ENSEIGNEMENT DE PROMOTION SOCIALE

Cours de

STRUCTURE DES ORDINATEURS

- Le bus et le chipset -

Version provisoire

H. Schyns

Fevrier 2013

Sommaire

1. VUE GENERALE

1.1. Analogie

1.2. Principe de fonctionnement

1.3. Transfert synchrone ou asynchrone

1.4. Caractéristiques d'un bus

2. LES PREMIERES ARCHITECTURES

2.1. Au commencement était le bus…

2.2. La libération du CPU

2.3. Diviser pour régner

2.4. Le contrôleur ou pont

2.4.1. Rôle du contrôleur

2.4.2. Transfert Lent **ð** Rapide

2.4.3. Transfert Rapide **ð** Lent

3. LE BUS ISA

3.1. A l'origine

3.2. Avantages et inconvénients

4. LE BUS SYSTEME

4.1. Au commencement…

4.2. Frontside, Backside

4.3. Le contrôleur de mémoire

4.4. Multiples et diviseurs

4.5. Autres problèmes

4.6. La diversification des bus I/O

5. L'APRES ISA : UNE SUCCESSION DIFFICILE

5.1. Un peu d'histoire

5.2. MCA

5.3. EISA

5.4. VLB

6. LE BUS PCI : ZORRO EST ARRIVE

6.1. Description

6.2. Plug and Play

6.3. IRQ internes

6.4. Maîtrise du bus

7. LE BUS AGP

8. LES BUS "SERIE"

9. SOURCES

9.1. Ouvrages

1. Vue générale

1.1. Analogie

Un ordinateur est un ensemble de composants interconnectés qui fonctionnent à la manière d'une ville ou d'une région. Dans ce chapitre, nous nous focaliserons plus sur la **nature des connexions** que sur les composants eux-mêmes.

Partons d'une comparaison. Dans la vie quotidienne d'une région, certains habitants (ou certaines zones) ont des besoins qui peuvent êtres satisfaits par d'autres habitants (ou d'autres zones) qui, eux, disposent des ressources nécessaires. Les uns sont les consommateurs, les autres sont les producteurs.

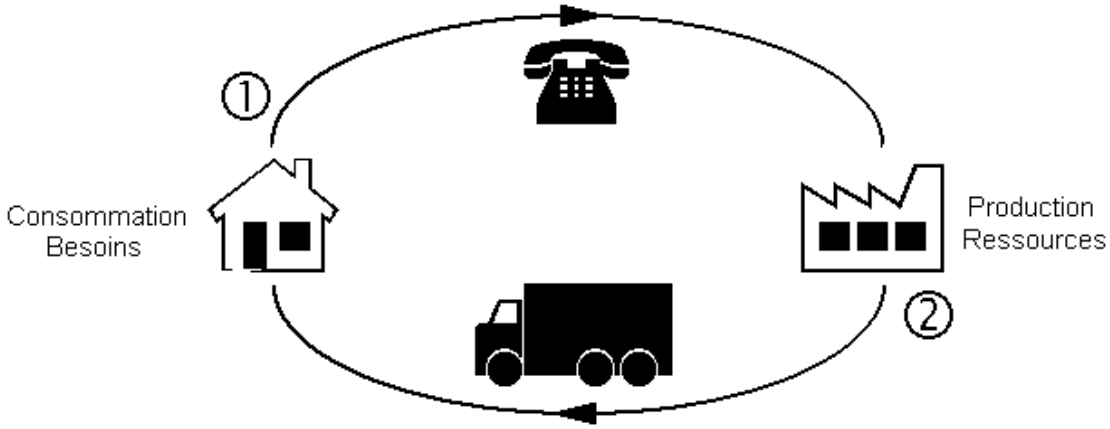

Figure 1.1 Communications entre consommateur et producteur

Pour transmettre le besoin du consommateur au producteur, le système a besoin d'un canal de **communication** tel qu'une ligne téléphonique. Pour transmettre la ressource du producteur au consommateur, le système emprunte un autre canal de **transport** tel qu'une route.

Quand le nombre de consommateurs et de producteurs augmente les systèmes de communication et de transport deviennent de plus en plus complexes et diversifiés. Il devient indispensable d'installer des systèmes de **contrôle** : centraux téléphoniques, centres de tri du courrier, feux de circulation, passages à niveau, etc.

D'un autre côté, toutes les branches du réseau de transport ne fonctionnent pas à la même vitesse et ne permettent pas d'assurer les mêmes flux de véhicules : un transporteur ne circule pas dans une zone rurale comme sur une autoroute. Il appartiendra au système de contrôle de régulariser et de synchroniser la circulation entre les différentes branches. De même, il sera parfois nécessaire de prévoir des zones de transbordement et de stockage temporaire tels que gares ferroviaires, gares routières, aéroports et ports.

Dans l'ordinateur, les choses se passent de manière similaire.

Les différents composants tels que CPU, mémoire RAM, clavier, écran, etc. doivent se transmettre des **demandes** et échanger des **informations**. Ils sont donc reliés par un ensemble de liaisons physiques tels que câbles, pistes de circuits imprimés qu'ils peuvent exploiter en commun afin de communiquer. Cet ensemble de connexions est le **bus** (*ang.: bus*). On peut voir le bus comme un mini-réseau (*ang.: network*) installé au cœur même de la machine.

Plusieurs contrôleurs sont chargés de réguler la circulation de l'information sur le bus. Ces contrôleurs sont programmés "en dur" dans un ensemble de circuits intégrés nommés **ponts** (*ang.: bridges*), répartis que la carte mère. Ils forment le **chipset**.

1.2. Principe de fonctionnement

Physiquement, un bus n'est rien d'autre qu'une nappe de câbles de connexion ou une série d'une dizaine de traces parallèles sur la carte mère.

Ces lignes sont mises à la disposition des différents éléments de l'ordinateur afin qu'ils puissent se transmettre des informations. Dans le cas d'une ligne "privée", dédicacée à la communication entre deux composants spécifiques, on parle plutôt de port (port série, port parallèle, port USD).

En quelque sorte, le bus est une autoroute sur laquelle les informations voyagent d'un point à l'autre de l'ordinateur.

Le bus se compose de trois parties ([1]) :

- le bus d'adresses (*ang.: address bus*), qui véhicule l'adresse du composant auquel les demandes ou les données sont destinées;
- le bus de données (*ang.: data bus*), qui véhicule l'information proprement dite;
- le bus de contrôle (*ang.: control bus*), qui sert au transport des signaux de synchronisation et de validation (clock, request, acknowledge,…)

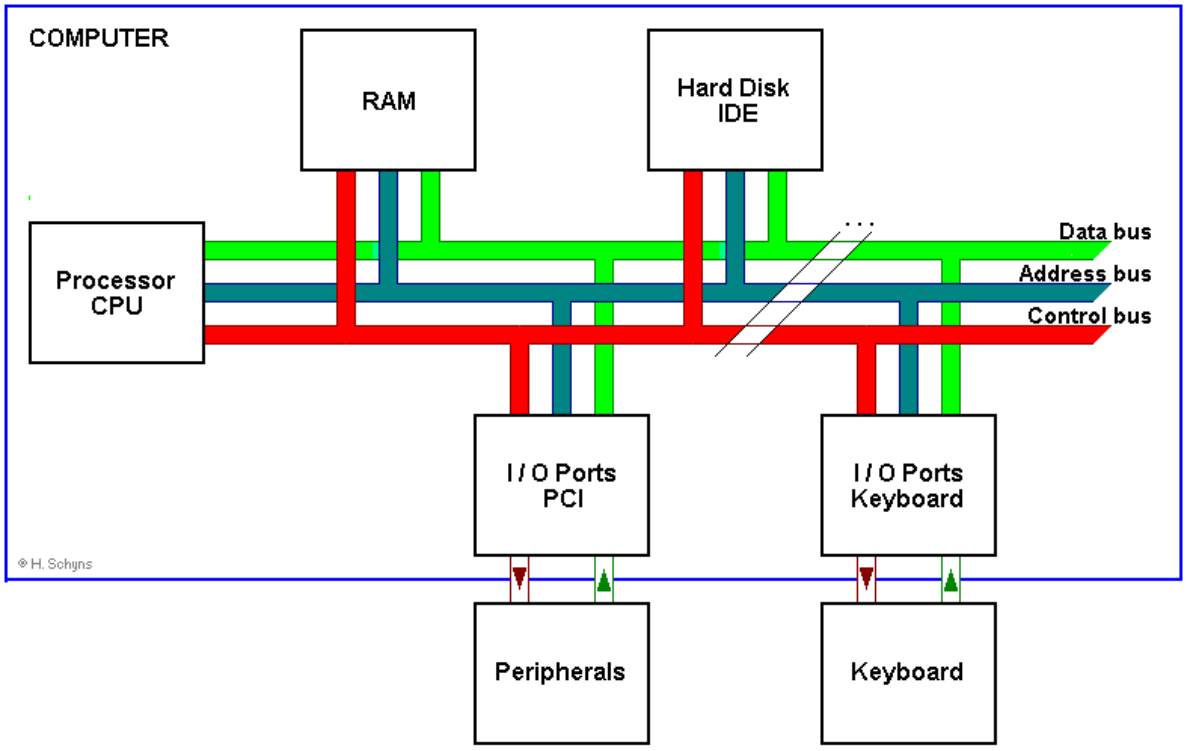

Figure 1.2 Principe de base du bus

Dans un ordinateur, chaque composant possède une adresse. Quand un composant veut demander une information, il utilise le **bus d'adresses** pour envoyer l'identification du composant auquel la demande est destinée. Si le destinataire est capable de fournir plusieurs services (lecture, écriture, remise à zéro,…), l'expéditeur envoie aussi un signal sur le **bus de contrôle** afin de préciser sa demande.

Quand le destinataire est prêt, il renvoie l'information sur le **bus de données**. S'il y a un risque de confusion, il utilise aussi le bus d'adresse pour renvoyer l'adresse du composant auquel les données sont destinées. De même, il utilise le bus de contrôle pour signaler que la demande a été exécutée.

1 Certains constructeurs ordinateurs tels DEC (Digital Equipment Corp) et PDP utilisent la technologie Unibus : les trois fonctions d'adresse, de données et de contrôle sont véhiculées par un seul bus. A bien y réfléchir, cette séparation des lignes en trois tâches est assez arbitraire.

1.3. Transfert synchrone ou asynchrone

(Image de la chaîne pour éteindre un incendie)

(à développer…)

1.4. Caractéristiques d'un bus

Un bus est caractérisé par

- sa largeur
- sa fréquence

La **largeur du bus** est le nombre de bits qu'il peut transmettre simultanément. Elle correspond au nombre de lignes sur lesquelles les informations peuvent être envoyées en parallèle.

