n'est pas nécessaire de connaître le détail exact de ce type d'opération: l'accès aux zones de configuration peut en effet se faire de manière totalement transparente par l'intermédiaire de routines spéciales: dans un PC ce sont des interruptions logicielles du BIOS et dans un PowerMAC, ce sont des fonctions de bibliothèques.

La figure ci-dessous présente les 64 premiers octets de la zone de configuration d'une carte PCI dont l'organisation est à respecter:



6 Base Address Registers (BADR0 à BADR5)				14h 18h 1Ch 20h 24h
Reserved				28h
Reserved				2Ch
Expansion ROM Base Address				30h
Reserved				34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

Figure II-5: la zone de configuration

Identification du périphérique:

- *Vendor ID*: ce champ contient l'identification du vendeur. Cette valeur doit être différente de FFFF pour que le périphérique puisse être reconnu sur le bus.

- *Device ID* identifie le périphérique lui-même. Il s'agit d'un modèle particulier défini par le vendeur.

- *Revision ID*: ce registre permet de connaître la version du périphérique.

- *Class Code* est utilisé pour identifier la fonctionnalité du périphérique (contrôleur de mémoire, de disque, carte multimédia, carte vidéo, ...)

- *Header Type*: il permet d'identifier le type de la zone de configuration. L'organisation de celle-ci est en effet différente si l'on est dans le cas d'un pont PCI ou d'une carte multifonctions (avec plusieurs zones de configuration).

Ces cinq registres ne sont accessibles qu'en lecture.

Command est un registre utilisé pour le contrôle des capacités du périphérique à générer ou à répondre à des opérations du bus PCI (agir en maître, accéder en mémoire ou dans le champ I/O, effectuer des tests de parité,..)

Status contient des informations relatives à des événements produits sur le bus (erreur de parité, interruption d'une transaction,...). Il permet également de préciser le temps de réponse du signal DEVSEL#.

Fonctions spéciales:

- *Latency Timer* est le temps minimum d'une transaction que garantit l'arbitre à l'agent qui a la maîtrise du bus. Au delà de ce temps, l'arbitre peut accorder la maîtrise du bus à un autre agent. Le maître actuel est alors tenu de finir la transaction en cours au plus vite.
- *Interrupt Line*: ce registre contient le numéro de l'interruption générée par le périphérique. Ce numéro est attribué par le contrôleur d'interruption (uniquement dans un PC).
- *Interrupt Pin* détermine la broche utilisée pour cette interruption (INTA#,..,INTD#). Seule INTA# peut être choisi dans le cas d'une carte non multifonctions.
- *MIN_GNT* spécifie la durée minimale d'un BURST.
- *MAX_LAT* indique la fréquence d'utilisation du bus.
- *BIST* sert à l'autotest des cartes du bus PCI.
- *Cache Line Size* indique la taille d'une ligne de cache.
- *Expansion ROM Base Address* précise la taille de la ROM présente sur la carte.

Adresses de base:

Une des plus importantes fonctions de l'autoconfiguration est sa capacité à attribuer à chaque périphérique une ou plusieurs zones mémoires qui lui sont réservées. Les six registres BADR disponibles dans la zone de configuration déterminent chacun l'adresse de base d'une zone mémoire. Un périphérique peut ainsi posséder six espaces mémoires distincts.

Avant l'initialisation effectuée par le BIOS du PC ou l'Open Firmware du PowerMAC, ces registres, accessibles en écriture et lecture, doivent contenir une information indiquant la taille mémoire à réserver ainsi que son emplacement (espace I/O, mémoire en dessous de 1 Mo, etc..).

Pendant l'initialisation, le système d'exploitation viendra y écrire les adresses de bases correspondant aux six zones du périphérique.

II.2.5 La version 2.1 des spécifications

Dans tous les ordinateurs actuels, seule la version 2.0 est utilisée. La version 2.1 apporte quelques compléments qu'il est intéressant de connaître si l'on veut améliorer les performances des cartes sur le bus PCI:

- deux vitesses d'horloge sont disponibles: 33 MHz et 66 MHz. Un signal supplémentaire, M66EN, indique si la carte peut fonctionner à 66 MHz. Si au moins une carte du bus ne peut fonctionner qu'à 33 MHz alors la fréquence d'horloge sera de 33 MHz. Il faut également remarquer que le fonctionnement à 66 MHz n'est autorisé que pour les cartes 3.3V compte tenu des contraintes que cette vitesse impose.
- deux registres supplémentaires ont été ajoutés en zone de configuration pour une meilleure identification des cartes: Subsystem ID et Subsystem Vendor ID.

III. L'INTERFACE PCI

III.1 Choix de l'interface

La réalisation d'une carte PCI nécessite l'obtention d'une interface PCI. Nous avons principalement deux voies: la conception d'une interface en logique programmable ou l'achat d'un circuit spécifique.

L'inconvénient de la conception en logique programmable vient des programmes fournis par les sociétés ALTERA ou XILINX qui n'ont pas été pleinement testés et qui demandent un développement et une mise au point fastidieux.

L'avantage du circuit spécifique est la facilité d'utilisation malgré un prix plus élevé. C'est pourquoi nous avons opté pour cette dernière solution.

Il existe, sur le marché, plusieurs interfaces PCI: la plupart d'entre elles sont d'utilisation complexe car elles disposent de nombreuses fonctionnalités, ou sont, au contraire, dédiées à des applications trop spécialisées. Néanmoins, nous avons trouvé chez la société AMCC, une interface (série S593X) qui correspond à nos besoins en terme de simplicité et de fonctionnalité.

III.2 Etude détaillée

III.2.1 Caractéristiques principales

Les composants de la série S593X de la société AMCC sont des interface souples et performantes.

La zone de configuration que tout agent du bus PCI doit posséder peut être totalement paramétrée par l'utilisateur grâce à la connexion d'une mémoire non volatile de faible capacité, série ou parallèle.

C'est une interface qui peut agir aussi bien en esclave qu'en maître, capable d'effectuer des transferts directement en mémoire à haut débit (jusqu'à 132 Mo/s) sans intervention du microprocesseur.

Le bus dont elle dispose est un bus simple se destinant à des applications variées. L'échange des données entre ce bus (que l'on appellera bus AMCC) et le bus PCI se fait par l'intermédiaire de registres internes de l'interface.

On peut distinguer trois méthodes de transfert possibles:

• Quand l'interface est la cible d'une transaction (accès dans unes des zones mémoires définies dans la zone de configuration), deux méthodes sont possibles:

- la première consiste à venir lire ou écrire dans des registres de l'interface dédiés à l'échange des données entre les deux bus. Il s'agit de registres boîtes aux lettres (Mailbox) et de FIFOS. Ces transferts asynchrones nécessite un contrôle assuré par d'autres registres internes.

L'ensemble de ces registres, accessibles sur les deux bus, se situe dans la première zone mémoire de l'interface (BADR0).

- la deuxième méthode appelé "mode Pass-Thru" permet à une transaction du bus PCI d'être répercutée en temps réel sur le bus AMCC. Cette méthode, qui accepte les données en rafale (BURST), peut être utilisée pour des transferts rapides et parfaitement contrôlés entre le bus PCI et les périphériques adressables présents sur le bus AMCC. Ce mode se produit lorsqu'un accès est réalisé dans une des quatre autres régions de l'interface BADR1 à BADR4 (l'interface ne peut utiliser BADR5) qui sont mis à la disposition de l'utilisateur.

Deux registres, uniquement accessibles sur le bus AMCC, sont utilisés: un pour le stockage de l'adresse et l'autre pour le stockage des données.

• La troisième méthode concerne toute opération de lecture ou d'écriture effectuée par l'interface lorsque celle-ci est maître. Il faut alors spécifier l'adresse destination ainsi que le nombre d'octets à transférer. Les données lues ou écrites transitent par les FIFOS décrits précédemment.

Quatre versions de l'interface sont disponibles:

- S5930: fonctionnement sur 16 bits uniquement et mémoire non volatile série.

- S5931: 16 bits et mémoire série ou parallèle.
- S5932: 32 bits et mémoire série.
- S5933: 32 bits et mémoire série ou parallèle.

Le schéma ci-dessous indique les principaux éléments fonctionnels de l'interface.

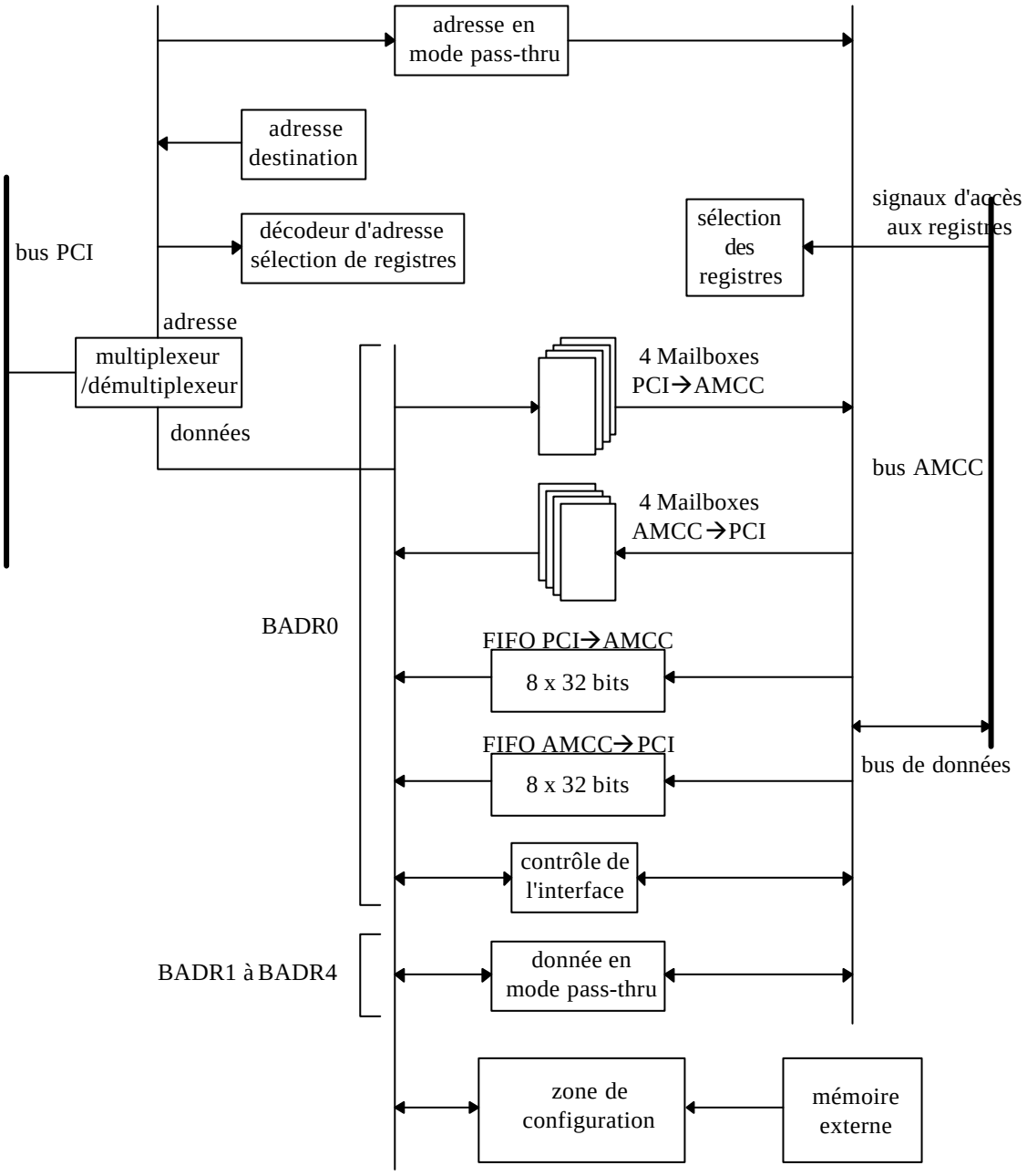


Figure III-1: synoptique de l'interface S593X

III.2.2 Description des signaux de l'interface

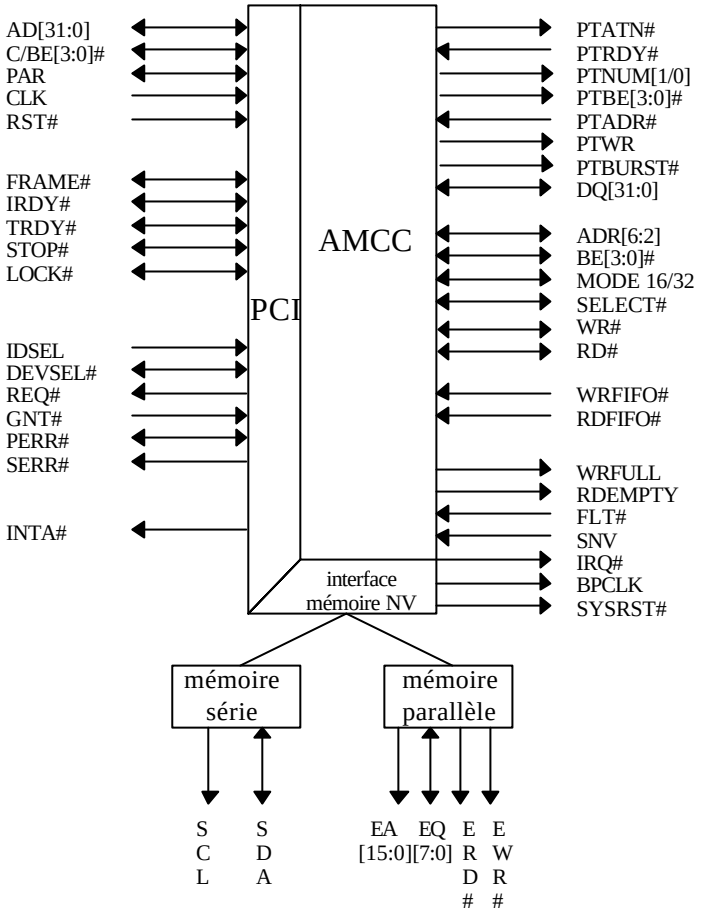


Figure III-2: les signaux de l'interface S5933

Mémoire série:

- SCL est l'horloge de la mémoire série.