A titre d'exemple, imaginons un circuit imprimé formé de 16 traces parallèles structurées en 8 lignes destinées aux données, 6 lignes destinées aux adresses et 2 lignes réservées au contrôle ([1]).

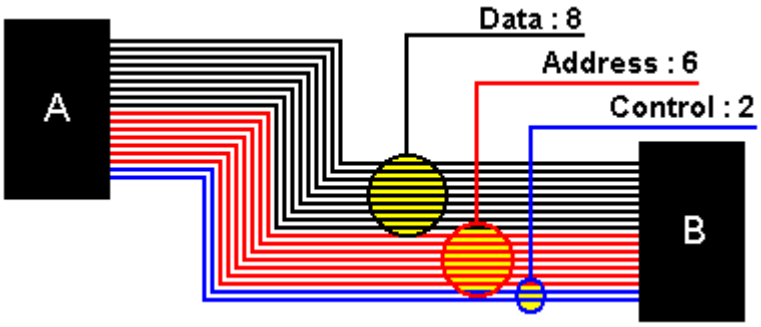

Figure 1.3 Exemple de bus illustrant la largeur des différentes composantes

La largeur *totale* du bus est de 16 bits tandis que la largeur du bus de *données* est de 8 bits.

Dans le langage courant, quand on parle de largeur de bus, on parle généralement de la largeur du **bus de données**.

Plus le **bus d'adresse** est large, plus le nombre d'adresses accessibles augmente. Dans le cas de notre exemple imaginaire, le bus d'adresse a une largeur de 6 bits, ce qui lui permet d'atteindre 2^6, soit 64 adresses. Ce serait insuffisant pour servir une barrette de mémoire RAM mais suffisant pour adresser un clavier, une souris et quelques autres périphériques peu sophistiqués.

Les premiers processeurs possédaient un bus d'adresse de 16 bits qui leur permettait d'accéder à 64kB de mémoire. Quand les constructeurs ont voulu ajouter plus de mémoire à leurs ordinateurs, ils ont obligatoirement dû augmenter la largeur du bus d'adresses ([2]). Depuis la génération 386, les processeurs travaillent avec 32 bits d'adresse et peuvent ainsi utiliser 4 GB de mémoire.

1 En pratique, sur une carte mère, il est impossible de distinguer les différentes composantes du bus. Les connexions sont d'ailleurs souvent disposées en plusieurs couches réparties dans l'épaisseur de la carte mère. Il est donc indispensable de consulter la notice technique.

2 Il est possible de tricher et d'utiliser les lignes du bus de contrôle pour élargir le champ du bus d'adresse. C'est ce qui est fait au niveau de l'accès à la RAM par l'envoi de messages RAS et CAS

Plus le **bus de données** est large, plus le volume d'information envoyé en une fois augmente. On sait par exemple qu'un nombre réel est codé sur 32 bits. Il faudra quatre cycles d'horloge pour envoyer un tel nombre sur le bus de données de notre exemple (8 bits de large). Il n'en faudrait qu'un si le bus de données avait une largeur de 32 bits. Le processeur, dont les performances sont presque toujours limitées par les vitesses de transfert de l'information dans le PC, semblera travailler quatre fois plus vite.

Plus le **bus de contrôle** est large, plus la variété des ordres transmis entre les éléments de l'ordinateur augmente et plus le contrôle est souple et fin. Le bus de notre exemple dispose d'un état de repos (p.ex.: 00) et de trois ordres possibles (p.ex.: 01=read, 10= write, 11=erase).

La **fréquence du bus** est le nombre de paquets de données envoyés ou reçus par seconde. Elle s'exprime en hertz (Hz) et représente la cadence à laquelle les paquets de bits sont transmis. A l'inverse, la durée de cycle est le temps nécessaire à l'envoi d'*un* paquet de données. La fréquence est identique pour toutes les composantes du bus.

La **bande passante** du bus est le débit de données qu'il peut transporter. Elle s'obtient en multipliant la fréquence par la largeur.

$$\text{Bande passante} = \text{Largeur} \cdot \text{Fréquence}$$

Plus la bande passante est large, plus la vitesse apparente d'exécution des applications est élevée et plus la puissance de traitement de l'ordinateur augmente.

A titre d'exemple, la bande passante d'un bus de données d'une largeur de 16 bits, cadencé à une fréquence de 133 MHz est :

$16 \cdot 133 \cdot 10^6 = 2128 \cdot 10^6$ bits/s

$2128 \cdot 10^6 / 8 = 266 \cdot 10^6$ Bytes/s

$266 \cdot 10^6 / 1024 = 257.7 \cdot 10^3$ kBytes/s

$257.7 \cdot 10^3 / 1024 = 253.7$ MB/s

Notez au passage la division par 1024 et non par 1000 comme c'est souvent le cas en informatique ([1]).

1 Pour éviter les confusions entre les facteurs 1000 et 1024, il faudrait écrire
253,7 MB/s = 266 B/µs ou 2 GB/s = 2,15 B/ns

2. Les premières architectures

2.1. Au commencement était le bus…

En 1980, les premiers PC se composaient

- d'un processeur fonctionnant à 4.77 ou 6 MHz (mode turbo),
- d'une mémoire RAM de 64 ou 640 kB (XT),
- d'un lecteur de disquette 1.44 MB,
- de connecteurs d'extension
- d'un clavier,
- d'un écran monochrome 25 lignes x 80 colonnes (mode texte).

Ces éléments se partageaient un seul bus d'une largeur de 8 bits de données, 20 bits d'adresses et une douzaine de lignes de contrôle.

Ce **bus système** unique (*ang.: system bus*) fonctionnait en parfait synchronisme à la fréquence du CPU (4,77 ou 6 MHz), ce qui assurait une bande passante de 4,7 à 6 MBps. Cette vitesse était en quelque sorte le dénominateur commun auquel tous les composants pouvaient s'accorder (Figure 2.1).

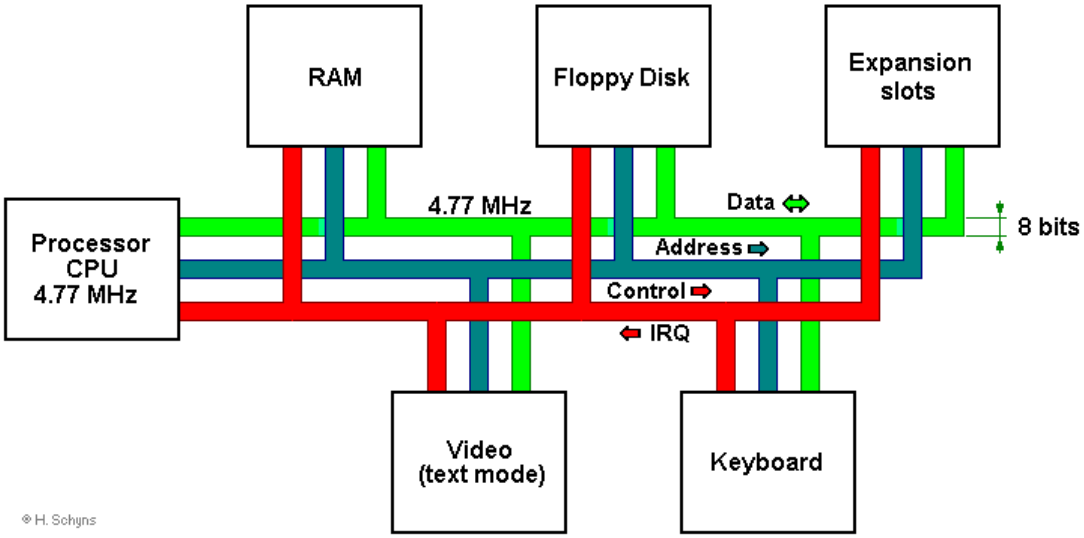

Figure 2.1 Bus système dans un XT

C'est aussi le CPU qui assurait l'essentiel du contrôle du bus ainsi que l'échange de données entre les composants. Le bus d'adresses était unidirectionnel mais comme il fallait bien que les données reviennent vers le processeur, le bus de données, lui, était bidirectionnel.

A cette époque où les chips de RAM s'enfichaient directement dans la carte mère, il était possible d'ajouter de la mémoire supplémentaire (très chère) en plaçant une carte d'extension spéciale dans un des slots disponibles.

2.2. La libération du CPU

Dans l'architecture de ces premiers PC, le transfert d'une donnée d'un périphérique vers la RAM se faisait en trois étapes :

- le CPU s'adresse au périphérique pour lui demander une donnée,
- le périphérique renvoie la donnée sur le bus de données,
- le CPU s'adresse à la RAM pour lui demander de stocker la donnée.

Le processus inverse avait lieu quand il s'agissait d'envoyer une donnée à un périphérique :

- le CPU s'adresse à la RAM pour lui demander une donnée,
- la RAM renvoie la donnée sur le bus de données,
- le CPU s'adresse au périphérique pour lui demander de stocker ou d'afficher la donnée.

On comprend que, dans ce système, le CPU soit fort sollicité par les tâches annexes, notamment par l'affichage vidéo. Dans ces conditions, il était indécent de lui demander en plus de faire le tour des périphériques pour savoir s'ils ont besoin de ses services.

La première idée était donc d'autoriser les périphériques à solliciter l'attention du CPU quand ils en avaient besoin (et uniquement dans ce cas) en émettant des **requêtes d'interruption** (*ang.: interrupt request* ou *IRQ*) sur le bus de contrôle. C'est, par exemple, le cas du clavier qui vient de recevoir la frappe d'un caractère et qui désire l'envoyer au CPU car lui-même ne sait pas ce qu'il faut en faire. Ces célèbres **IRQ** ont donné des cauchemars à plus d'un informaticien; nous aurons l'occasion d'y revenir dans un prochain chapitre.

La deuxième idée était de décharger le processeur des tâches de contrôle en les confiant à une puce externe nommée **contrôleur** (*ang.: controller*).

La synthèse des deux idées était d'intégrer la gestion du bus et des IRQ dans la même puce : la notion de **chipset** était née; le CPU s'était doté d'une secrétaire ! Nous verrons ci-après les différentes étapes de ce processus.

2.3. Diviser pour régner

L'évolution technique se poursuivant, les processeurs et les mémoires ont vu leur fréquence de travail augmenter rapidement. Dans le même temps, leur capacité de traitement est passée de 8 à 16 bits. Par contre, du côté des périphériques, l'évolution n'était pas aussi rapide.

Pour éviter de pénaliser le processeur, les ingénieurs de Compaq ont imaginé d'isoler le dialogue entre le CPU et la RAM du reste des périphériques. Les concepteurs ont donc scindé le bus en deux parties :

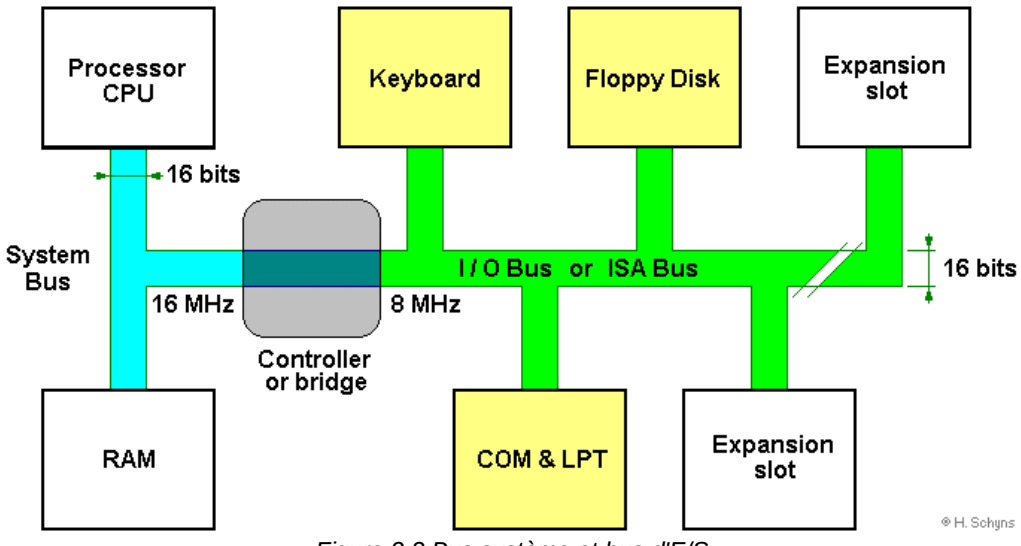

Figure 2.2 Bus système et bus d'E/S

- un **bus système** (*ang.: system bus*) ou **bus local** (*ang.: local bus*) qui connecte le CPU et la mémoire RAM et qui peut fonctionner à haute vitesse (p.ex.: 16 MHz à cette époque)

- un **bus d'entrées/sorties** (*ang.: I/O bus*) qui relie les périphériques plus lents (p.ex.: 8 MHz). Le bus d'entrée/sortie permet aux divers composants de la carte-mère de communiquer entre eux mais il permet surtout l'ajout de nouveaux composants périphériques grâce aux connecteurs d'extension (*ang.: expansion slots*) qui y sont connectés.
- un contrôleur ou **pont** (*ang.: bridge*) qui assure le transfert et la synchronisation des informations entre les deux bus. Il deviendra le chipset.

2.4. Le contrôleur ou pont

2.4.1. Rôle du contrôleur

Le rôle du contrôleur est d'assurer la compatibilité entre deux éléments qui ne fonctionnent pas à la même fréquence et/ou qui n'ont pas la même largeur de bus. On peut le comparer à un entonnoir.

La tâche du contrôleur sera plus aisée si les largeurs et les fréquences des bus situés de part et d'autre sont dans un rapport entier ou demi-entier (2/1, 4/1, 3/2 plutôt que 17/5 ou 97/13).

A titre d'illustration, imaginons un système composé de :

- un bus lent et étroit qui fonctionne à 8 MHz, avec une largeur de 8 bits,
- un bus rapide et large qui fonctionne à 16 MHz, avec une largeur de 16 bits.
- un contrôleur qui assure le transfert entre ces deux bus.

2.4.2. Transfert Lent ð Rapide

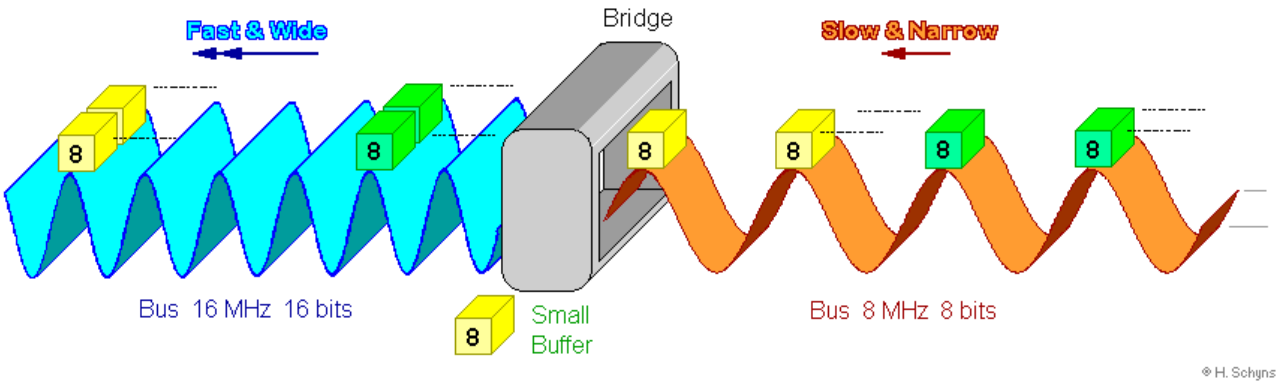

Figure 2.3 Transfert du bus lent et étroit vers le bus rapide et large

Sur la Figure 2.3, l'information provient du bus lent et étroit (à droite). Elle doit être transférée sur le bus rapide et large (à gauche) afin d'atteindre le processeur.

Le pont reçoit un flux continu de paquets de 8 bits venant de divers périphériques. Chaque cycle d'horloge du bus lent lui apporte un nouveau paquet. Le contrôleur pourrait transférer instantanément chaque paquet de l'autre côté mais le bus rapide ne véhicule que des paquets de 16 bits. Le pont est donc obligé de stocker temporairement le byte dans une petite mémoire tampon (*ang.: small buffer*). Dès que le paquet suivant arrive, le pont l'assemble en mémoire avec le paquet qui s'y trouvait déjà. Le tout sera déposé sur le bus rapide lors de son prochain cycle (Notez qu'il n'est pas nécessaire que les deux bus soient synchrones). Le bus rapide ayant une bande passante quatre fois plus grande que celle du bus lent, il laisse passer trois cycles d'attente entre deux cycles utiles.

Pour éviter les problèmes, le pont ne regroupe que des paquets en provenance du même périphérique. Cette situation risque de poser des problèmes lorsque le

nombre de paquets à transmettre est impair. Une solution consiste à fabriquer de toutes pièces un paquet supplémentaire, rempli de zéros, que l'on colle au dernier paquet afin de pouvoir l'expédier sur le bus rapide. Bien entendu, à l'autre bout de la chaîne, le processeur doit être au courant de cette pratique. En fait, la manière de terminer les trains incomplets est définie par les spécifications du processeur et ce sont les ponts qui s'adaptent.

La situation peut être comparée une station de métro : les voyageurs y arrivent par un escalier mécanique relativement lent et s'engouffrent dans une rame qui les conduit rapidement à destination. Le quai fait office de mémoire tampon ([1])([2]).

2.4.3. Transfert Rapide ð Lent

La situation inverse est beaucoup plus délicate à gérer. On peut la comparer à l'arrivée d'une rame de métro bondée, dans une station que tous les voyageurs doivent quitter en passant par un escalier mécanique relativement lent.

Sur la Figure 2.4, l'information provient du processeur par le bus rapide et large (à gauche). Elle doit être transférée vers le périphérique par le bus lent et étroit (à droite).

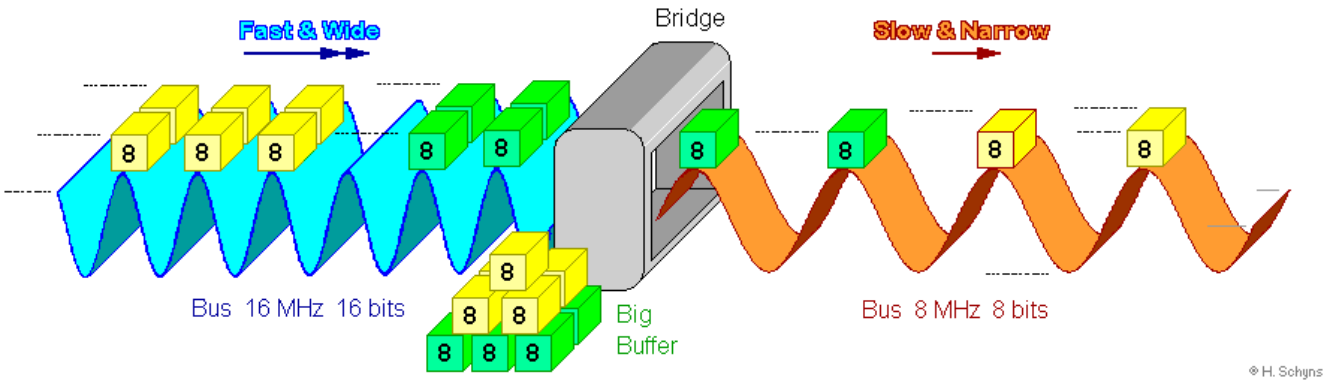

Figure 2.4 Transfert du bus rapide et large vers le bus lent et étroit

Cette fois, le pont doit faire face à un afflux considérable de paquets de 16 bits venant du processeur. Chaque cycle d'horloge du bus rapide lui apporte un nouveau lot de 16 bits. Le contrôleur doit scinder chaque lot en deux, stocker une moitié et envoyer l'autre sur le bus lent. La seconde moitié sera expédiée lors du prochain cycle lent.

Le problème, c'est que, dans cet exemple, la fréquence du bus lent est deux fois plus faible que celle du bus rapide. A chaque cycle du côté de la sortie correspondent deux cycles du côté de l'entrée. Pour chaque byte quittant le pont, quatre nouveaux y arrivent.

Pour faire face à cette avalanche, le pont doit disposer d'une grande mémoire de stockage temporaire… en espérant qu'une accalmie de trafic lui permettra de résorber le travail en retard.

Si l'accalmie ne se produit pas spontanément, le contrôleur envoie un message "wait state" sur le bus de contrôle. Il s'agit d'un message d'avertissement destiné au processeur (ou au contrôleur placé en amont) lui demandant de suspendre la communication pendant un ou plusieurs cycle. Bien entendu, de proche en proche, c'est toute l'activité de l'ordinateur qui risque de se voir ralentie.

1 La comparaison suppose qu'il passe plus de rames de métro qu'il n'arrive de visiteurs ce qui n'est certes pas le cas dans la réalité.

2 Finalement, c'est peut-être pour ça qu'on parle de "bus".

3. Le bus ISA

3.1. A l'origine

Le **premier bus d'entrées-sorties** a été standardisé dès sa conception en 1984 et, pour cette raison, baptisé ISA (*ang.: Industry Standard Architecture*)([1]). Le premier bus ISA avait une largeur de **8 bits** et fonctionnait à **8 MHz**. Il a été rapidement amélioré en portant sa largeur à **16 bits** (toujours sous **8 MHz**).