- SDA est le bit utilisé pour transférer l'adresse et les données à la mémoire.

- SNV: au niveau haut, il indique la présence d'une mémoire série. Au niveau bas, il s'agit d'une mémoire parallèle.

Mémoire parallèle:

- EA[15:0]: signaux d'adresse de la mémoire parallèle.

- ERD#: signal de lecture.

- EWR#: signal d'écriture.

- EQ[7:0] sont les données lue ou écrites dans la mémoire.

Système:

- SYSRST# (System Reset) est l'équivalent de RST# présent sur le bus PCI. Il peut être également commandé par l'utilisation d'un des registres de l'interface.
- BPCLK est l'horloge CLK du bus PCI retardé de quelques nanosecondes.
- IRQ# est utilisé pour la génération d'interruption sur le bus AMCC.

Accès aux registres d'opérations:

- DQ[31:0] sont les données lues ou écrites dans les registres de l'interface.
- ADR[6:2] sont les lignes d'adresse qui sélectionnent le registre auquel on veut accéder (voir tableau ci-dessous)

ADR[6:2]	**Nom du registre**
0 0 0 0 0	Incoming Mailbox
0 0 0 0 1	Incoming Mailbox
0 0 0 1 0	Incoming Mailbox
0 0 0 1 1	Incoming Mailbox
0 0 1 0 0	Outgoing Mailbox
0 0 1 0 1	Outgoing Mailbox
0 0 1 1 0	Outgoing Mailbox
0 0 1 1 1	Outgoing Mailbox
0 1 0 0 0	FIFO
0 1 0 0 1	Bus Master Write Address
0 1 0 1 0	Pass-Thru Address
0 1 0 1 1	Pass-Thru Data
0 1 1 0 0	Bus Master Read Address
0 1 1 0 1	Mailbox Empty/Full Status
0 1 1 1 0	Interrupt Control

0 1 1 1 1	General Control/Status
1 0 1 1 0	Bus Master Write Count
1 0 1 1 1	Bus Master Read Count

Tableau III-1: adresse des registres de l'interface

- BE[3:0]# sélectionne les octets concernés dans l'accès à un des registres.

- SELECT# sélectionne l'interface. Ce signal doit être actif (au niveau bas) pour chaque lecture ou écriture (c'est l'équivalent d'un "Chip Select").

- WR# réalise l'écriture des données dans le registre choisi. Il est synchrone avec le front montant de BPCLK.

- RD# effectue la lecture d'un registre. C'est un signal de lecture asynchrone.

- MODE: ce signal détermine la largeur des données durant un accès à un registre (16 ou 32 bits). Si MODE est à l'état haut, les accès se font sur 32 bits.

Accès aux FIFOS:

L'interface possède deux FIFOS. Un dans le sens AMCC→PCI et l'autre dans le sens PCI→AMCC. Ces deux FIFOS se trouvent à la même adresse, la distinction entre les deux se faisant d'après le type de l'accès effectué (lecture ou écriture).

- WRFIFO# est le signal qui permet l'écriture directe du FIFO AMCC→PCI sans avoir besoin d'utiliser les signaux décrits précédemment (SELECT#, ADR, RD# et WR#).

- RDFIFO# permet de même la lecture directe du FIFO PCI→AMCC.

- WRFULL indique que le FIFO AMCC→PCI (écriture) est plein.

- RDEMPTY indique que le FIFO PCI→AMCC (lecture) est vide.

Signaux en mode "Pass-Thru":

Cette méthode de transfert utilise un protocole bien particulier (détaillé plus loin) faisant appel aux signaux suivants:

- PTATN# (Pass-Thru ATteNtion): il signale un accès en mode pass-thru dans une des régions de l'interface (BADR1 à BADR4, BADR0 étant réservé pour les registres d'opérations coté PCI).

- PTBURST# indique à l'état bas qu'il s'agit d'un transfert BURST (plus d'une phase de donnée).
- PTRDY# (Pass-Thru ReaDY) indique que l'on a réalisé la phase de donnée en cours et que l'on prêt pour la suivante (fonctionnement semblable à TRDY# du bus PCI).
- PTNUM[1:0] identifient la région à laquelle on accède. 00 correspond à BADR1 et 11 correspond à BADR4.
- PTBE[3:0]# indiquent quels octets sont valides pendant la phase de donnée en cours.
- PTADR# permet la lecture directe de l'adresse contenue dans APTA sans devoir utiliser les signaux classiques d'accès aux registres.
- PTWR indique le type d'opération (écriture si état haut).

III.2.3 Les registres d'opérations

L'interface S593X possède de nombreux registres de 32 bits servant aux transferts de données et au contrôle de l'interface.

On peut considérer deux ensembles de registres: ceux qui sont accessibles sur le bus PCI et ceux qui le sont sur le bus AMCC. Comme on pourra le voir, un certain nombre de ces registres sont accessibles sur les deux bus.

Les registres du bus PCI

Les registres du bus PCI se situent en zone mémoire. Cet espace mémoire est spécifié par l'adresse de base du registre BADR0 défini dans la zone de configuration de l'interface. Pour y accéder, il suffit de connaître cette adresse de base et son emplacement (champ I/O ou champ mémoire) ainsi que les offsets correspondant aux différents registres.

On dispose de 16 registres:

- **OMB1 à OMB4** (Outgoing Mailboxes): il s'agit de quatre registres boîtes aux lettres, accessibles en écriture, servant à l'envoi d'une commande ou d'un message vers le bus AMCC.

- **IMB1 à IMB4** (Incoming Mailboxes) sont également quatre boîtes aux lettres, accessibles en lecture, utilisées pour la réception d'une commande ou d'un message en provenance du bus AMCC.

- **FIFO**: ce registre permet l'accès à deux FIFOS. L'un de ces FIFOS est utilisé pendant les opérations d'écriture (PCI vers AMCC) et l'autre pendant les opérations de lecture (AMCC vers PCI).

Ces neuf registres sont également présents et accessibles sur le bus AMCC.

- **MWAR** (Master Write Address Register): ce registre contient l'adresse destination pendant les écritures de l'interface lorsque celle-ci fonctionne en maître (bus mastering write). Ce registre est constamment mis à jour pendant le processus de transfert. Il pointe toujours sur une zone non écrite.

- **MWTC** (Master Write Transfer Count) indique à l'interface le nombre d'octets restant à transférer pendant les opérations d'écritures en mode maître. La valeur de ce registre est décrémentée à chaque phase d'écriture. Lorsque l'on atteint zéro, le transfert cesse et une interruption peut être générée.

- **MRAR** (Master Read Address Register): ce registre contient l'adresse destination lorsque l'interface est maître du bus et qu'elle effectue une lecture (bus mastering read). Tout comme MWAR, ce registre est régulièrement mis à jour au cours du transfert et pointe sur une zone qui n'a pas encore été lue.

- **MRTC** (Master Read Transfer Count) indique à l'interface le nombre d 'octets restant à transférer pendant une lecture en mode maître. La lecture cesse lorsque ce registre, décrémentée par chaque phase de lecture, s'annule. Là aussi, une interruption peut être générée en fin de transfert.

- **MBEF** (Mailbox Empty / Full Status) permet de connaître l'état de chaque octet des huit registres boîtes aux lettres.

- **INTSCR** (Interrupt Control / Status Register): ce registre est utilisé pour la génération d'interruptions sur le bus PCI (broche INTA#). Il établit les conditions provoquant nue interruption (transaction interrompue, fin d'un transfert maître, remplissage d'un octet d'une boîte aux lettres,..) et permet de visualiser et d'annuler les différentes causes d'une interruption. Il permet également la conversion de structures "endian" avec les registres FIFOS.

- **MCSR** (Bus Master Control / Status Register) sert au contrôle global de l'interface. Ce registre permet la mise en œvre de transfert en mode maître, indique le taux de remplissage des deux FIFOS et peut réinitialiser les boîtes aux lettres et les FIFOS de même que les composants connectés au bus AMCC (avec le signal SYSRST#). Il sert également aux passages des données et d'adresses pour la programmation de la mémoire externe non volatile.

Les registres du bus AMCC

18 registres sont accessibles sur ce bus. Le choix du registre se fait avec les lignes d'adresses ADR[6:2]. Il suffit ensuite de sélectionner l'interface avec SELECT# et de réaliser une lecture avec RD# ou une écriture avec WR#. Deux de ces registres sont directement accessibles à l'aide de signaux particuliers: AFIFO avec RDFIFO# et WRFIFO# et APTA avec PTADR#.

- **AIMB1 à AIMB4**: ce sont les quatre boîtes aux lettres, accessibles en lecture, servant à la réception d'un message en provenance du bus PCI. Il s'agit des registres OMB1 à OMB4 du bus PCI.

- **AOMB1 à AOMB4**: ces quatre registres, accessibles en écriture, sont utilisés pour l'envoi d'un message vers le bus PCI. Ce sont les registres IMB1 à IMB4 du bus PCI.

- **AFIFO**: il s'agit du registre FIFO du bus PCI, accessible en lecture et écriture.

- **APTA** (AMCC Pass-Thru Address): ce registre contient l'adresse lorsqu'un accès est réalisé dans une des quatre zones définies par les adresses de base de la zone de configuration (provoquant un transfert en mode pass-thru).

- **APTD** (AMCC Pass-Thru Data): c'est dans ce registre que les données sont stockées lorsqu'un transfert en mode pass-thru intervient. On le lit si l'accès à l'interface est une écriture (PTWR=1) et on y écrit si cet accès est une lecture (PTWR=0).

- **AMBEF**: comme le registre MBEF, il indique l'état de chaque octet des huit registres boîtes aux lettres.

- **AINT**: il permet la gestion d'interruption sur le bus AMCC (avec IRQ#): il définit les conditions d'interruption et permet la visualisation et l'annulation de la cause d'interruption.

- **AGCSTS** (AMCC General Control / Status) est l'équivalent du registre MCSR pour le bus AMCC.

- Les registres **MWAR**, **MWTC**, **MRAR** et **MRTC** servant aux transferts en mode maître peuvent être accessibles sur le bus AMCC si l'interface se trouve dans un mode bien particulier concernant le "bus mastering", mode défini à l'initialisation dans la mémoire externe. Ces quatre registres ne sont alors plus accessibles sur le bus PCI.

III.2.4 Transfert par les registres d'opérations (FIFO et Mailboxes)

Transfert par boîtes aux lettres

L'interface S593X possède huit registres Mailbox de 32 bits. Ces registres sont généralement utilisés pour le passage d'une commande particulière ou d'information sur des états de composants du bus AMCC. Le coté PCI de l'interface a quatre registres pour l'envoi de données d'un bus à l'autre dans le sens PCI→AMCC (Outgoing Mailboxes ou OMB) et quatre autres pour la réception de donnée dans le sens AMCC→PCI (Incoming Mailboxes ou IMB). Le bus AMCC possèdent également ces huit registres (AIMB et AOMB). En interne, il s'agit exactement des mêmes registres. La correspondance des registres du bus PCI et du bus AMCC est indiquée dans le tableau suivant:

Bus PCI	**Bus AMCC**
Outgoing Mailbox 1 (OMB1)	Incoming Mailbox 1 (AIMB1)
Outgoing Mailbox 2 (OMB2)	Incoming Mailbox 2 (AIMB2)
Outgoing Mailbox 3 (OMB3)	Incoming Mailbox 3 (AIMB3)
Outgoing Mailbox 4 (OMB4)	Incoming Mailbox 4 (AIMB4)
Incoming Mailbox 1 (IMB1)	Outgoing Mailbox 1 (AOMB1)
Incoming Mailbox 2 (IMB2)	Outgoing Mailbox 2 (AOMB2)
Incoming Mailbox 3 (IMB3)	Outgoing Mailbox 3 (AOMB3)
Incoming Mailbox 4 (IMB4)	Outgoing Mailbox 4 (AOMB4)

Tableau III-2: correspondance des mailboxes

L'état des boîtes aux lettres peut être contrôlé de deux manières: soit en scrutant les bits du registre MBEF du bus PCI, soit avec le registre AMBEF du bus AMCC. Ces deux registres, identiques, indiquent les états (vide/plein) de chacun des octets des huit registres

Mailboxes. Les 32 bits de MBEF ou AMBEF représentent donc les états des 32 octets des boîtes aux lettres.

Ces états peuvent être réinitialisés avec le registre MCSR du bus PCI ou sur le bus AMCC avec le registre AGCSTS.

Les accès aux boîtes aux lettres se faisant de chacun des cotés de l'interface de manière totalement asynchrone, il faut toujours effectuer des vérifications avant une lecture ou une écriture:

- Lecture d'un registre IMB ou AIMB:
  - vérifier que les octets à lire ont reçus des données (MBEF).
  - si oui, on peut lire; si non, on attend.

- Ecriture d'un registre OMB ou AOMB:
  - vérifier que les octets où l'on va écrire les données sont vides (MBEF).
  - si oui, on peut écrire; si non, on attend.

Les boîtes aux lettres offrent également la possibilité de générer des interruptions sur les deux bus (INTA# sur le bus PCI et IRQ# sur le bus AMCC).

Chacune des interfaces PCI et AMCC peuvent en effet définir deux conditions d'interruptions relatives aux deux types de boîtes aux lettres. Une interruption peut être générée quand un octet d'un registre IMB ou AIMB (qui reçoit les données) est occupé (il vient de recevoir une donnée qu'il faut lire) et/ou lorsqu'un octet d'un registre OMB ou AOMB (qui envoie les données) est vide (la donnée qu'il contenait a été lue).

Le choix des octets provoquant les interruptions est fait dans les registres INTSCR (pour INTA#) et AINT (pour IRQ#).

Là encore, certaines étapes sont nécessaires:

- Mise en place de l'interruption (dans le registre INTSCR ou AINT)

- choisir un des registres Mailboxes.
- choisir l'octet de ce registre devant provoquer l'interruption.
- autoriser les interruptions dues aux boîtes aux lettres

• Traitement de l'interruption

- identifier la source de l'interruption dans INTSCR ou AINT.
- vérifier l'état des boîtes aux lettres dans MBEF ou AMBEF.
- accéder éventuellement à la boîtes aux lettres ayant provoquée l'interruption.
- remise à zéro de l'interruption dans INTSCR ou AINT.

Transfert par FIFO

L'interface S593X possède deux FIFOS. Le premier est utilisé pour les transferts du bus PCI vers le bus AMCC et le second pour les transferts du bus AMCC vers le bus PCI. Chacun d'entre eux a une capacité de 8 mot de 32 bits. Extérieurement, il s'agit du même registre car ils se situent à la même adresse (aussi bien sur le bus PCI que sur le bus AMCC). Le choix du FIFO est fait implicitement par l'interface en fonction de l'accès réalisé (écriture ou lecture).

Le FIFO peut être la cible mais aussi le maître d'une transaction. En tant que cible, il permet à un maître du bus PCI (le microprocesseur par exemple) d'avoir accès aux données du bus AMCC. La mise en œuvre de cette méthode est très semblable à celle décrite précédemment. Le fonctionnement en tant que maître est décrit dans le paragraphe concernant les transferts en mode maître (le bus mastering).

Ces registres se révèlent utiles dans le cas de transferts où les deux agents impliqués dans la transaction échangent un grand nombre de données à leur propre rythme, les FIFOS faisant office de mémoires tampons.

Sur le bus PCI, l'accès se fait de la même manière que les registres Mailboxes ou tout autre registre d'opération situé dans la première zone de l'interface (région BADR0).
Comme les boîtes aux lettres, la vérification du taux de remplissage de chaque FIFO est nécessaire afin de ne pas lire dans un FIFO vide (FIFO AMCC→PCI) ou écrire dans FIFO plein (FIFO PCI→AMCC). Pour cela, la lecture du registre MCSR, qui indique l'état des deux FIFOS, est indispensable.

Sur le bus AMCC, laccès s'effectue très simplement à l'aide de signaux spécialement prévus pour cela: RDFIFO# et WRFIFO# permettent respectivement lectures et écritures directes des FIFOS sans utilisation des signaux SELECT#, ADR[6:2], WR# et RD#.

De même, deux autres signaux, WRFULL et RDEMPTY, indiquent l'état des FIFOS, sans avoir à lire le registre AGCSTS qui, comme MCSR, indique les états des FIFOS. WRFULL signale que le FIFO AMCC→PCI est plein (on ne doit pas y écrire) et RDEMPTY signale que le FIFO PCI→AMCC est vide (on ne doit pas y lire).

La lecture et l'écriture directe des FIFOS peuvent être synchrone sur le front montant de BPCLK ou asynchrone. Ce mode de fonctionnement est déterminé au cours de la configuration de l'interface, dans la mémoire externe.

Les FIFOS possèdent également une caractéristique intéressante: celle de pouvoir faire une conversion de la structure "endian" des données. Dans une architecture de type INTEL, le poids des octets dans un mot croient avec l'adresse. Dans une architecture de type MOTOROLA, c'est l'inverse: le poids des octets dans un mot croient alors que l'adresse diminue. Ces deux structures portent respectivement le nom de "little endian" et "big endian". La figure suivante représente un mot de 4 octets (HH = octet de poids fort, LL = octet de poids faible) dans les deux structures:

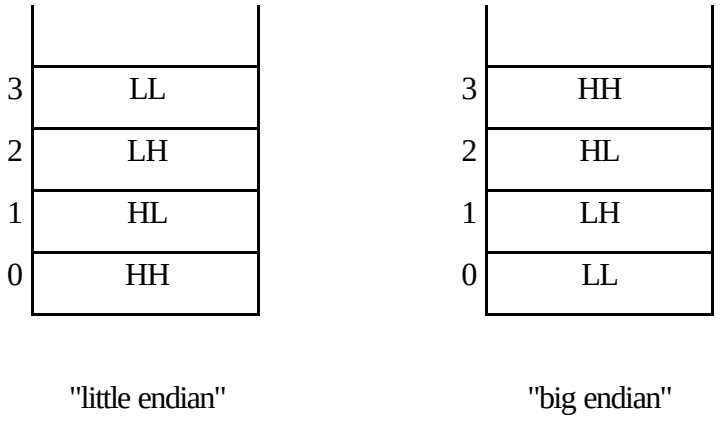


Figure III-3: structures little et big endian

Pour passer d'une structure à l'autre, il est nécessaire d'effectuer une inversion de l'ordre des octets dans un mot. Cette conversion est possible au travers des FIFOS. Pour la réaliser, il suffit d'utiliser le registre INTSCR qui indique le type de conversion employée (pas de conversion, conversion sur 16 bits, sur 32 bits, ..).

III.2.5 Mode Pass-Thru

Dans ce mode, l'interface permet au maître de la transaction d'accéder directement aux ressources des périphériques du bus AMCC par l'utilisation d'un protocole de transfert semblable à celui de bus PCI. Ce mode est très utile pour le traitement d'informations en temps réel et pour l'échange de données avec des mémoires ou des registres qui sont des composants adressables. Cela permet également de simplifier fortement l'accès au niveau du bus PCI puisque tous les contrôles sont effectués sur le bus AMCC grâce au protocole. Cette simplification accélère la rapidité des transferts.

Ce type de transfert ne peut être utilisé que lorsque l'interface est la cible d'une transaction et que cette transaction est destinée à une des quatre régions BADR1 à BADR4 définies dans la zone de configuration. Ces quatre régions représentent un espace mémoire occupé par les composants du bus AMCC. Elles ne sont utilisables qu'avec la présence de la mémoire externe non volatile. C'est elle qui contient les informations indiquant l'emplacement et la taille de la mémoire à réserver pour chaque région ainsi que la largeur du bus de donnée des

périphériques correspondants. Ce dernier paramètre n'est utilisé que par l'interface et non pas par le bus PCI car il permet la connexion de périphériques 8, 16 ou 32 bits sur le bus de donnée AMCC.

Le transfert des données implique l'utilisation de deux registres:

- APTA qui contient l'adresse, stockée pendant la phase d'adressage de la transaction effectuée sur le bus PCI.
- APTD par lequel passent toutes les données.

Description du protocole de transfert

Les signaux utilisés pendant un mode pass-thru sont les suivants:

- PTATN#, PTBURST#, PTNUM[1:0], PTWR et PTBE[3:0]# fournis par l'interface.
- SELECT#, ADR[6:2], BE[3:0]#, WR#, RD#, PTRDY# et PTADR# que l'on doit commander.

La démarche à suivre est la suivante (se reporter également aux chronogrammes qui suivent):

- le début d'une transaction est annoncé par l'état bas de PTATN#. Il indique également l'état du maître (prêt ou non) au cours d'une transaction en rafale (BURST).
- le signal PTBURST# indique un transfert BURST (au moins deux phases de données).
- PTNUM[1:0] identifie la région concernée par l'accès.
- lecture de l'adresse contenue dans le registre APTA avec le signal PTADR# qui permet un accès direct à ce registre.
- PTBE[3:0]# indiquent les octets du mot de 32 bits du registre APTD à lire ou écrire. Ils permettent aux périphériques de positionner BE[3:0]# qui opèrent un contrôle sur les octets du bus de données (voir accès 8 et 16 bits plus loin).
- PTWR indique si l'on fait une lecture ou une écriture.

⇒ si c'est une lecture, l'interface doit déposer ses données sur le bus PCI. Il faut alors écrire dans le registre APTD avec les signaux SELECT# et ADR[6:2] qui sélectionnent le

registre, et WR#. Cette écriture est synchrone sur le front montant de BPCLK et n'est valide que lorsque le maître et le périphérique du bus AMCC sont prêts, indiqué respectivement par PTATN# et PTRDY#.

⇒ si c'est une écriture, il faut lire le registre APTD avec RD#. Cette lecture est asynchrone. Les données lues ne sont valides que pendant l'état bas de RD# et seulement si le maître est prêt c'est à dire si PTATN# est à l'état bas. Là encore, le signal PTRDY# est nécessaire pour indiquer que le périphérique a lu la donnée (il est prêt pour la suite).

C'est le signal PTRDY# qui permet de passer à la phase suivante (nouvelle phase de donnée ou fin de la transaction).

- la fin d'une transaction est signalée par les états hauts de PTATN# et PTBURST#.

Les deux figures des pages suivantes illustrent les transactions en mode pass-thru:

Ecriture pass-thru:

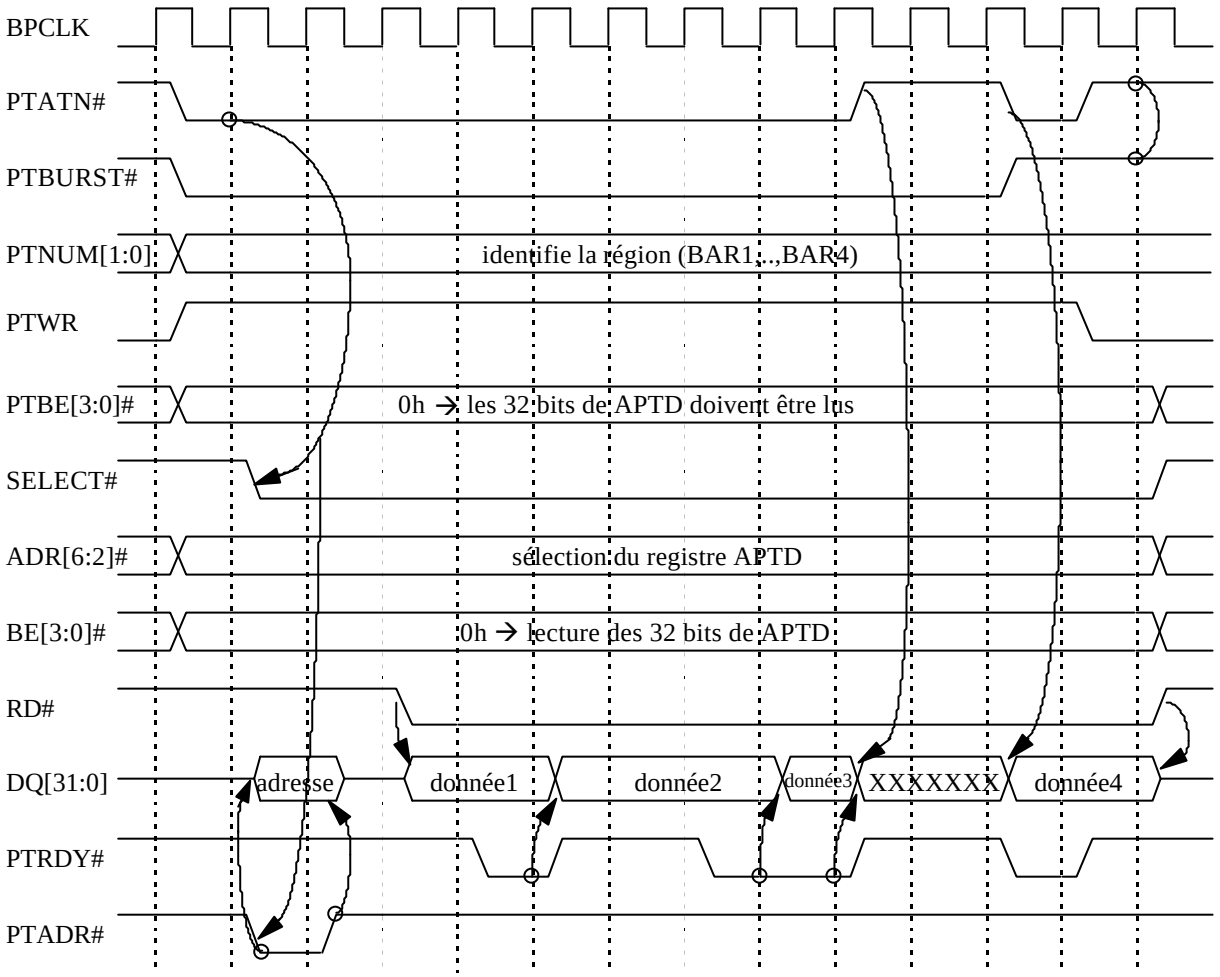


Figure III-4: écriture en mode pass-thru

Lecture pass-thru:

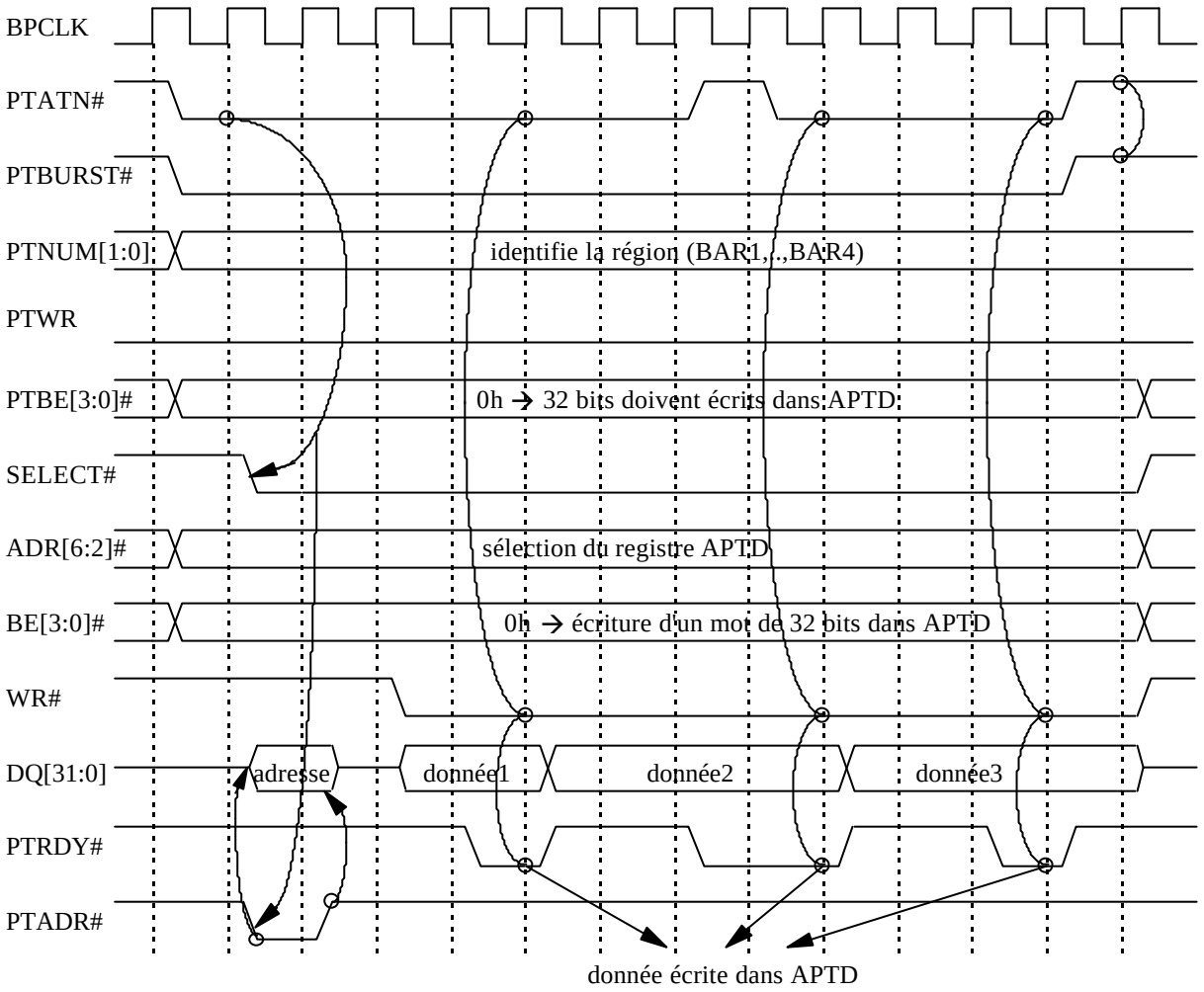


Figure III-5: lecture en mode pass-thru

L'utilisation des signaux PTBE[3:0]# et BE[3:0]# est assez complexe et demande une explication. Cette utilisation dépend de trois facteurs:

- la largeur du bus de donnée du périphérique auquel on accède.
- la taille des données.
- l'adresse des données.

Lorsqu'un périphérique ne fonctionne que sur 8 bits ou 16 bits, il est nécessaire de le placer dans une région dite de largeur 8 ou 16 bits. Dans ce cas, seuls les signaux DQ[7:0] ou DQ[15:0] du bus de donnée DQ sont utilisables pour le transfert des données. La connexion de ces signaux avec les octets du registre APTD est la suivante:

largeur du bus	registre APTD			
	OCTET3	OCTET2	OCTET1	OCTET0
32 bits	DQ[31:24]	DQ[23:16]	DQ[15:8]	DQ[7:0]
16 bits	DQ[15:8]	DQ[7:0]	DQ[15:8]	DQ[7:0]
8 bits	DQ[7:0]	DQ[7:0]	DQ[7:0]	DQ[7:0]

Tableau III-3: le registre APTD et le bus de donnée

Cette largeur de bus ainsi que la taille de la donnée à lire ou à écrire va déterminer le nombre d'étapes à réaliser. Par exemple, un mot de 32 bits va nécessiter quatre étapes sur un bus de 8 bits contre deux seulement pour un bus de 16 bits.

L'adresse intervient également à cause de la structure 32 bits du bus PCI. Toutes les données qui ne sont pas à des adresses 32 bits (bit A1 et A0 nuls) sont décalées dans le mot de 32 bits du bus de donnée PCI. L'exemple ci-dessous représente un mot de 16 bits à l'adresse 0001 et un octet à l'adresse 1011.

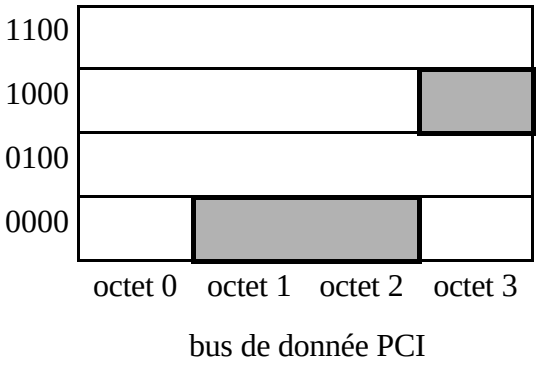


Figure III-6: exemple de données sur le bus PCI

PTBE[3:0]# et BE[3:0]# permettent de remédier à ces problèmes:

- PTBE[3:0]# indiquent les octets du registre APTD qu'il faut lire ou écrire.

- BE[3:0]# permet de réaliser des décalages dans le mot de 32 bits de APTD de manière à placer les octets indiqués par PTBE[3:0]# sur les bits de poids faible du bus de donnée DQ. Le tableau ci-dessous nous montre comment sont effectués ces décalages en fonction de la valeur de BE[3:0]#:

Byte Enable	registre APTD			
BE[3:0]#	DQ[31:24]	DQ[23:16]	DQ[15:8]	DQ[7:0]
x x x 0	OCTET3	OCTET2	OCTET1	OCTET0
x x 0 1	OCTET3	OCTET2	OCTET1	OCTET1
x 0 1 1	OCTET3	OCTET2	OCTET3	OCTET2
0 1 1 1	OCTET3	OCTET3	OCTET3	OCTET3
1 1 1 1	OCTET3	OCTET2	OCTET1	OCTET0

Tableau III-4: influence du Byte Enable

III.2.6 Transfert en mode maître: le "bus mastering"

Une des caractéristiques les plus importantes de l'interface est sa capacité à fonctionner sur le bus en tant que maître. Cette aptitude nous donne la possibilité de réaliser des transferts, sans intervention du microprocesseur, à la vitesse maximale du bus PCI, soit 132 Mo/s (débit crête). Pendant ce temps-là, le microprocesseur peut s'occuper d'autres taches si celles-ci ne demandent pas l'utilisation du bus PCI.

Le registre utilisé pour le passage des données est le registre AFIFO dont l'accès se fait exactement de la même manière que lorsque l'interface est esclave. Cependant sur le bus PCI, le fonctionnement est différent: c'est l'interface elle-même qui vide le FIFO sur le bus PCI s'il s'agit d'une écriture (bus master write) ou le remplit avec les données du bus s'il s'agit d'une lecture (bus master read).

Ces transferts nécessitent l'emploi de deux paramètres: l'adresse physique de la destination (une autre carte PCI, la RAM, etc..) et le nombre d'octets à transférer. Pour cela, quatre registres sont utilisés, deux par type d'accès:

- MWAR et MWTC pour les écritures.
- MRAR et MRTC pour les lectures.

Avant de commencer toute transaction, il faut définir les conditions de son départ qui portent sur le taux de remplissage du FIFO et autoriser les écritures et/ou les lectures en mode maître.

L'initialisation d'un tel transfert (stockage des adresses, du nombre d'octets, autorisation de transfert,..) peut être réalisé soit depuis le bus AMCC, soit depuis le bus PCI. Ce choix doit être déterminé avant l'installation de la carte dans la mémoire externe et ne peut donc pas changer en cours de fonctionnement. La première possibilité présente l'avantage d'accorder aux composants du bus AMCC une totale autonomie mais implique une gestion hardware assez élaborée alors que sur le bus PCI, cette initialisation peut se faire simplement à l'aide d'un programme.

"Bus mastering" depuis le bus PCI

Dans ce mode de transfert, les registres MWAR, MRAR, MWTC et MRTC ne sont accessibles que depuis le bus PCI. L'autorisation et les conditions de départ sont définis dans le registre MCSR.

L'initialisation doit suivre un enchaînement d'étapes bien précis:

1) Définir les conditions d'interruption dans INTSCR: une interruption peut en effet être générée sur le bus PCI (pas sur le bus AMCC) lorsqu'un transfert vient de se terminer (MRTC ou MWTC à zéro). Cette condition concerne les écritures et/ou les lectures.

2) Faire un reset des deux FIFOS dans le registre MCSR afin d'être sûr de leur état.

3) Etablir les conditions de départ du transfert dans le registre MCSR: il y a une condition dans le cas d'une écriture et une autre dans le cas d'une lecture.

Quand le transfert est autorisé, une écriture débute lorsqu'une ou plusieurs données ont été écrites dans le FIFO AMCC→PCI. Dans le cas d'une lecture, le transfert débute si au moins une place du FIFO PCI→AMCC est inoccupée.

4) Charger l'adresse dans MWAR ou MRAR.

5) Charger le nombre d'octets à transférer dans MWTC ou MRTC.

6) Autoriser le "bus mastering" en lecture ou en écriture dans MCSR.

Dès que le transfert débute, l'interface tente d'effectuer les transactions en mode BURST les plus longues possibles. Plusieurs cas de figure peuvent interrompre le transfert:
- le nombre d'octets à transférer a atteint zéro. C'est une fin normale.
- le FIFO AMCC→PCI est vide pendant une écriture.
- le FIFO PCI→AMCC est plein pendant une lecture.
- l'arbitre des transactions a retiré la possession du bus à la carte pour l'accorder (momentanément) à un autre agent PCI.

Il est nécessaire que les périphériques du bus AMCC accédant au registre AFIFO soient assez rapides pour éviter que le FIFO ne soit vide pendant une écriture, ou plein pendant une lecture, ce qui entraînerait une interruption du BURST et donc une diminution de la vitesse. La logique doit donc être rapide, cadencée par l'horloge du bus PCI (BPCLK) et les signaux d'écriture/lecture directe des FIFOS doivent être synchrones avec BPCLK.

"Bus mastering" depuis le bus AMCC

Ce fonctionnement n'est possible qu'avec l'utilisation de la mémoire non volatile série car il nécessite des signaux supplémentaires qui utilisent les broches d'accès à la mémoire parallèle (EQ4 à EQ7, EA14 et EA15).

La méthodologie que l'on doit appliquer pour ce mode de transfert est à peu près la même que celle indiquée précédemment. Les principales différences avec le "bus mastering PCI" sont les suivantes:

- les registres MWAR, MRAR, MWTC et MRTC ne sont accessibles que depuis le bus AMCC.
- six signaux supplémentaires sont utilisés pour faciliter le contrôle du transfert:
  - AMWEN (EQ6) et AMREN(EQ7) qui autorisent respectivement les transferts en écriture et lecture et remplacent les bits de MCSR utilisés à cet effet.
  - FWC# (EQ4) et FRC# (EQ5) qui permettent respectivement les resets du FIFO AMCC→PCI et du FIFO PCI→AMCC.
  - FWE (EA14) et FRF (EA15) qui indiquent respectivement deux autres états des FIFOS (différents de WRFULL et RDEMPTY) nécessaire pour ce type de transfert: l'état vide du FIFO AMCC→PCI (l'équivalent de WREMPTY) et l'état plein du FIFO PCI→AMCC (l'équivalent de RDFULL).
- il est possible d'interdire l'utilisation des registres MWTC et MRTC. Cela permet aux applications ne connaissant pas le nombre d'octets à transférer de faire du "bus mastering" sans avoir à mettre à jour constamment les registres MWTC et MRTC.

L'initialisation du "bus mastering AMCC" se fait de la manière suivante:

1) Autoriser / interdire les registres MWTC et MRTC dans AGCSTS.

2) Conditions d'interruptions dans AINT.

3) Resets des FIFOS avec FRC# et FWC# ou dans AGCSTS.

4) Etablir les conditions de départ dans MCSR (bus PCI).

5) Ecriture de l'adresse dans MWAR ou MRAR.

6) Ecriture éventuelle dans MWTC ou MRTC.

7) Autorisation du transfert avec AMREN ou AMWEN.

IV. CONCEPTION DE LA CARTE

Le schéma synoptique de la carte est le suivant:

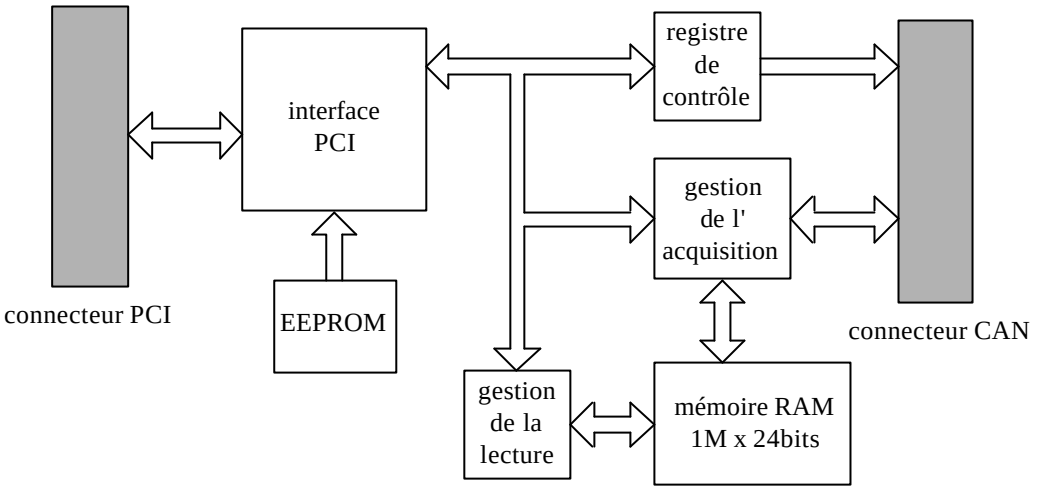


Figure IV-1: synoptique de la carte d'acquisition

La carte possède essentiellement deux modes de fonctionnement.