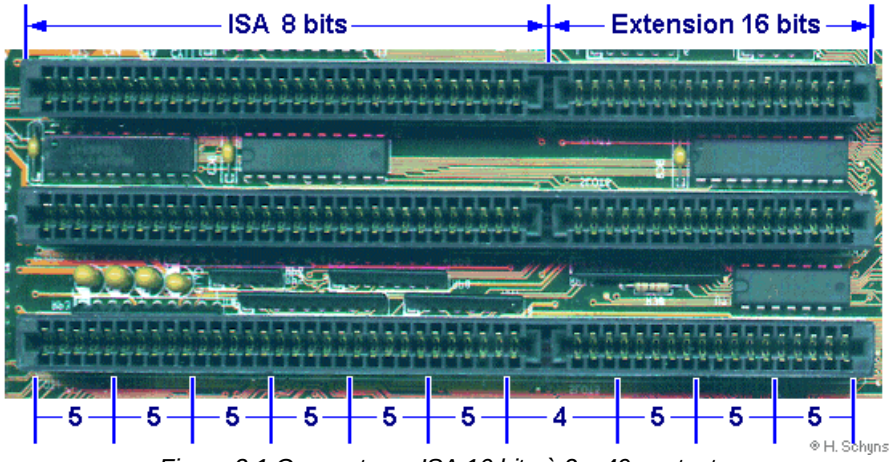

Figure 3.1 Connecteurs ISA 16 bits à 2 x 49 contacts

A l'origine, le bus ISA était conçu pour fonctionner de manière synchrone avec le CPU. Toutefois, quand la fréquence du bus système a dépassé 10 MHz, de nombreuses cartes d'extension ont commencé à battre la chamade. L'horloge du bus ISA a été réduite à une fraction entière de celle du bus système (grâce au pont) de manière à revenir dans la plage de 8 à 10 MHz.

Théoriquement, la bande passante du bus ISA est de **15.25 MB/s**. En pratique, le contrôleur du bus ISA a généralement besoin de 2 ou 3 cycles d'horloge pour déplacer 16 bits de données ce qui réduit fortement son débit.

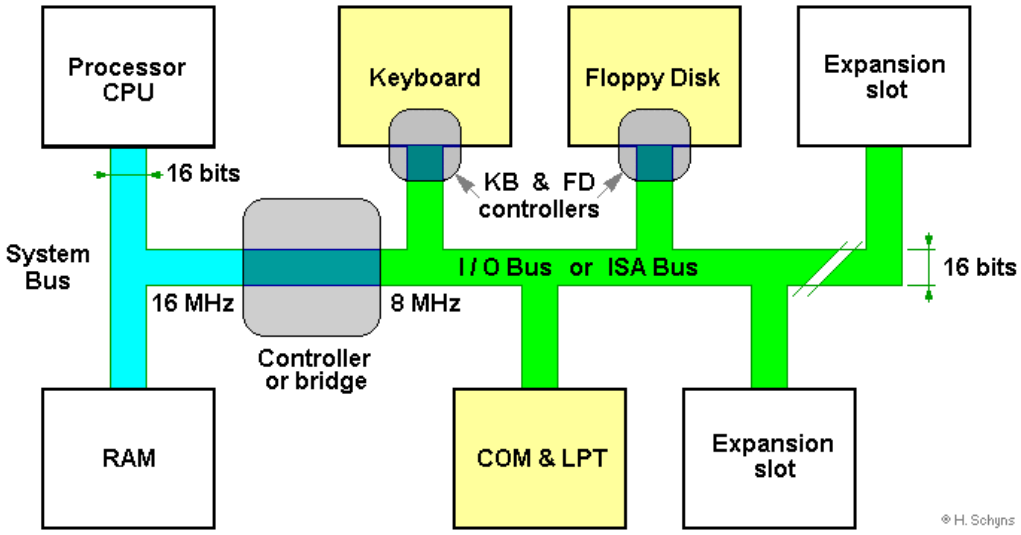

Figure 3.2 Le bus ISA représente une norme de communication
Certains périphériques sont pourvus d'un contrôleur afin de s'y adapter

1 Le célèbre MacIntosh de Apple, contemporain des premiers PC utilisait un autre type de bus nommé NuBus sur lequel nous ne nous étendrons pas. Les modèles plus récents utilisent un bus PCI.

Malgré sa relative lenteur, le bus ISA a lontemps été présent dans de nombreux PC de manière à maintenir la compatibilité avec d'anciennes cartes d'extension. Il faut reconnaître que sa bande passante est très largement suffisante pour satisfaire un clavier, une souris et un modem 56k.

3.2. Avantages et inconvénients

Le bus ISA joue deux rôles :

- un **rôle interne**, qui est la gestion des ports les plus simples tels que le clavier, la souris, le lecteur de disquette, les ports série (COM) et parallèles (LPT)
- un **rôle externe**, qui est la gestion des connecteurs d'extension (*ang.: expansion slots*) sur lesquels se connectent les cartes pourvues d'un adaptateur 16 bits ISA. C'est le cas des cartes modem 56k et des cartes son de type SoundBlaster qui ont connu leur heure de gloire.

Le gros **avantage** du bus ISA est précisément sa standardisation. Il n'est pas indispensable que les périphériques eux-mêmes fonctionnent à la vitesse du bus ISA. Il suffit qu'ils soient pourvus d'un contrôleur qui, lui, travaille en sortie selon le standard et la vitesse du bus ISA. Ce sera notamment le cas du clavier et du lecteur de disquette. Ce principe ouvre la porte à une grande souplesse et une grande diversité au niveau de la conception et de la fabrication des périphériques.

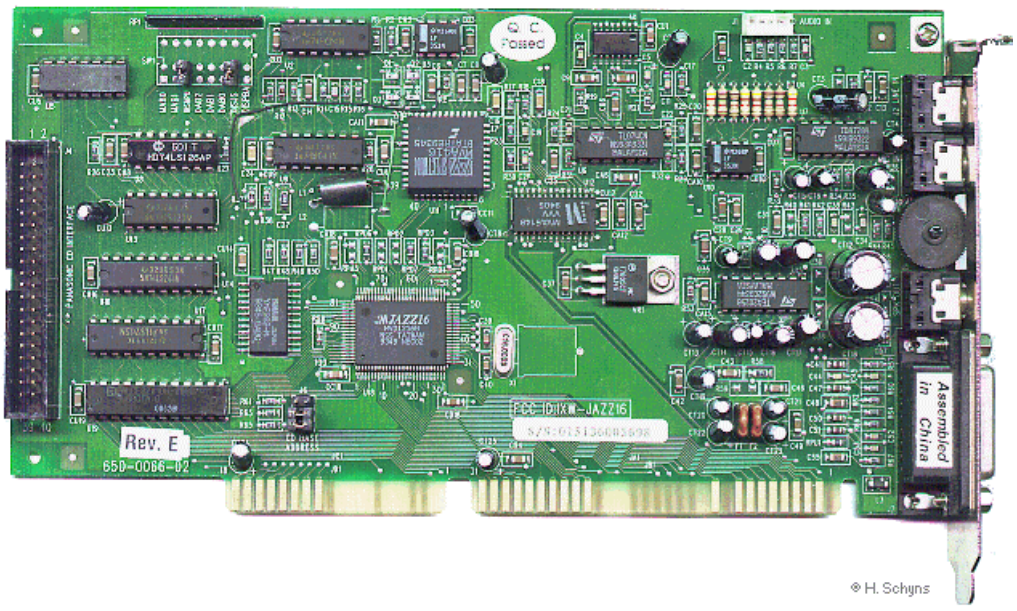

Figure 3.3 Carte son pour connecteur ISA.
Les 2 x 49 contacts ne sont pas tous utilisés. On note la présence d'un port IDE à gauche

Avec l'évolution technique, il est vite apparu que le bus ISA ne pourrait pas se charger de toutes les tâches. Il présente plusieurs **inconvénients** :

- il est lent,
- il n'a aucune intelligence,
- il provoque des états d'attente,
- il gère mal les demandes d'interruption.

La **lenteur** va devenir un problème avec l'amélioration des propriétés graphiques. Un écran monochrome de 480x640 pixels ne nécessite qu'un bit par pixel, soit 38 400 bytes par écran. En 256 couleurs, on passe à 307 200 bytes par écran. Le

débit du bus ISA ne permet guère de rafraîchir l'image plus de 10 fois par seconde ([1]).

Le **manque d'intelligence** du bus ISA impose au CPU de contrôler le transfert sur le bus. Le CPU ne peut pas lancer une nouvelle demande tant que le transfert précédent n'est pas terminé. Sur d'anciennes machines on peut remarquer que, quand le PC communique avec le lecteur de disquette, tout le reste du PC se met en état d'attente.

Les **états d'attente** (*ang.: wait states*) sont des demandes de pauses. Lorsque le bus ISA doit faire face à un flux de données trop important en provenance du bus système, son contrôleur envoie un message "wait state" sur le bus de contrôle. Ce message demande au processeur de suspendre la communication pendant un cycle du bus. Comme un cycle de bus correspond à plusieurs cycles de CPU, la vitesse des applications peut se voir considérablement diminuée.

Les **demandes d'interruption** ou **IRQ** (*ang.: interrupt requests*) sont des signaux que les composants envoient sur le bus de contrôle pour attirer l'attention du CPU. Pour les cartes d'extension installées dans les connecteurs ISA, l'attribution des IRQ doit être faite manuellement par l'utilisateur, ce qui peut devenir un véritable casse-tête. Nous reviendrons sur ce point dans un chapitre consacré aux ressources système.

Selon Intel, le bus ISA fait aujourd'hui partie des "technologies dépassées héritées du passé". Il n'est plus implanté dans les nouvelles cartes mères basées sur le chipset Intel 810. Il est remplacé par le bus USB; un bus série à faible débit dont il sera question plus loin.

1 Les concepteurs de jeu contourneront le problème en utilisant des "sprites". Les personnages tiennent dans de petits carrés de 16x16 pixels qui ne reprennent que la partie animée de l'image. Et dire qu'on revient à ce système dans les jeux pour téléphones mobiles !

4. Le bus système

4.1. Au commencement…

La norme ISA venait à peine de régler le problème des communications entre le CPU et les périphériques, que les choses ont commencé à se compliquer entre le CPU et la RAM.

On appelle **bus système** (*ang.: system bus*) le canal qui connecte le processeur (CPU) à la mémoire vive (RAM) et, dans certains cas, à une mémoire tampon nommée "cache L2".

Le bus système est tracé sur la carte mère. Il est toujours conçu pour s'adapter à un type de processeur bien spécifique. Dans les premiers PC, le bus système était un bus local : sa vitesse était toujours égale à celle du processeur ainsi qu'on peut le voir dans le tableau suivant.