Le premier de ces modes concerne l'acquisition de données en provenance du CAN relié à la carte par l'intermédiaire d'un connecteur et d'un câble en nappe. L'acquisition proprement dite doit être précédée du chargement de la taille de l'acquisition, suivi de la commande de départ. Lorsque l'acquisition est terminée, un bit d'état est activé indiquant au logiciel que l'on peut lire la mémoire.

Le deuxième mode est la lecture des données stockées dans la mémoire RAM de la carte. Cette lecture se fait à travers le bus PCI via l'interface.

L'EEPROM sert à la configuration de l'interface et à l'autoconfiguration de la carte dans l'ordinateur.

IV.1 L'interface PCI

Comme cela a été dit dans le chapitre III, l'interface qui doit fonctionner en esclave ne peut utiliser que les méthodes d'accès suivantes: les registres Mailbox, les FIFOS ou le mode Pass-Thru.

Les deux premières méthodes imposent un contrôle élaboré de la part du logiciel qui pilote la carte et de l'électronique présente sur celle-ci. Ces deux méthodes ne sont intéressantes qu'avec la présence d'un automate complexe ou d'un microprocesseur.

C'est pourquoi nous avons préféré le mode Pass-Thru. Le protocole que nous utilisons est le même que celui du chapitre III:

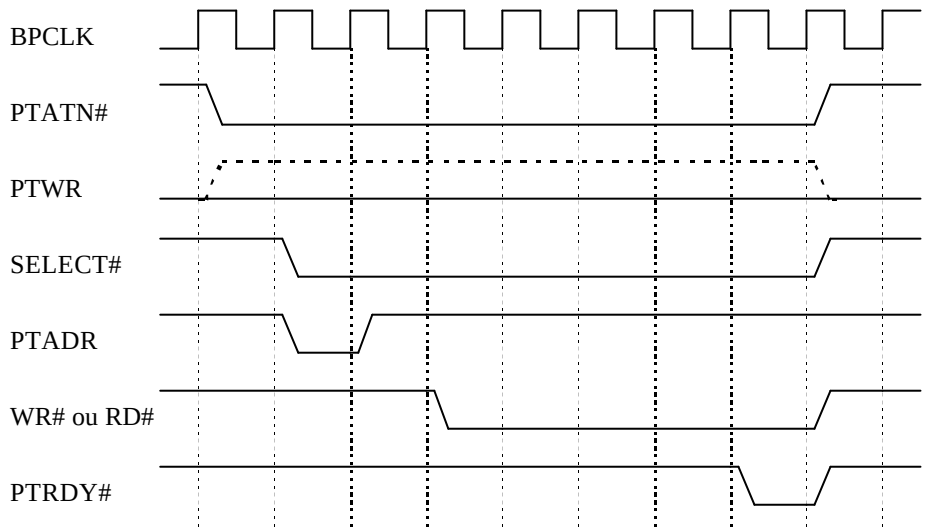


Figure IV-2: protocole pass-thru utilisé sur la carte

Ce protocole a été réalisé avec une machine d'état en langage AHDL (Altera Hardware Description Language) implantée dans un ALTERA 7032LC44 (voir **alt_pass.tdf** en annexe):

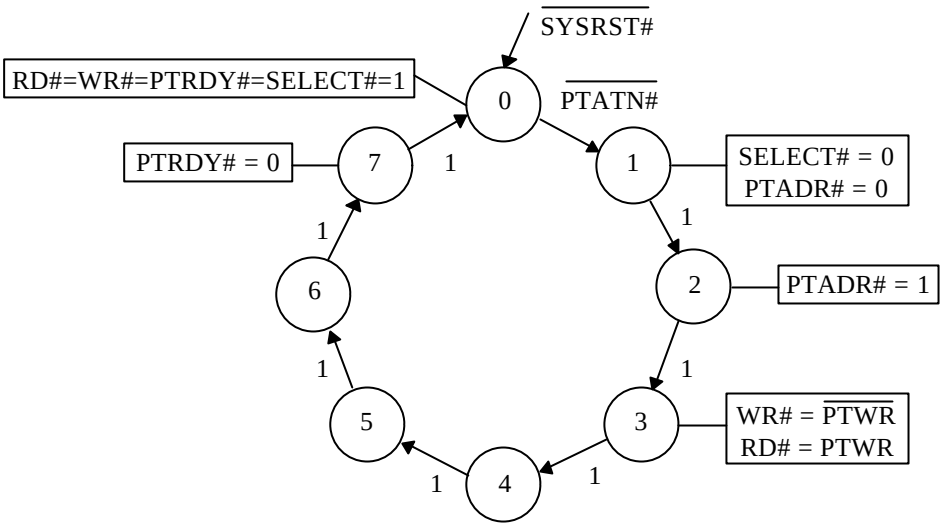


Figure IV-3: diagramme d'état pour le mode pass-thru

Dès qu'un accès débute (PTATN# à 0), l'adresse est lue avec le signal PTADR#. Dans notre cas, la connaissance de cette adresse n'est pas nécessaire puisque chaque région est identifiée par PTNUM[1:0]. En fonction du type d'accès (lecture ou écriture indiqué par PTWR), on réalise une lecture ou une écriture avec les signaux WR# ou RD#. La transaction se termine avec l'activation du signal PTRDY#.

Des états d'attente ont été introduits (états 4 à 7) pour s'assurer de la validité des données bien que cela limite un peu la vitesse de transfert sur le bus PCI.

Cet ALTERA génère aussi les "chip select" du registre de contrôle, du registre de chargement du compteur et de la RAM (voir plus loin). Les deux signaux PTNUM0 et PTNUM1 identifient l'espace mémoire adressé de la manière suivante:

Adresse de base	PTNUM 1	PTNUM 0	Chip Select
BADR1	0	0	registre de contrôle
BADR2	0	1	registre de chargement
BADR3	1	0	RAM
BADR4	1	1	RAM

Tableau IV-1: adresse des registres de la carte

Pour faciliter la sélection de la RAM, celle-ci peut être indifféremment adressée à partir de l'une ou l'autre des deux dernières adresse de base. Sur la carte, ces deux adresses pointent en effet sur la même zone (la RAM).

IV.2 L'autoconfiguration de la carte

L'autoconfiguration se fait au travers de l'interface grâce à l' utilisation d'une mémoire non volatile qui contient tous les paramètres exigés pour la configuration sur le bus PCI:

Paramètres	**Description**
Vendor ID	identification du vendeur
Device ID	identification du périphérique
Bus Master Config	configuration bus mastering
Revision ID	numéro de version du périphérique
Class Code	fonction du périphérique
Latency Timer	temps possession du bus PCI
Base Address Registers	réservation des zones mémoires
Expansion ROM Register	réservation d'une zone ROM
Interrupt Line	numéro de l'interruption
Interrupt Pin	broche d'interruption
Min_Gnt	durée minimale d'un BURST
Max_Lat	fréquence des BURST

Tableau IV-2: les paramètres de l'autoconfiguration

Les lignes grisées ne concernent que la configuration interne de l'interface et non la configuration sur le bus PCI.

Deux types de mémoire non volatile peuvent être connectés à l'interface:

- mémoire série (EEPROM, ..)
- mémoire parallèle (PROM, FLASH RAM, ..)

L'avantage de la mémoire série tient dans sa petite taille. Celui de la mémoire parallèle est sa rapidité et sa grande capacité permettant d'y stocker un BIOS propre à la carte.

Notre choix s'est porté sur la mémoire série car nous n'avons pas besoin d'une très grande capacité. En effet, seuls les paramètres de configuration nous sont indispensables. D'autre part, le nombre important de composants sur la carte limite la place disponible.

La programmation peut être accomplie de deux manières. La première consiste à utiliser le programme NVBUILD.EXE fourni avec le kit de développement AMCC. Ce programme permet de générer le fichier binaire (fichier .bin) ou hexadécimal (fichier .hex) image du contenu de la mémoire non volatile et de programmer cette dernière directement sur la carte. L'avantage de cette méthode est la facilité de programmation puisqu'aucun matériel particulier n'est requis.

La deuxième fait appel au logiciel ALLPRO utilisé pour la programmation de n'importe quel type de mémoire programmable à partir d'un fichier binaire ou hexadécimal.

IV.3 Le connecteur du convertisseur

Ce connecteur nous fournit les données du convertisseur et une horloge pour leur synchronisation. Il nous permet également de configurer et d'alimenter la carte sur laquelle se trouve le CAN.

Le connecteur choisi est un SUB-D femelle 62 broches haute densité avec la distribution suivante:

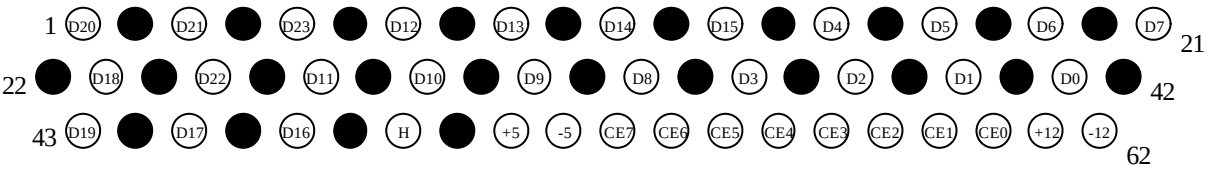


Figure IV-4: connecteur CAN

Les contraintes qui nous ont imposé ce choix sont:

- le grand nombre de signaux véhiculés.
- la taille réduite du connecteur fixée par les dimensions de l'ordinateur.
- la robustesse du connecteur.

Bien que le nombre de signaux utiles ne dépasse pas 39, l'insertion de fils de masses entre chaque bit de donnée augmente considérablement le nombre de broches du connecteur.

La présence de ces masses est nécessaire si l'on veut atteindre 100 MHz, diminuant l'effet de couplage entre les fils(voir figure ci-dessous).

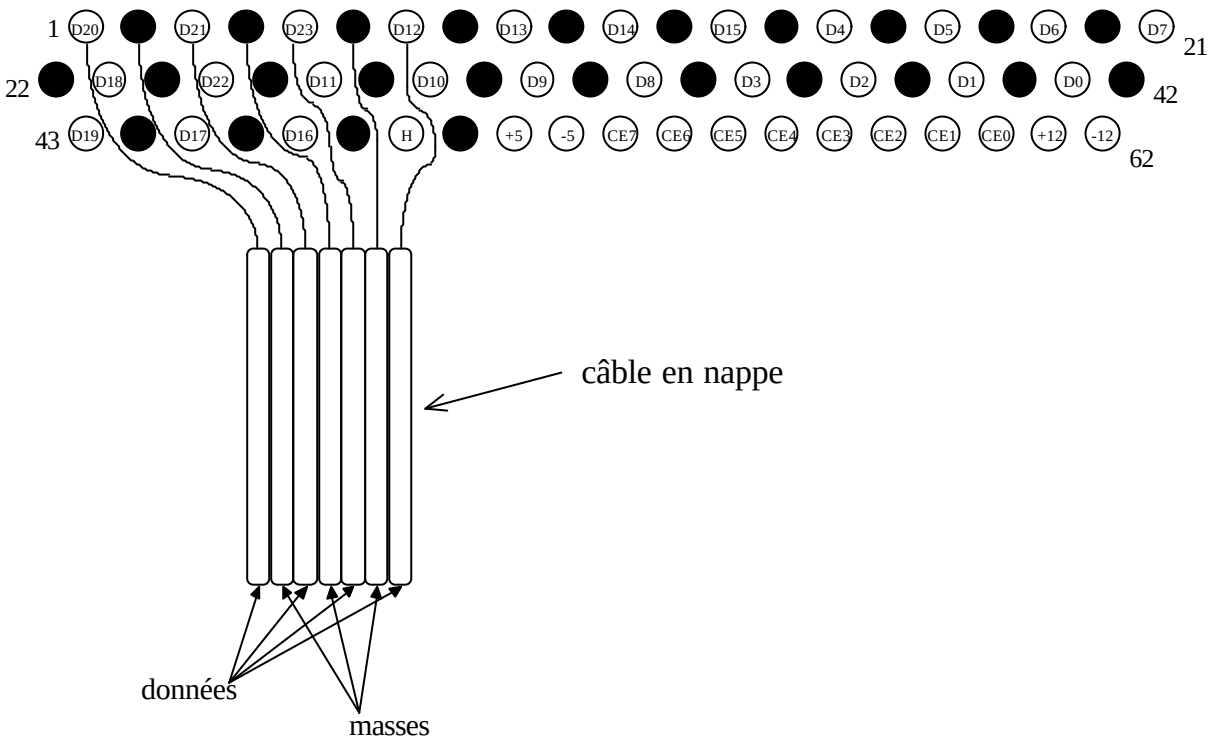


Figure IV-5: distribution données/masses sur le connecteur

IV.4 La mémoire RAM

La mémoire présente sur la carte doit pouvoir stocker 1 M échantillons de 24 bits ce qui représente une capacité de 3 Mo. Pour cela, la mémoire sera constituée de 24 boîtiers SRAM de 128 Ko 15ns.

Les accès en lecture et en écriture diffèrent radicalement à cause de la très haute fréquence d'acquisition qui peut être largement supérieure à la fréquence du bus PCI. Cela nous a amené à développer une structure particulière indiquée sur la figure suivante:

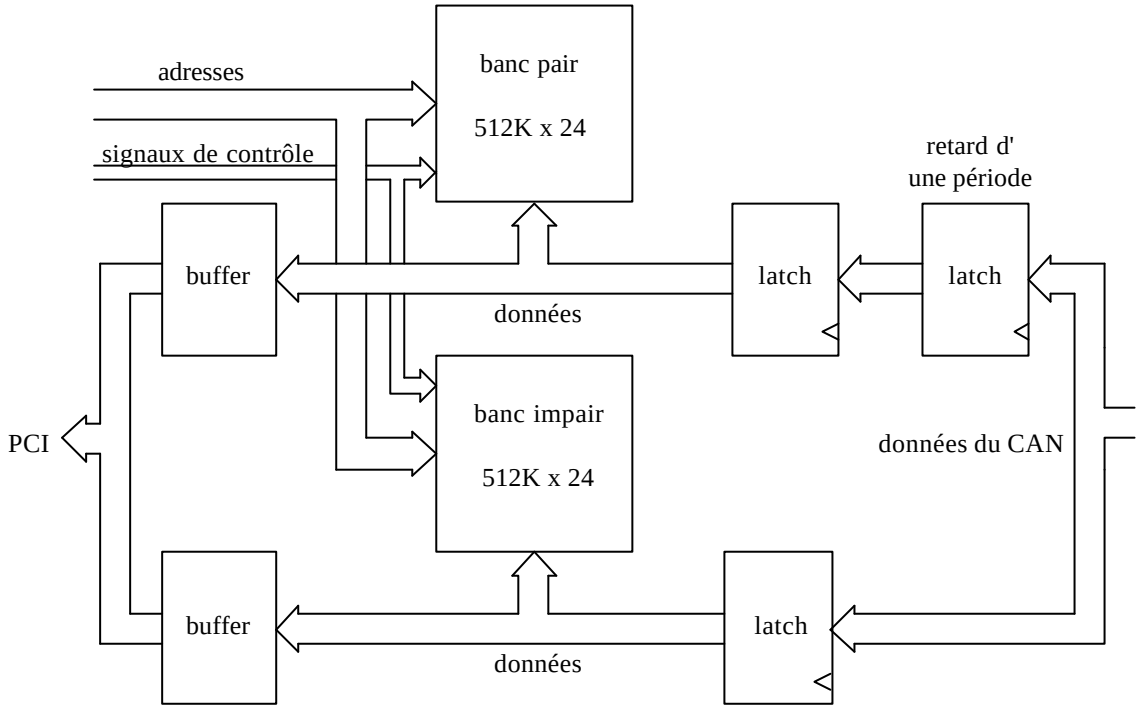


Figure IV-6: schéma bloc de la RAM

Les bancs pair et impair sont organisés selon le schéma de la page suivante:

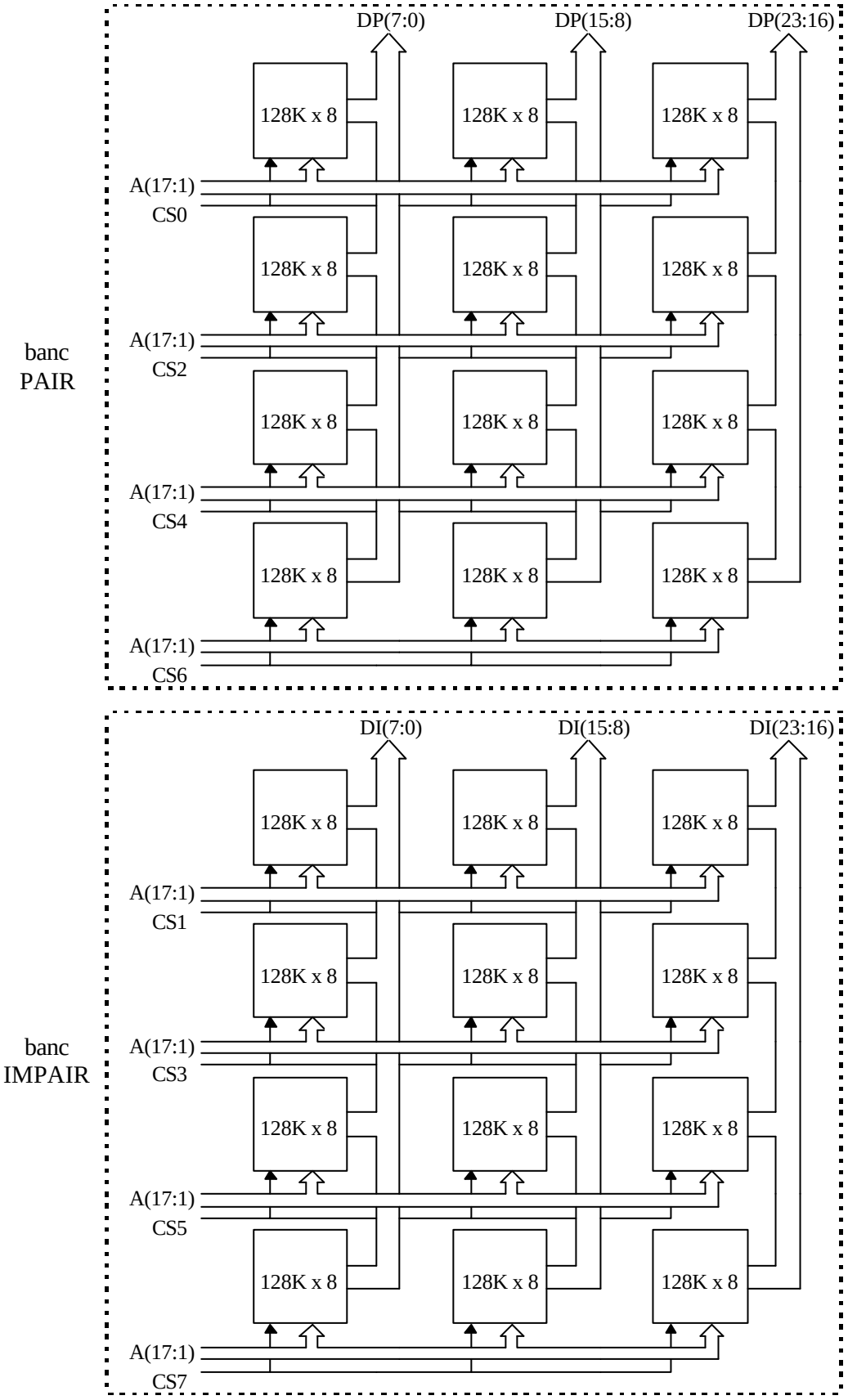


Figure IV-7: structure de la RAM

L'acquisition

La vitesse maximale d'acquisition étant de 100 MHz (10ns de période), le temps d'accès de notre mémoire (15ns) est insuffisant. Pour remédier à ce problème, deux écritures simultanées seront effectuées. Pour cela, il faut diviser la mémoire en deux blocs: un banc impair et un banc pair. Le cycle d'accès passe alors à 20ns ce qui est désormais satisfaisant.. L'inconvénient d'un tel découpage est la duplication du bus d'adresse et du bus de donnée rendant la réalisation de la carte bien plus complexe.

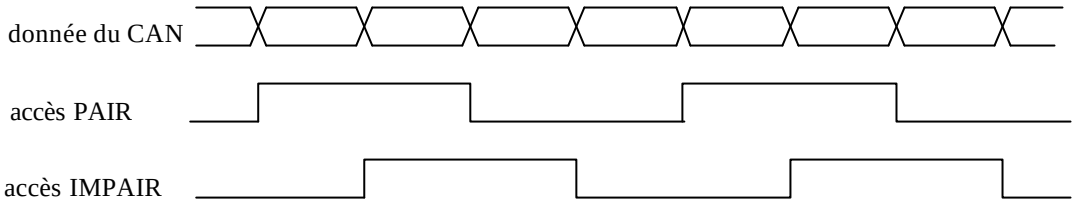


Figure IV-8: double accès en mémoire (ping pong)

Il est cependant possible de conserver le même bus d'adresse: il suffit en effet de retarder d'une période d'horloge la donnée du bus pair de manière à la synchroniser avec la donnée du bus impair. Les deux données peuvent alors être toutes deux écrites dans un même cycle avec le même signal d'écriture.

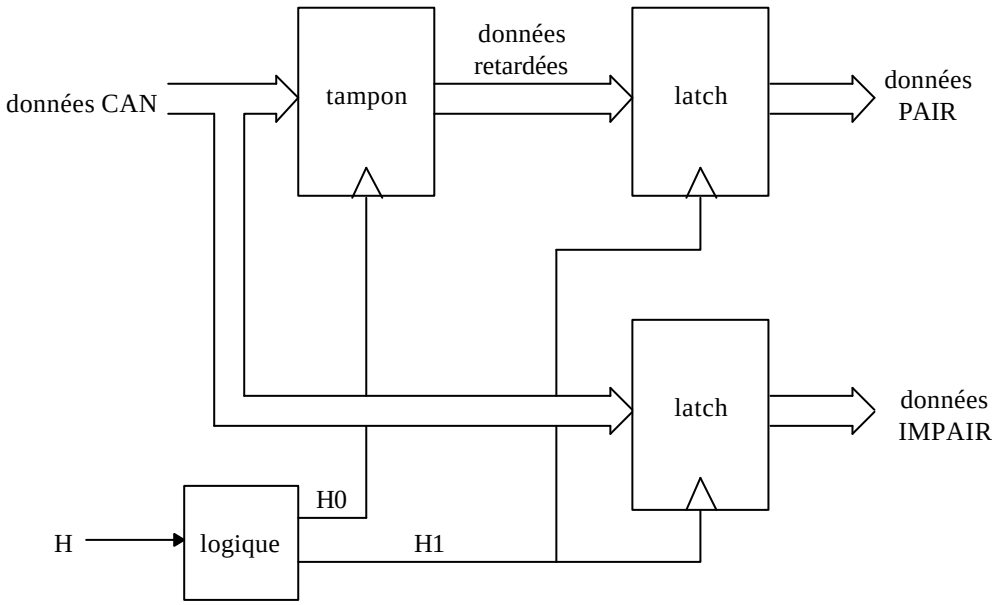


Figure IV-9: dédoublement du bus de donnée

Sur la figure ci-dessus, deux latchs sont utilisés pour la séparation des données de l'acquisition en données paires et impaires tandis qu'un troisième les synchronise. La forme des signaux obtenus est la suivante:

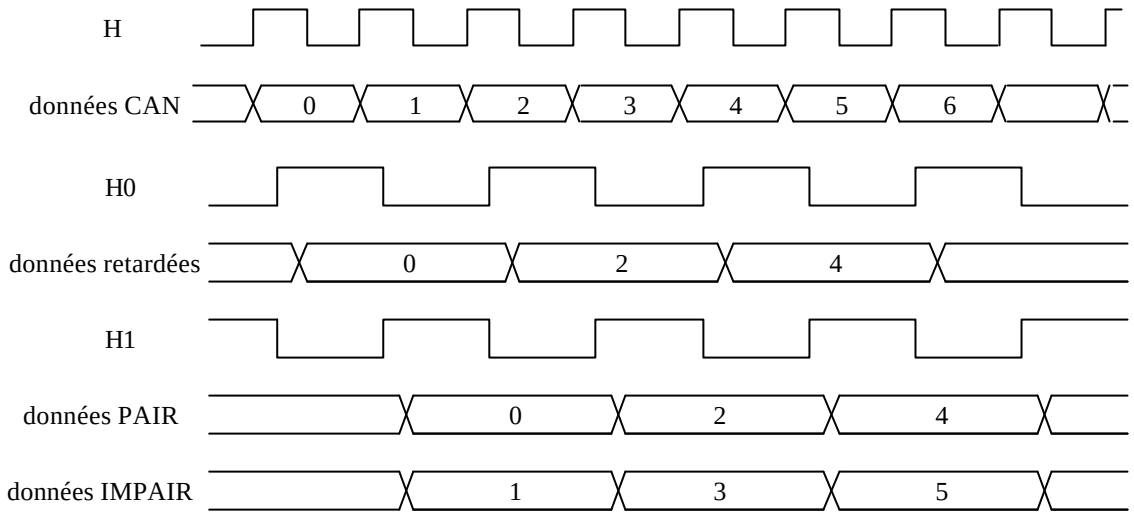


Figure IV-10: double accès en mémoire synchronisé

La génération des adresses se fait à l'aide d'un compteur programmable. Le chargement de ce compteur permet de modifier la taille de l'acquisition.

Le début de l'acquisition commence avec l'écriture du bit ON-OFF du registre de contrôle qui autorise le comptage. Ce registre et le registre de chargement seront détaillés dans le paragraphe suivant.

La grande vitesse d'acquisition a nécessité la mise en forme d'un signal d'écriture dissymétrique:

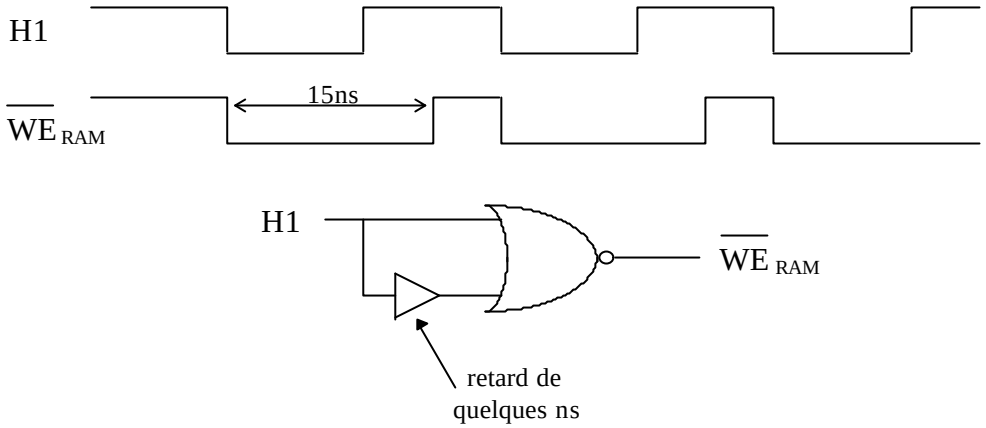


Figure IV-11: signal d'écriture

Pendant l'acquisition, la sélection du banc pair et du banc impair est identique: CS0=CS1 ou CS2=CS3 ou CS4=CS5 ou CS6=CS7.