CPU	Bus système	
	Largeur (bits)	Fréquence (MHz)
8088	8	4.77
8086	16	8
80286 -12	16	12
80386 SX-16	16	16
80386 DX-25	32	25

Augmenter la vitesse du processeur imposait d'augmenter la vitesse du bus. Dès lors une bonne part des recherches technologiques a porté sur l'augmentation du trafic sur la carte mère.

4.2. Frontside, Backside

Malheureusement, la fréquence de travail des **barrettes DRAM** est limitée par des temps morts qui sont imposés par les processus de rafraîchissement ([1]). Le **processeur**, qui utilise des mémoires internes et un cache L1 de type SRAM, n'est pas limité par ce problème. Ceci fut une première source de tensions entre les deux partenaires.

L'évolution technologique se poursuivant, la fréquence de travail des processeurs augmentait plus vite que celle de la mémoire RAM. Les performances de l'ordinateur étaient de plus en plus limitées par celles de la mémoire. Ceci ne fit qu'exacerber les tensions.

Pour contourner le problème, les ingénieurs reprirent l'idée du "pont entre deux bus", en l'appliquant cette fois au transfert des données entre le CPU et la RAM.

Pour commencer, on ajouta un cache de niveau 2 (*ang.: Level 2 cache* ou *L2*), de type SRAM, donc très rapide et sans temps morts de rafraîchissement, dans lequel le processeur peut stocker des pages complètes de mémoire.

Ensuite, on remplaça le bus système standard par une structure à deux bus indépendants (*ang.: Dual Independent Bus* ou *DIB*). Cette structure comprenait un bus frontal (*ang.: frontside bus*) et un bus arrière (*ang.: backside bus*) (Figure 4.4).

1 Ce problème est discuté en détail dans le chapitre consacré à la mémoire

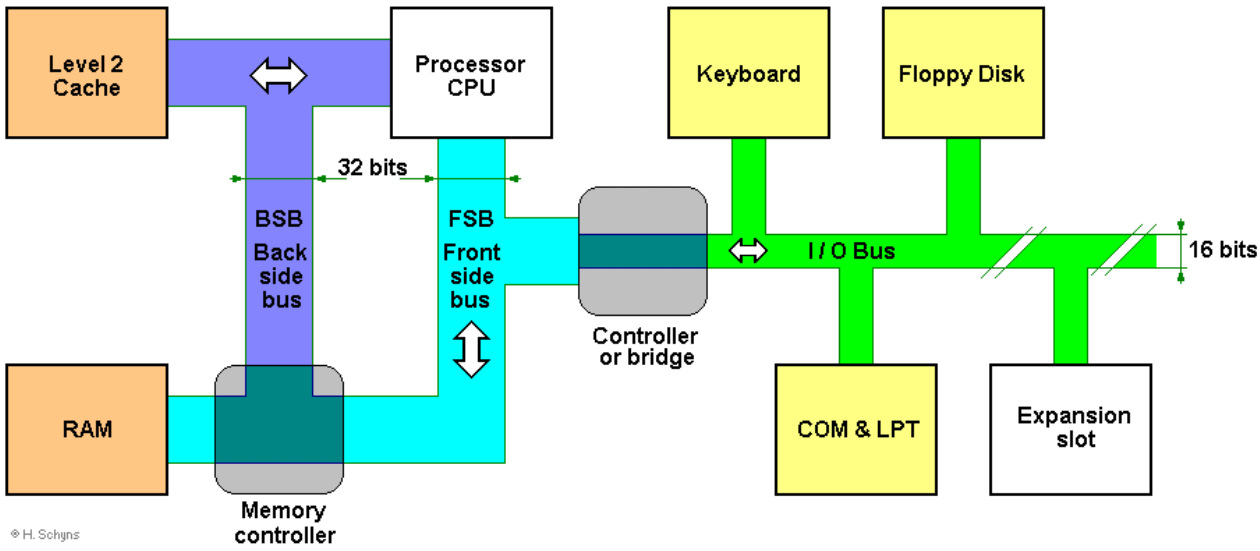

Figure 4.4 Organisation du bus système en Frontside bus et Backside bus

Le rôle du **backside bus** ou **BSB**, parfois appelé **bus processeur** ([1]), est de fournir un chemin direct entre le CPU et le cache L2. Comme ce bus ne véhicule que des données directement utiles au processeur, il doit être extrêmement rapide. Sa largeur et sa fréquence sont donc accordées sur celles du processeur, sans passer par un contrôleur.

Le **frontside bus** ou **FSB**, parfois appelé **bus mémoire** ([2]), plus lent, qui connecte la mémoire RAM :

- au CPU,
- aux autres bus d'entrée/sortie, via le pont déjà rencontré lors de la création du bus ISA. Ce pont deviendra bientôt le pont Nord (*ang.: Northern Bridge*)

Par ailleurs, une communication est nécessaire entre le frontside bus et le backside bus. Comme ces deux bus fonctionnent à des fréquences différentes, la connexion ne peut se faire qu'au travers du contrôleur de mémoire.

De plus, pour communiquer efficacement, ces deux bus doivent impérativement avoir la même largeur. Ceci explique pourquoi les barrettes de mémoire DRAM EDO (32 bits), qui fonctionnaient parfaitement avec les processeurs de type 486 (bus de 32 bits), devaient toujours être utilisées par paire sur les systèmes basés sur les processeurs de type Pentium (bus de 64 bits). Par contre, les barrettes SDRAM de 64 bits peuvent être utilisées seules.

4.3. Le contrôleur de mémoire

Le contrôleur de mémoire est la puce qui règle le flux de données entre le processeur et la mémoire vive :

1 Parce qu'il fonctionne à la vitesse du processeur

2 Parce qu'il fonctionne à la vitesse de le RAM

- il assure l'identification et le test des barrettes de mémoire au démarrage,
- il gère la synchronisation des horloges qui sont situées, l'une du côté de la carte mère, et l'autre du côté des barrettes de mémoire,
- il dirige les demandes du CPU vers les différents bancs de mémoire,
- il gère les cycles de rafraîchissement et empêche le CPU d'accéder à la RAM à ces moments,
- il autorise l'accès direct à la mémoire (*ang.: direct memory access* ou *DMA*) par certains périphériques.

Le contrôleur de mémoire est souvent supervisé par une unité de gestion de la mémoire (*ang.: Memory Managment Unit ou MMU*) qui dépend du CPU. Le MMU gère l'espace mémoire *virtuel* du processeur. Par mémoire virtuelle, on entend d'une part la mémoire réelle des barrettes de RAM et, d'autre part, le disque dur où le MMU stocke les pages de mémoire qui n'ont pas été utilisées récemment par le CPU. La pagination et la mémoire virtuelle sont largement utilisées par Windows.

4.4. Multiples et diviseurs

L'architecture FSB (frontside bus) / BSB (backside bus) permet de faire fonctionner le CPU et la mémoire RAM à des fréquences différentes. Le cache L2, directement accessible par le CPU, permet de limiter l'échange sur le FSB aux demandes de nouvelles informations, celles qui n'ont pas été utilisées dans un passé récent ([1]).

Comme l'écart (*ang.: gap*) entre la vitesse du processeur et celle de la RAM ne fait qu'augmenter, ce système très performant équipe toujours tous les ordinateurs. La différence est qu'aujourd'hui le cache L1 et le cache L2 sont intégrés *dans* le chip du processeur.

Les processeurs de quatrième génération, les célèbres 486, furent les premiers à fonctionner a une vitesse différente de celle du FSB. Dans les machines basées sur le 80486 DX2-50, le CPU travaillait à une fréquence *interne* de 50 MHz alors que la fréquence du bus (horloge *externe*) n'était que de 25 MHz (d'où le sigle DX2).

CPU série 486	Bus système (FSB) Largeur (bits)	Fréquence (MHz)	Facteur
80486 SX 25	32	25	1
80486 DX 33	32	33	1
80486 DX2 50	**32**	**25**	**2**
80486 DX 50	32	50	1
80486 DX2 66	**32**	**33**	**2**
80486 DX4 100	**32**	**40**	**2.5**
Cyrix 5X86 133	**32**	**33**	**4**

De nos jours (2004) les bus système, Frontside bus et Backside bus ont une largeur de **64 bits**. La fréquence du FSB est passée progressivement de 66 MHz (Pentium) à 100 MHz (Pentium II), 133 MHz pour atteindre 200 MHz puis 400 MHz.

Le processeur fonctionne toujours à une fréquence qui est un multiple entier ou demi-entier de la fréquence du FSB. Ainsi, un bus à 66 MHz peut accepter des processeurs fonctionnant à des fréquences différentes comme le montre le tableau ci-dessous :

1 Pour le CPU d'un ordinateur, le passé récent se limite souvent à quelques millisecondes, voire quelques secondes. Au delà, c'est l'antiquité.

CPU famille Pentium	Bus système (FSB) Largeur (bits)	Fréquence (MHz)	Facteur
Intel P100	64	66	1.5
AMD K5-133	64	66	2
Intel P166	64	66	2.5
Cyrix 6X86 166+	64	66	2.5
Pentium Pro 200	64	66	3

Il est impossible de faire fonctionner un Pentium 150 sur ce bus car 150 n'est pas un multiple entier ou demi-entier de 66 (150/66=2.27)([1]). Par contre, un Pentium 150 fonctionnera parfaitement avec un bus cadencé à 60 MHz (facteur 2.5).

La technologie DRAM-EDO était limitée à une fréquence de 66 MHz. L'arrivée des mémoires synchrones (SDRAM, DDR-SDRAM et Rambus DRAM) va repousser cette limite à 100, 133, 200 MHz ([2]) sans pour autant remettre en cause l'architecture FSB/BSB ni le principe des multiples.

Les performances des PC récents dépendent surtout des caractéristiques du bus système : un PC équipé d'un processeur à 400 MHz sera plus rapide s'il est basé sur un FSB cadencé à 133 MHz (3 x 133 MHz) que si le FSB de la carte-mère fonctionne à 100 MHz (4 x 100 MHz). Dans le premier cas, on économise un cycle de processeur à chaque transfert d'information.

4.5. Autres problèmes

Notons que ces hautes fréquences génèrent des bruits et des interférences. Tant qu'elles étaient limitées aux circuits de très petite taille (quelques centimètres pour le Backside bus) ces effets étaient négligeables. Maintenant, le Frontside bus, dont les conducteurs sont plus longs utilise aussi des hautes fréquences. Dès lors, les conducteurs du bus commencent à se comporter comme des antennes émettrices. On imagine sans peine toutes les conséquences désagréables que cela comporte. Il est donc important :

- de concevoir soigneusement les cartes-mères afin de réduire les distances et les effets (cartes multicouches, condensateurs,…)
- de réduire fortement la fréquence lorsqu'il s'agira de transférer de l'information dans les bus d'entrée/sortie, les cartes d'extension et les autres périphériques.

Comme la fréquence du processeur est encore plus élevée, il faut également réduire la longueur du BSB. La tendance actuelle est d'inclure le cache L2 dans le chip du processeur.