La gestion de l'acquisition (compteur, signal d'écriture, génération des horloges,...) a été implantée dans un ALTERA 7096LC84 (voir **alt_comp.gdf** en annexe).

Lecture de la mémoire après l'acquisition

Contrairement au cas de l'écriture, on ne peut lire qu'un seul mot à la fois.

Dans une première approche, nous avions envisagé la lecture directe de la mémoire (réservation de 4 Mo dans l'ordinateur). La complexité de l'accès dans un PC sous Windows (mode réel, mode protégé) nous a forcé à changer de stratégie.

La méthode que nous avons finalement choisie consiste à lire toujours à la même adresse, la carte se chargeant de générer elle-même les adresses de la mémoire grâce au compteur utilisé précédemment.

Après l'acquisition, le compteur est automatiquement chargé avec la même valeur programmée pour l'acquisition. Il n'est donc pas utile de recharger le compteur pour réaliser la lecture.

La lecture ne doit être faite qu'à la fin de l'acquisition indiquée par le bit ON-OFF du registre de contrôle. Pendant l'acquisition, la lecture de la mémoire n'est pas prise en compte.

Pour la lecture, un seul "chip select" est activé. CS0, CS2, CS4 ou CS6 sont actifs pour les adresses paires et CS1, CS3, CS5 ou CS7 sont actifs pour les adresses impaires.

La génération de ces huit signaux et du signal de lecture de la RAM se situe dans l'ALTERA **alt_pass.gdf** (voir annexe). La gestion de la lecture est effectuée dans l'ALTERA **alt_comp.gdf** (voir annexe).

IV.5 Les registres de la carte

Mis à part les registres de l'interface, nous avons réservé trois registres dans l'espace d'entrée/sortie du PC. Il s'agit, pour les deux premiers, d'un registre de contrôle et d'un registre de chargement de la taille de l'acquisition situés respectivement aux adresses spécifiées par les registres d'adresses de base 1 et 2 situés dans la zone de configuration de la carte.

Le troisième registre est utilisé pour la lecture de la mémoire. Ce registre est situé aux adresses de base 3 et 4. L'accès peut se faire indifféremment dans l'une ou l'autre de ces zones, le registre y étant dupliqué.

En raison de l'autoconfiguration, ces adresses de base sont susceptibles de changer en fonction de l'ordinateur et des cartes connectées. C'est pourquoi le logiciel d'accès à ces registres devra en tenir compte.

Le registre de contrôle

Ce registre de 32 bits ne possède actuellement que 10 bits utiles ce qui permettra d'ajouter d'autres fonctions aux futures versions de la carte.

Le bit ON-OFF indique le début et la fin de l'acquisition. Il est positionné à ON par l'utilisateur et mis automatiquement à OFF à la fin de l'acquisition.

Le bit FRONT permet d'inverser l'horloge provenant du CAN. Le but de cette opération est de synchroniser l'acquisition sur le front montant ou descendant de l'horloge originale.

Ces deux bits sont stockés dans deux bascules D situées dans l'ALTERA servant à la gestion de l'acquisition (**alt_comp.gdf**).

Les huit derniers bits sont utilisés pour la configuration des convertisseurs connectés à notre carte. Ces bits sont mémorisés avec l'aide d'un latch 74F374.

	0							7		29	30	31
adresse de	CE0	CE1	CE2	CE3	CE4	CE5	CE6	CE7	XXXXXXXXX	ON/OFF	FRONT	X
base 1	W	W	W	W	W	W	W	W	XXXXXXXXX	R/W	R/W	X

Figure IV-12: registre de contrôle

Le registre de chargement

Ce registre détermine la taille de l'acquisition. La génération des adresses par un compteur pendant l'acquisition nous impose le stockage dans ce registre de la différence entre la taille maximale d'acquisition (1M x 24 bits) et la taille désirée.

Registre de chargement = 1M - taille désirée

Exemple: acquisition de 100 kilo-échantillons de 24 bits. La valeur à charger sera de 1M - 100k soit 900000 (en décimal).

	0	1		18	19	20 31
adresse de	C0	C1		C18	C19	X X X X X X X X X X X X
base 2	W	W		W	W	X X X X X X X X X X X X

Figure IV-13: registre de chargement

Il n'est pas nécessaire de procéder à une recharge systématique de ce registre avant chaque acquisition, la taille de l'acquisition étant mémorisé dans des bascules D de l'ALTERA de gestion de l'acquisition.

La mise en œvre de l'a cquisition doit suivre une procédure utilisant les registres de contrôle et de chargement indiquée par l'organigramme suivant:

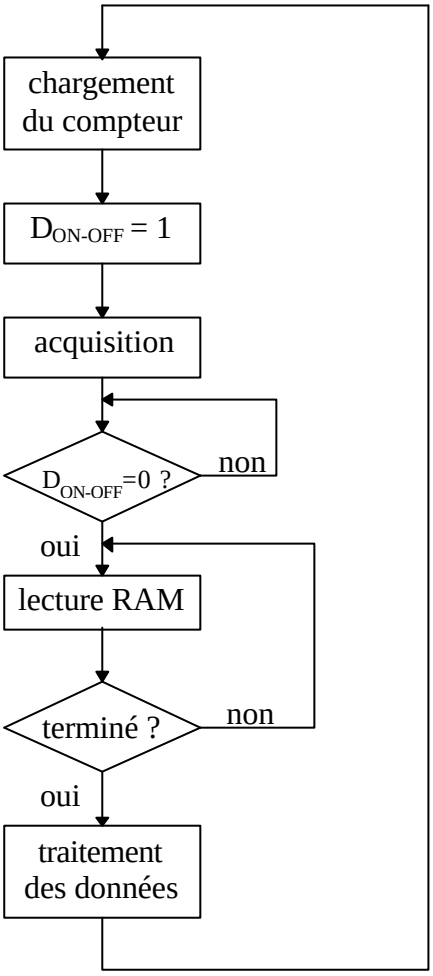


Figure IV-14: organigramme de l'acquisition

V. REALISATION ET TEST DE LA CARTE

V.1 Le prototype wrappé

Il nous a semblé intéressant d'utiliser le kit de développement PCI de la société AMCC pour un test préliminaire sur PC.

Ce kit est constitué d'une carte PCI sur laquelle se trouve l'interface S5933 et des connecteurs pour le développement de prototypes. Le synoptique de ce kit est le suivant:

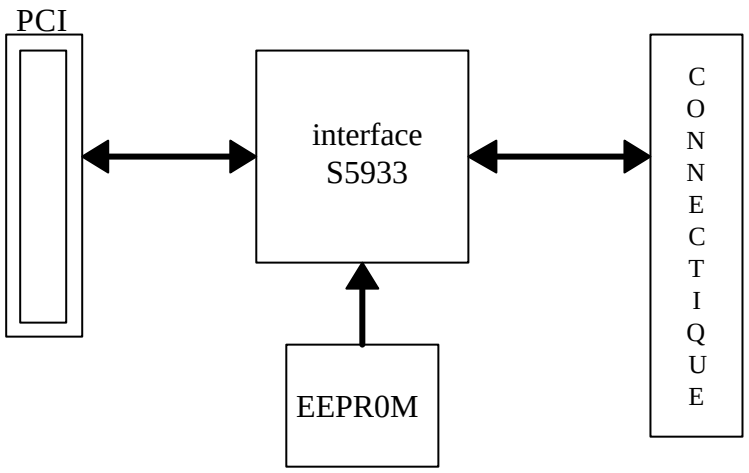


Figure V-1: synoptique du kit PCI

La société AMCC fournit également quelques programmes pour faciliter l'utilisation du kit sur PC:

- programmation de l'EEPROM (NVBUILD.EXE).
- accès aux registres de configuration (AMCCDIAG.EXE).
- accès aux registres de l'interface (AMCCDIAG.EXE).

Le prototype wrappé que nous avons connecté au kit (voir figure ci-dessous) met en œuvre toutes les fonctionnalités de la carte d'acquisition PCI et nous a permis d'en effectuer la mise au point.

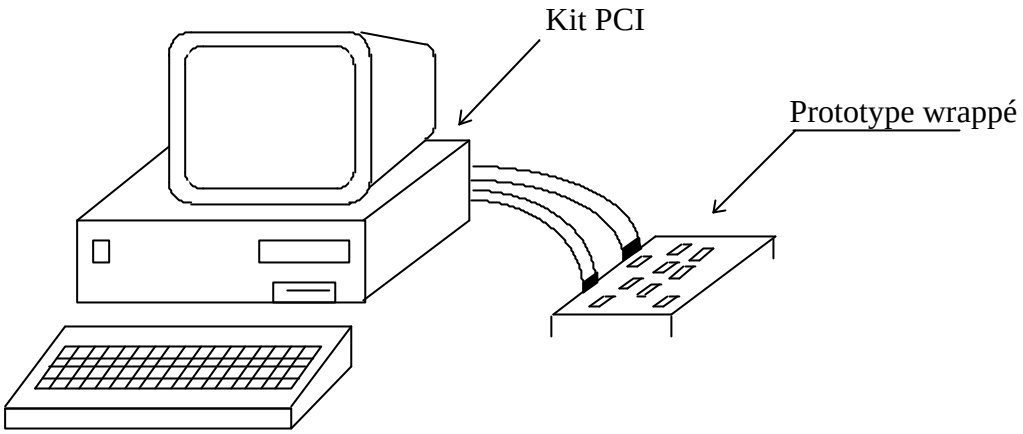


Figure V-2: connexion du prototype au kit PCI

Les tests que nous avons réalisés portent sur les points suivants:

- autoconfiguration de la carte.
- accès aux registres de l'interface.
- écriture du registre de contrôle.
- chargement de la taille de l'acquisition.
- déclenchement de l'acquisition.
- indication de fin d'acquisition.
- stockage des données pendant l'acquisition.
- lecture du registre de contrôle de la carte.
- lecture des données stockées en mémoire.

Plusieurs acquisitions successives, de taille et de fréquence variables, ont été réalisées avec succès malgré quelques erreurs que nous attribuons à l'utilisation de longs câbles en nappe pour la connexion de la carte wrappée au kit.

Nous avons en effet constaté que les signaux de la carte wrappée sont bruités alors qu'ils sont propres après la déconnexion des câbles en nappe. Le prototype wrappé, nécessairement placé à l'extérieur de l'ordinateur nous impose l'utilisation de câbles en nappe d'environ 50 cm introduisant des retards, des couplages entre signaux et captant toutes les perturbations externes.

Le test complet de la carte à la fréquence d'acquisition de 100 MHz ne pourra donc être réalisé qu'avec la carte définitive, après le routage.

V.2 Routage de la carte

La complexité de la carte nous impose le choix d'une carte huit couches et un routage en classe 4.

Plusieurs essais en routage automatique ont été accomplis mais sans succès: seulement 20% au grand maximum de la carte a été routé. Cette limitation et le fonctionnement de la carte à 100 MHz nous contraint à faire le routage à la main.

Une attention particulière a été portée sur le découplage et le filtrage ainsi que sur tous les signaux sensibles (notamment les horloges). Le placement des composants a également été soigné puisqu'il a été fait manuellement pour l'optimisation du routage.

Cependant, en raison de la complexité et des caractéristiques de la carte, il aurait été préférable de faire appel à un spécialiste pour le routage de la carte entière.

VI. LES LOGICIELS

VI.1 PC

Dans les PC, le BIOS (Basic Input Output System) dispose de plusieurs fonctions facilitant l'accès logiciel aux cartex PCI. La liste des fonctions généralement disponibles dans les BIOS PCI sont les suivantes:

Fonction	Description
Test for PCI BIOS present	vérifie la présence d'un BIOS pour PCI
Find PCI device	recherche d'une carte
Find PCI class code	recherche d'un type de carte
Generate special cycle	génération d'un "Special Cycle"
Read configuration byte	lecture d'un octet en zone de configuration
Read configuration word	lecture d'un mot
Read configuration dword	lecture d'un mot long
Write configuration byte	écriture d'un octet en zone de configuration
Write configuration word	écriture d'un mot
Write configuration dword	écriture d'un mot long

Tableau VI-1: les fonctions du BIOS PCI

L'appel de ces fonctions se fait avec l'interruption logicielle INT 1Ah et le choix de la fonction s'effectue avec les valeurs de deux registres du microprocesseurs, AH et AL.

Le programme PCI.EXE que nous avons développé (Visual C++) utilise certaines des fonctions du BIOS. Ce programme nous a permis de faire la mise au point de la carte et constitue un exemple d'accès logiciel à cette carte.

La procédure à suivre pour accéder à une carte PCI est la suivante :

- test de la présence d'un BIOS PCI
- recherche de la carte
- lecture des zones mémoires attribuées à la carte

- accès à la carte

La configuration ne peut être faite par l'utilisateur car c'est le BIOS qui s'en charge au démarrage de l'ordinateur.

L'organigramme présenté ci-dessous indique les principales étapes ainsi que les fonctions du programme PCI.EXE.

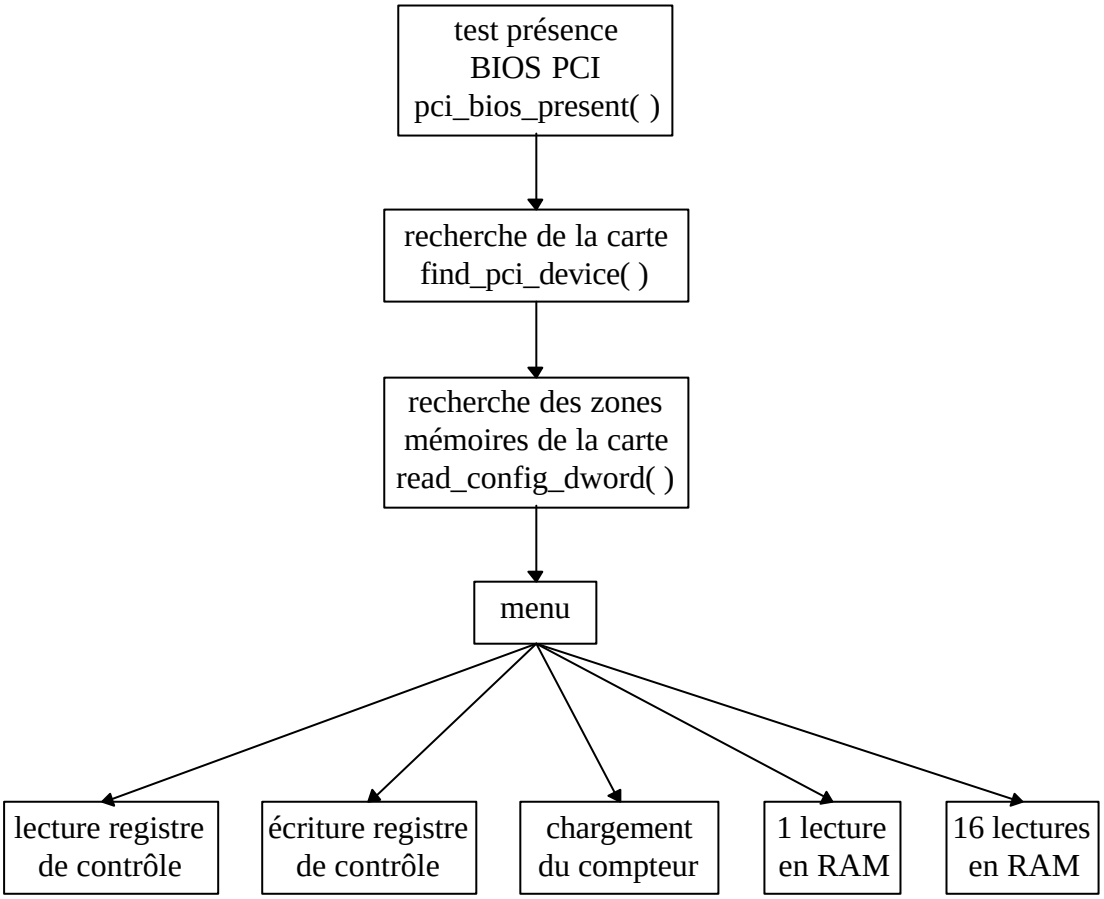


Figure VI-1: organigramme pour l'accès à la carte PCI

Le fichier source PCI.C qui suit utilise les fonctions du BIOS suivantes: *Test for PCI BIOS present*, *Find PCI device* et *Read Configuration dword*.

PCI.C

```
#include "amcc.h" // DEFINITION DE CONSTANTES

/* TEST DE LA PRESENCE DU BIOS PCI */
int pci_bios_present()
{
    byte reg_ah,reg_bh,reg_bl;
    dword reg_edx;
    _asm
    {
        mov AH,0b1h             //
        mov AL,01h              // CHOIX DE LA FONCTION TEST_FOR_PCI_BIOS_PRESENT
        int 1Ah             // APPEL DE LA FONCTION
/* RECUPERATION DES PARAMETRES DE RETOUR */
        mov reg_edx,EDX         // SIGNATURE DU BIOS PCI
        mov reg_bh,BH   // VERSION DU BIOS (octet haut)
        mov reg_bl,BL           // VERSION DU BIOS (octet bas)
        mov reg_ah,AH  //  RESULTAT DU TEST
}
    printf("\n** INFORMATIONS CONCERNANT LE BUS PCI **\n");
    if ((reg_ah==SUCCESSFUL)&&(reg_edx==0x20494350))
    {
        printf("Version du BIOS: %X.%X\n",reg_bh,reg_bl);
        return(SUCCESSFUL); // LA FONCTION RETOURNE SUCCESSFUL SI LE BIOS EST TROUVE
    }
    else
    {
        printf("Pas de BIOS PCI!\n");
        return(NOT_SUCCESSFUL); // SINON ELLE RETOURNE NOT_SUCCESSFUL
    }
}

/* RECHERCHE LA CARTE */
int find_pci_device(word vendor_id, word device_id, byte *bus_num, byte *dev_fct)
{
    byte reg_ah,reg_bl,reg_bh;
    _asm
    {
        mov AH,0b1h
        mov AL,02h              // CHOIX DE LA FONCTION FIND_PCI_DEVICE
        mov CX,device_id        //
        mov DX,vendor_id        // VALEURS D'ENTREES: IDENTIFICATIONS DE LA CARTE
        mov SI,0                //
        int 1Ah
        mov reg_bl,BL           // VALEURS DE SORTIE NECESSAIRE POUR L'ACCES
        mov reg_bh,BH   // EN ZONE DE CONFIGURATION
        mov reg_ah,AH  // RESULTAT DE LA RECHERCHE
    }
    printf("\n** INFORMATIONS SUR LA CARTE PCI **\n");
    if (reg_ah==SUCCESSFUL)
    {
        *bus_num=reg_bh;
        *dev_fct=reg_bl;
        return(SUCCESSFUL);
    }
```

```
    else
    {
        printf("La carte VID=%X DID=%X est absente !\n",vendor_id,device_id);
        return(NOT_SUCCESSFUL);
    }
}

/* LECTURE D'UN MOT LONG EN ZONE DE CONFIGURATION */
int read_configuration_dword(byte bus_num, byte dev_fct, word reg_num, dword *data)
{
        byte reg_ah;
        dword reg_ecx;
        _asm
        {
                mov AH,0b1h             //
                mov AL,0ah              // CHOIX DE LA FONCTION READ_CONFIG_DWORD
                mov BH,bus_num          //
                mov BL,dev_fct          // PARAMETRES IDENTIFIANT LA CARTE
                mov DI,reg_num          // CHOIX DU REGISTRE
                int 1Ah
                mov reg_ecx,ECX         // VALEUR LUE DANS LE REGISTRE
                mov reg_ah,AH           // RESULTAT DE L'ACCES (réussi ou non)
        }
        if (reg_ah==SUCCESSFUL)
        {
                *data=reg_ecx;
                return(SUCCESSFUL);
        }
        else
        {
                printf("Mauvais numéro de registre !\n");
                return(NOT_SUCCESSFUL);
        }
}

main()
{
  int fin=0;
  int i;
  dword donnee;

/* VARIABLES NECESSAIRES POUR L'ACCES EN ZONE DE CONFIGURATION
( OBTENUES AVEC LA FONCTION FIND_PCI_DEVICE)  /*
  byte device_and_function, bus_number;

/* VARIABLES CONTENANT LES ADRESSES DE BASE DE LA CARTE */
  dword adresse_amcc,adresse_registre,adresse_compteur,adresse_ram;

  if (pci_bios_present()==SUCCESSFUL)
  {
    if(find_pci_device(VENDOR_ID,DEVICE_ID,&bus_number,&device_and_function)==SUCCESSFUL)
    {
/* LECTURE DES ADRESSES DE BASES DE LA CARTE */
      read_configuration_dword(bus_number,device_and_function,PCI_CS_BADR0,&adresse_amcc);
      read_configuration_dword(bus_number,device_and_function,PCI_CS_BADR1,&adresse_registre);
      read_configuration_dword(bus_number,device_and_function,PCI_CS_BADR2,&adresse_compteur);
      read_configuration_dword(bus_number,device_and_function,PCI_CS_BADR4,&adresse_ram);
```

```
/* MISE A ZERO DES DEUX BITS DE POIDS FAIBLE DE L'ADRESSE QUI INDIQUENT QUE
L'ADRESSE SE SITUE EN I/O */
      adresse_amcc&=0xFFFC;
      adresse_registre&=0xFFFC;
      adresse_compteur&=0xFFFC;
      adresse_ram&=0xFFFC;

/* AFFICHAGE DES ADRESSES */
      printf("Adresse des registres de l'interface AMCC = %X\n",adresse_amcc);
      printf("Adresse du registre de contrôle = %X\n",adresse_registre);
      printf("Adresse du compteur = %X\n",adresse_compteur);
      printf("Adresse de la RAM = %X\n",adresse_ram);

      while(!fin)
      {
/* AFFICHAGE DU MENU */
            printf("\n1- Lecture du registre de contrôle\n2- Ecriture du registre de contrôle\n");
            printf("3- Chargement du compteur \n");
            printf("4- Lecture simple en RAM\n5- Lectures multiples en RAM\nq- Quitter\n\nVotre choix: ");
            switch(getchar())
            {
/* LECTURE DU REGISTRE DE CONTROLE */
                  case '1': printf("\nRegistre de contrôle = %X\n",_inpd(adresse_registre));
                  break;

/* ECRITURE DU REGISTRE DE CONTROLE */
                  case '2': printf("\nValeur du registre ? ");
                  scanf("%x",&donnee);
                  _outpd(adresse_registre,donnee);
                  break;

/* CHARGEMENT DU COMPTEUR */
                  case '3': printf("\nTaille de l'acquisition en octet (hexa)? ");
                  scanf("%x",&donnee);
                  donnee=0x100000-donnee; // CALCUL DE LA VALEUR A CHARGER
                  printf("Chargement du compteur avec la valeur %X\n",donnee);
                  _outpd(adresse_compteur,donnee); // ECRITURE
                  break;

/* LECTURE D'UNE DONNEE EN RAM */
                  case '4': printf("\nDonnée RAM = %X\n",_inpd(adresse_ram));
                  break;

/* LECTURE DE 16 DONNEES EN RAM */
                  case '5':
                  for(i=0;i<=15;i++) printf("\nDonnée RAM = %X",_inpd(adresse_ram));
                  printf("\n");
                  break;

/* SORTIE DU MENU */
                  case 'q':
                  case 'Q': fin=1;
                  break;

                  default:
                  break;
```

```
            }
            getchar();
        }
      }
    }
}
```

Parallèlement à ce programme, deux autres utilitaires permettent l'accès à la carte PCI:

- DEBUG.EXE sous DOS réalise des accès directs en entrées/sorties et en mémoire. Pour pouvoir accéder à une carte PCI, il faut au préalable connaître ses adresses de base (avec AMCCDIAG.EXE).
- "Propriétés Système" du Panneau de Configuration sous Windows 95, nous informe sous les zones occupées par chaque carte PCI. Il permet également de reconfigurer manuellement les cartes, ce qu'aucun autre programme ne propose.

VI.2 PowerMAC

La méthodologie d'accès PCI sur PowerMAC est complètement différente de celle utilisée par les PC. Dans un PC, les fonctions PCI du BIOS sont facilement accessibles et conviviales alors que dans un PowerMAC, ces fonctions demande une connaissance approfondie du système.

Toutes les fonctions concernant le bus PCI sont à obtenir auprès d'APPLE qui fournit tous les includes et les bibliothèques nécessaires (programmation C).

Le nombre de fonctions et de variables à prendre en compte pour l'accès à une carte PCI est si grand qu'il est très difficile de les utiliser.

Les fonctions que nous emploierons sont principalement des fonctions d'accès en zone de configuration et en entrées/sorties, définies dans **pci.h**:

Nom	Description
ExpMgrConfigReadByte	Lecture d'un octet en zone de configuration
ExpMgrConfigReadWord	Lecture d'un mot en zone de configuration
ExpMgrConfigReadLong	Lecture d'un mot long en zone de configuration
ExpMgrConfigWriteByte	Ecriture d'un octet en zone de configuration
ExpMgrConfigWriteWord	Ecriture d'un mot en zone de configuration
ExpMgrConfigWriteLong	Ecriture d'un mot long en zone de configuration
ExpMgrIOReadByte	Lecture d'un octet en entrée/sortie
ExpMgrIOReadWord	Lecture d'un mot en entrée/sortie
ExpMgrIOReadLong	Lecture d'un mot long en entrée/sortie
ExpMgrIOWriteByte	Ecriture d'un octet en entrée/sortie
ExpMgrIOWriteWord	Ecriture d'un mot en entrée/sortie
ExpMgrIOWriteLong	Ecriture d'un mot long en entrée/sortie

Tableau VI-2: fonctions PCI du PowerMAC

La recherche d'une carte et de ses adresses passe par le **device tree**. Il s'agit d'une liste d'informations concernant les périphériques du PowerMAC et notamment les cartes PCI. Cet arbre se situe en RAM.

Deux programmes peuvent nous aider dans la recherche de carte:

- *PciSlots* qui affiche toutes les cartes PCI reconnues. Il indique leur identificateur et leur numéro de connecteur.

- *DisplayNameRegistry* qui présente le détail de toutes les informations disponibles pour une carte (voir le fichier texte en annexe):

  - idendification (pci1234,5678 pour une carte PCI avec Vendor ID = 0x1234 et Device ID = 0x5678).
  - les adresse de bases (physiques et logiques).
  - le numéro du slot utilisé.
  - et tous les autres champs de la zone de configuration.

Ces deux programmes nous ont permis de vérifier la compatibilité du bus PCI des PC et des PowerMAC puisque la carte a été parfaitement reconnue.

VII. CONCLUSION

Le but du stage était l'étude et la réalisation d'un nouvelle version de carte d'acquisition pour le test de convertisseur Analogique/Numérique. La plupart des objectifs ont été atteints puisque les étapes suivantes ont été effectuées:

- choix de l'interface
- conception de la carte
- réalisation d'un prototype wrappé
- test du prototype
- interface logicielle
- routage de la carte

Le projet s'est révélé particulièrement intéressant et instructif tant au niveau personnel que professionnel mais il nous a semblé un peu long notamment en ce qui concerne le routage.

La suite du projet va consister à fabriquer et à monter la carte puis à tester son fonctionnement.

BIBLIOGRAPHIE

- PCI SYSTEM ARCHITECTURE
  MindShare, INC. Tom Shanley / Don Anderson
  1995

- PCI HARDWARE AND SOFTWARE
  Annabooks Edward Salari & George Willse
  1995

- S5930-S5933 PCI Controllers
  AMCC
  1995

- Documentations ALTERA

- La Bible du PC
  Micro Application Michael Tischer
  1992

ANNEXE

Schéma de la carte

Routage

Altera

Logiciels