4.6. La diversification des bus I/O

L'architecture multi-bus, révolutionnaire à l'époque où elle fut développée par Compaq, est restée un grand succès industriel. Ce succès tient à la grande simplicité du concept :

1 A moins de pratiquer l'overclocking mais nous ne détaillerons pas ces pratiques périlleuses.

2 Les générations récentes (2004) de DDR-DSRAM montent à 266 et 333 MHz.

- tous les composants qui travaillent à la même fréquence et sur le même nombre de bits partagent un même bus (p.ex.: bus de 16 bits et 8 MHz; bus de 16 bits et 16 MHz, bus de 32 bits et 66 MHz)
- les bus s'enchaînent en allant du plus rapide et plus large au plus lent et plus étroit. Entre deux bus, un pont (ou contrôleur) assure la conversion et le synchronisme (Figure 4.5).
- les bus définissent des standards de communication. Chaque périphérique peut lui-même intégrer un contrôleur afin de s'adapter aux conditions de travail du bus.

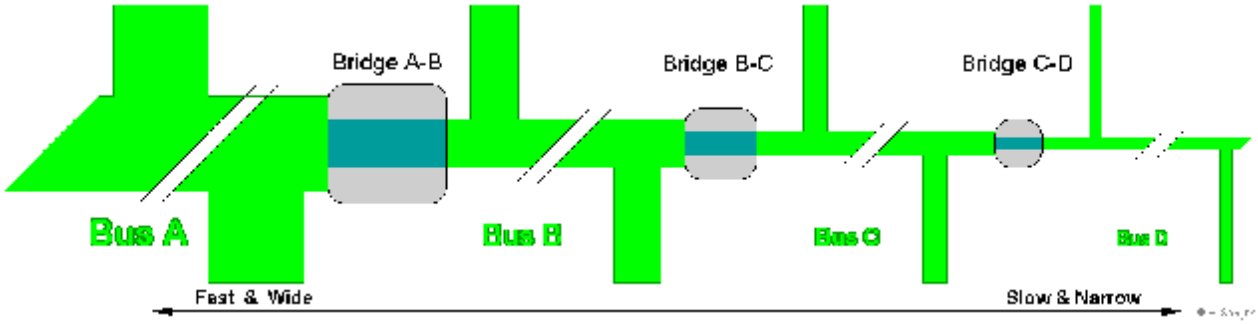

Figure 4.5 Schéma de principe d'un ordinateur à plusieurs bus et contrôleurs.
Chaque périphérique peut lui-même avoir un contrôleur pour s'adapter au standard du bus.

Ce réseau de transfert de l'information dans l'ordinateur ressemble au réseau de distribution électrique.

De grandes lignes à très haute tension (>100 000 V) relient les pays et les régions. Les régions se branchent sur ce réseau grâce à des stations de transformation qui abaissent la tension vers 10 000 V. Cette énergie est distribuée vers les villes au moyen de pylônes et de câbles aériens. Les villes, à leur tour, prélèvent l'énergie sur ce réseau, abaissent sa tension à 220 V et la distribuent aux habitants. Ces 220 V 50 Hz sont le standard du réseau électrique.

Peu importe que votre PC travaille en 5 ou en 12 V, l'important est que son interface d'alimentation électrique soit compatible avec le "bus" 220 V.

5. L'après ISA : une succession difficile

5.1. Un peu d'histoire

Grâce à l'architecture bus/pont, tous les PC actuels comptent un certain nombre de bus d'entrée/sortie : ISA, PCI, AGP, USB, … Aujourd'hui, la situation est a peu près normalisée mais il n'en a pas toujours été ainsi. Dans les années 1980, une demande pour un bus plus puissant s'est développée. De nombreux constructeurs ont essayé de se faire une place au soleil en développant de nouvelles technologies et en tentant d'imposer leur standard. On a alors assisté à l'éclosion d'un bouquet de bus aux sigles les plus divers, souvent éphémères.

Le bus ISA a résisté à cette concurrence pendant des années pour deux raisons principales :

- il était pratiquement le seul à garantir une compatibilité à long terme avec un grand nombre de fabricants
- hormis les applications multimédia, peu de périphériques étaient réellement limités par sa bande passante et sa faible vitesse.

5.2. MCA

IBM a ouvert les hostilités en 1987 en proposant son bus MCA (*ang.: Micro Channel Architecture*). MCA était un chef d'œuvre qui adaptait la meilleure technologie de bus de ses mainframes au monde des PC.

C'était un bus large de **32 bits**, fonctionnant de manière asynchrone avec le bus système à une fréquence de **10.33 MHz**. Avec une telle bande passante, MCA représentait un net progrès par rapport au bus ISA. De plus, ce bus était "**intelligent**" dans la mesure où les cartes se configuraient d'elles-mêmes en fonction des IRQ. Elles pouvaient être installées sans devoir modifier la position de jumpers et autres DIP-switches.

IBM eut la mauvaise idée de breveter son nouveau bus, de créer des adaptateurs spéciaux et de réclamer des redevances à tous les constructeurs qui souhaitaient incorporer ce bus à leurs cartes mères. Dès lors, le bus MCA ne fut utilisé que dans les PC développés par IBM et fut un échec commercial.

5.3. EISA

En 1988--1989, neuf sociétés connues sous le vocable "gang de neuf" (AST, Compaq, Epson, Hewlett-Packard, NEC, Olivetti, Tandy, Wyse et Zenith) décidèrent de contrer le bus MCA de IBM en développant leur propre bus : EISA (*ang.: Extended ISA*).

Ce bus avait une largeur de **32 bits** et fonctionnait à une fréquence de **8 MHz** afin de maintenir une compatibilité totale avec le bus ISA. Tout comme le bus ISA, c'était un bus **synchrone** et sa fréquence de fonctionnement était un sous-multiple de la fréquence du bus système.

Les connecteurs avaient les mêmes dimensions que le slot ISA mais contenaient deux étages de contacts. Les cartes EISA s'enfichaient à fond et atteignaient les contacts situés au fond du connecteur tandis que les anciennes cartes ISA étaient bloquées à mi-hauteur et n'atteignaient que la première rangée.

Le bus EISA était "intelligent". Il permettait l'auto-configuration des cartes, la gestion et le partage des IRQ. De plus, les cartes pouvaient gérer elles-mêmes les transferts de données grâce à la technique du "maître du bus" (*ang.: bus mastering*) qui sera reprise plus tard par le bus PCI. D'un autre côté, EISA utilisait des transferts de données en mode compressé (*ang.: compressed mode*) et en rafale (*ang.: burst mode*), ce qui le rendait très performant. Il fut détrôné par le bus PCI avant de devenir un réel succès.

5.4. VLB

Les utilisateurs des PC souhaitaient de meilleures performances graphiques : meilleure définition, nombre de couleurs accru, animations, etc. Or, ainsi qu'il a été dit plus haut, la lenteur du bus ISA était un frein important à ce développement. Même les améliorations apportées par ses successeurs MCA et EISA étaient insuffisantes.

En 1993, un consortium de constructeurs de moniteurs et de cartes graphiques nommé VESA (*ang.: Video Electronics Standards Association*) introduisit le concept et le standard VLB (*ang.: Vesa Local Bus*)([1]). Le VLB fut largement utilisé sur les cartes mères dotées d'un bus système à 33 MHz conçues pour le processeur 486.

VLB faisait appel à une technologie simple et bon marché : le bus fonctionne avec une largeur de (**32 bits**) et à la vitesse du bus système (**33 MHz**). Dès lors, nul besoin de pont. Le transfert de données se fait de manière synchrone à la vitesse du CPU ou a un facteur entier ou demi entier près.

De plus, pour des raisons de compatibilité, les concepteurs lui ont donné l'aspect d'un connecteur ISA 16 bits sur lequel s'aligne un connecteur d'extension de 2 x 56 broches. Un connecteur VLB peut donc recevoir une carte ISA ([2]). L'ensemble possède les 98 contacts de la carte ISA et les 112 broches de l'extension, soit 210 lignes !

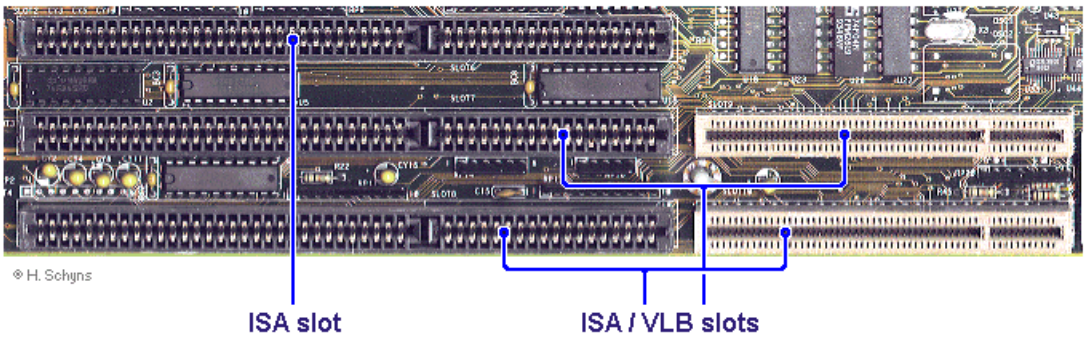

Figure 5.1 Un connecteur VLB ressemble à un connecteur ISA pourvu d'une extension

La simplicité du concept est précisément le problème du VLB : un bus connecté directement au processeur peut accepter un périphérique mais guère plus. Au-delà, le partage et l'arbitrage du bus entre les périphériques commencent à générer des cycles d'attente, à provoquer des interférences et à pénaliser le processeur. En pratique, le VLB fut donc réservé au périphérique qui en avait le plus besoin : la carte graphique. On retrouvera une démarche similaire quelques années plus tard avec l'introduction du port AGP.

1 La norme VESA fut un grand soulagement pour les développeurs de logiciels. Auparavant, chaque carte graphique avait son propre jeu d'instructions et chaque programme devait rechercher à quelle carte il avait affaire avant de lancer les commandes graphiques adéquates. Un cauchemar pour la portabilité !

2 Il est évidemment plus logique de brancher les cartes ISA sur les connecteurs ordinaires et de laisser le ou les connecteurs VLB libres pour les cartes qui en auront vraiment besoin (p.ex. carte vidéo)

6. Le bus PCI : Zorro est arrivé

6.1. Description

En 1993, Intel a développé un nouveau standard de bus E/S : le bus PCI (*ang.: Peripheral Component Interconnect*). Bien que développé par Intel, ce bus n'est pas lié à une famille précise de processeurs. La famille Mac, de Apple, construite autour des processeurs Motorola, utilisent aussi un bus PCI.

Le PCI est une sorte d'hybride entre les concepts ISA et VLB. Il fournit un **accès direct à la mémoire** (*ang.: direct memory access* ou *DMA*) à ses périphériques mais utilise un **contrôleur** (pont) pour se connecter au FSB (frontside bus) et donc au processeur.

Le bus PCI original avait une largeur de **32 bits** de données et opérait à **33 MHz**. Cette bande passante représentait un bond d'un **facteur 8** par rapport au bus ISA; bien plus que tous ses concurrents. Les versions ultérieures (2.1) ont augmenté la largeur à 64 bits et porté la vitesse à 66 MHz ([1]). La version la plus récente, PCI-X, travaille à 133 MHz, ce qui offre une bande passante de 1 GB/s !

La vitesse du bus PCI peut être fixée de manière synchrone ou asynchrone en fonction de la carte mère et des caractéristiques de différents contrôleurs (qui forment le chipset). Habituellement, sur la plupart des PC, le bus fonctionne en mode **synchrone** à une fréquence qui est une fraction entière ou demi-entière de celle du FSB (souvent 1/2) ([2]).

Mais le contrôleur PCI peut aussi faire fonctionner le bus en mode **asynchrone** en régulant la vitesse du bus indépendamment de celle du processeur. Ceci se fait généralement au moyen de cavaliers (*ang.: jumpers*) placés sur la carte mère ou via le programme de setup du BIOS.

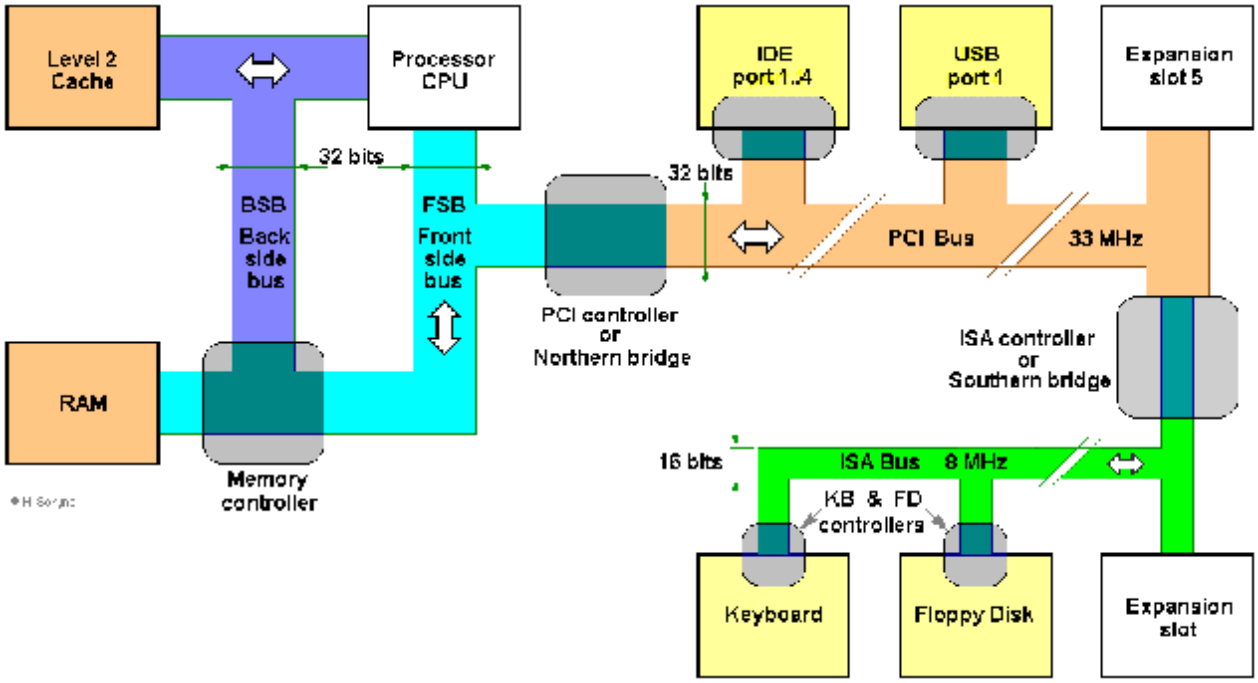

Figure 6.1 Architecture des bus après l'apparition du bus PCI.

1 La version la plus fréquente tourne à 66 MHz avec un bus de 32 bits.

2 Notons à ce sujet que l'overclocking du bus système sur un PC qui utilise le mode PCI synchrone provoque ipso facto l'overclocking des périphériques PCI, ce qui conduit souvent à des problèmes d'instabilité.

La gestion du bus PCI élimine les conflits et les interférences que l'on trouvait sur le VLB. Le bus est donc plus stable et plus fiable que le VLB, ce qui lui permet d'accepter jusqu'à **cinq** périphériques externes. Il est aussi permis d'avoir plusieurs bus PCI indépendants sur une même carte mère quoique cette solution soit rarement implémentée.

Autre avantage : le contrôleur du bus ISA peut dialoguer avec le bus PCI qui se charge du transit des informations vers le FSB (Figure 6.1).

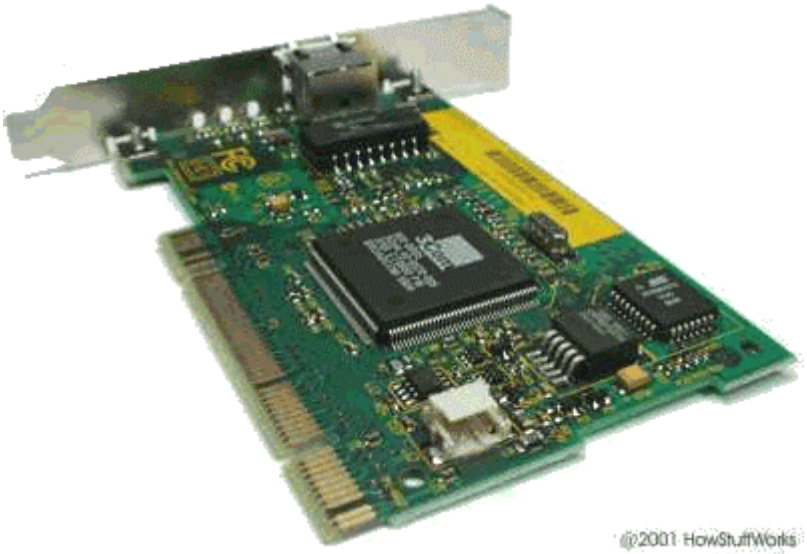

Figure 6.2 Une carte PCI classique.

Le bus PCI existe en versions 32 ou 64 bits. Il supporte les cartes alimentées en 3.3 ou en 5 V. Pour distinguer les différents cas, les connecteurs possèdent une géométrie particulière et un détrompeur qui est orienté dans l'un l'autre sens selon le modèle de carte mère.

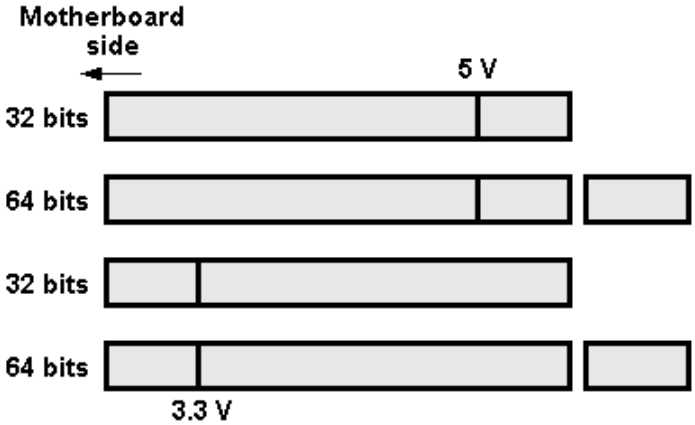

Figure 6.3 Position du détrompeur et de l'extension

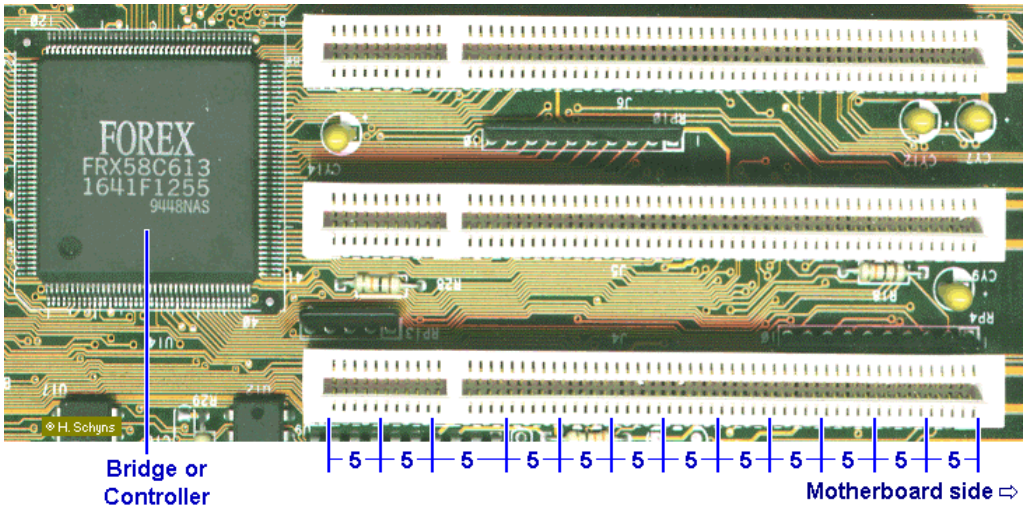

Figure 6.4 Détail des connecteurs PCI 5 V, 32 bits

Par contre, il est intéressant de noter que la plupart des cartes d'extension présentent **deux** encoches. Elles peuvent se connecter indifféremment dans les deux sens (5V et 3.3V).

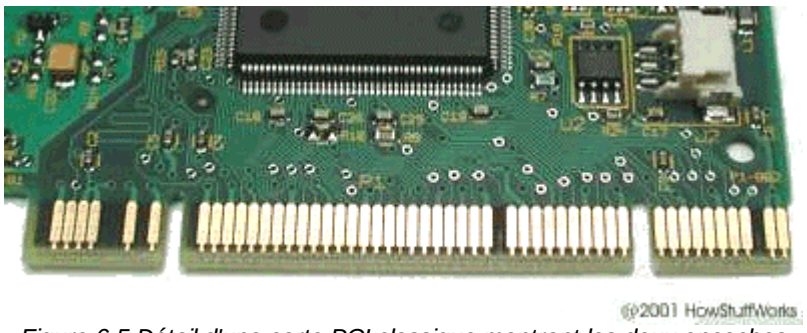

Figure 6.5 Détail d'une carte PCI classique montrant les deux encoches

Les connecteurs PCI comprennent 60 contacts répartis de part et d'autre du détrompeur. Cependant, tous ces contacts ne sont pas utilisés. Une carte PCI classique utilise **47 contacts**. Une carte capable de contrôler le bus sans intervention du processeur (*ang.: bus mastering*) utilise **49 contacts**.

Comment le bus PCI, qui travaille avec 32 bits au moins, peut-il s'en sortir avec 47 contacts alors que le bus VLB, moins puissant, avait déjà besoin de 210 contacts ?

Le même type de problème a déjà été rencontré pour la mémoire DRAM. Lors de l'élargissement du bus d'adresse de 16 à 32 bits, le nombre de broches à fixer sur la puce de mémoire devenait trop encombrant. Les concepteurs de mémoire DRAM ont résolu le problème en recourant au **multiplexage** : l'adresse est envoyée en plusieurs parties et des signaux supplémentaires (RAS et CAS) indiquent de quelle partie il s'agit. Les concepteurs du bus PCI ont utilisé le même type de solution.

Poursuivant l'analogie avec la mémoire DRAM, les ingénieurs ont doté le bus PCI d'un mode de **transmission en rafale** (*ang.: burst mode*). Lorsqu'une adresse est transmise sur le bus, le périphérique renvoie en cadence un flux de données et non un simple jeu de 4 bytes.

6.2. Plug and Play

Intel a jeté les bases du standard PCI dès 1991. Pourtant, celui-ci n'est réellement devenu populaire qu'avec l'arrivée de Windows 95. Pourquoi un tel délai ? Parce que le bus PCI était le premier a présenter la propriété **PnP** (*ang.: Plug and Play - litt.fr.: Brancher et Jouer*) et que Windows 95 était le premier système capable d'exploiter cette propriété. Grâce à l'arrivée de Windows 95, la demande de PC

équipés du bus PCI explosa et les cartes d'extension ISA furent bientôt regardées comme des antiquités.

Dans le concept PnP, le contrôleur du bus PCI se charge de l'identification des cartes d'extension. Il transmet cette information au BIOS qui la transmettra plus tard au système d'exploitation. D'autre part, le contrôleur de bus dialogue avec le BIOS pour distribuer les ressources système de manière à éviter les conflits entre périphériques. Auparavant, ces tâches étaient accomplies soit manuellement en positionnant des cavaliers (*ang.: jumpers*) sur les cartes d'extension, soit grâce à un programme utilitaire fourni avec le matériel.

Par ressources système, on entend ([1]) :

- les requêtes d'interruption ou **IRQ**
- les canaux d'accès direct à la mémoire ou **DMA**
- les plages d'**adresses** de mémoire (*ang.: memory addresses*)
- les ports d'entrée/sortie (*ang.: I/O ports*)

En clair, sur un ordinateur "Plug and Play", l'utilisateur peut facilement connecter un périphérique ou insérer une nouvelle carte d'extension. Celle-ci sera automatiquement reconnue et configurée par le système afin de fonctionner harmonieusement avec le matériel déjà présent. Le principe est élégant, mais sa mise en place exigea un effort de coopération considérable entre les différents acteurs du monde informatique ([2]). En effet, quatre éléments sont nécessaires à l'implémentation d'un système PnP :

- des **cartes** d'extension PnP

  Les cartes doivent fournir un certain nombre d'informations techniques à la demande telles que leur fonction (son, image,...), l'identification du fabricant, etc.
- un **BIOS** PnP

  Le BIOS doit activer l'option PnP et détecter ces périphériques. Il doit aussi lire le fichier de configuration (ESCD) pour retrouver l'information nécessaire.
- un **fichier** de configuration (*ang.: **E**xtended **S**ystem **C**onfiguration **D**ata*)

  Le fichier ESCD contient toutes les données relatives aux périphériques PnP déjà installés
- un **système d'exploitation** PnP

  Tout système d'exploitation tel que Windows 95b / 98 / Me et suivants qui dispose des fonctions nécessaires pour compléter la configuration amorcée par le BIOS

Ainsi qu'il a été dit, le principe de fonctionnement du PnP et relativement simple. Admettons qu'un utilisateur ajoute une nouvelle carte à sa machine. Ceci fait, il appuie sur le bouton pour redémarrer le PC.

- la mise sous tension provoque le chargement en mémoire du BIOS;
- le BIOS parcourt le bus PCI à la recherche de matériel. Il envoie un signal à chaque connecteur et demande à chaque carte de s'identifier;
- chaque périphérique connecté renvoie son identification au BIOS via le bus PCI;
- le BIOS parcourt le fichier ESCD pour vérifier si les identifications qu'il vient de recevoir s'y trouvent déjà;

1 Les ressources système IRQ, DMA, etc. sont développées dans un autre chapitre.

2 De fait, dans la réalité, les choses étaient nettement moins brillantes et les utilisateurs avaient surnommé ce système "Plug and Pray" (*fr.: brancher et prier*). Heureusement, les choses se sont arrangées par la suite.

- s'il s'agit d'une nouvelle carte qui vient d'être installée, le fichier ESCD ne contient encore aucun renseignement à son sujet. Le BIOS lui attribue un jeu de ressources (IRQ, DMA, etc.) compatibles avec celles utilisées par les périphériques déjà répertoriés et sauve ces nouvelles données dans le fichier ESCD;
- le système d'exploitation (*ang.: **O**perating **S**ystem*) démarre à son tour. Il explore le fichier ESCD, le bus PCI et sa propre base de données (registry) à la recherche des informations de configuration;
- s'il s'agit d'un nouveau périphérique, l'information est absente de la base de données et l'OS annonce fièrement qu'il a localisé un nouveau périphérique et qu'il recherche le pilote (*ang.: driver*) correspondant;
- s'il n'arrive pas à identifier le périphérique ou à localiser le pilote adéquat, il ouvre une fenêtre et demande l'assistance de l'utilisateur;
- la configuration et les références du pilote sont sauvées dans la base de données;
- éventuellement, le système d'exploitation demande de redémarrer le système.

6.3. IRQ internes

Le nombre d'IRQ disponible sur un système est très restreint. Or chaque périphérique, chaque carte d'extension a besoin d'un numéro d'IRQ différent afin de s'identifier sans confusion auprès du CPU. Avec les cartes ISA, les conflits étaient fréquents et faire fonctionner plusieurs cartes simultanément pouvait tourner au cauchemar.

Le bus PCI a résolu le problème en utilisant sont propre système interne d'interrupt. Dès lors, le contrôleur PCI agit comme un routeur placé entre un LAN et l'Internet : vis à vis des cartes et périphériques qui utilisent ses services, il utilise son propre jeu d'IRQ. Vis à vis du CPU, il utilise les IRQ du système. Entre les deux, il maintient une table de correspondance. Evidemment, si le nombre de cartes d'extension n'est pas trop élevé et si les ressources système sont suffisantes, la correspondance est immédiate.

6.4. Maîtrise du bus

La norme PCI permet aux périphériques de prendre la maîtrise totale du bus pendant un court intervalle de temps (ang.: **bus mastering**) ce qui améliore considérablement les performances du système.

Lorsque deux périphériques souhaitent prendre le contrôle du bus au même moment, le conflit est arbitré par le chipset. Celui-ci s'assure qu'aucun périphérique, y compris le processeur, ne bloque le bus en se réservant en permanence un accès exclusif.

Avec ce système, le bus PCI devient une sorte de LAN (*ang.: local area network*) au sein de la carte mère, permettant aux différents périphériques de parler les uns aux autres via un canal de communication qu'ils partagent et qui est géré par le chipset.

7. Le bus AGP

Le premier **bus AGP** (*ang.: Accelerated Graphics Port*) a été développé par Intel et est sorti en même temps que le Pentium II en mai 1997. Il est connecté au Pont Nord et au contrôleur de mémoire (Figure 7.6).

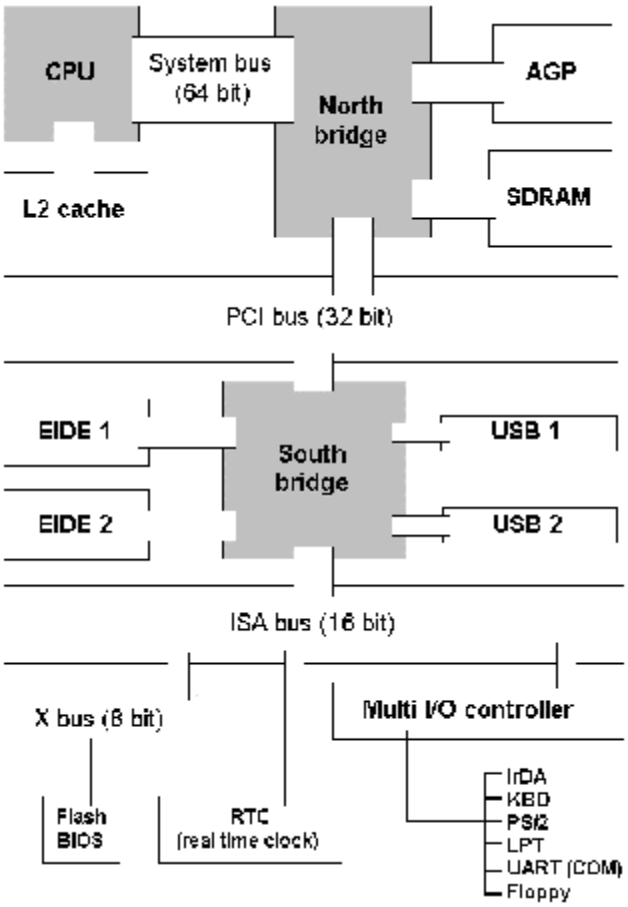

Figure 7.6 Intégration du bus AGP dans l'architecture générale (source: Michael Karbo)

Le bus AGP sert essentiellement voire uniquement à la connexion de cartes vidéos. En effet, sa conception est basée sur celle du bus PCI mais avec des améliorations considérables autorisant les débits élevés exigés par les graphiques 3D.

L'avantage du bus AGP est de permettre le partage de la RAM pour stocker des textures, ce qui diminue la quantité de mémoire à installer sur la carte graphique elle-même, ce qui réduit son coût. L'inconvénient est que, pour cette raison, un PC équipé d'un bus AGP a besoin d'une plus grande quantité de mémoire RAM puisqu'une partie de celle-ci sera dédicacée à la carte graphique ([1]).

Le bus AGP a une largeur de 32 bits. La première version était cadencée à 66 MHz (soit deux fois plus que le bus PCI original), ce qui offrait une bande passante de 264 MBps. Au fil du temps, plusieurs versions offrant une bande passante double, quadruple ou octuple sont apparues, respectivement nommées AGP 2X, 4X et 8X (2112 MBps ou 2,06 GBps).

1 A l'époque, ce point a perturbé bon nombre d'utilisateurs qui, bien qu'ayant placé des barrettes de RAM pour 256 ou 512 MB ne retrouvaient pas ce nombre lorsqu'ils interrogeaient l'OS pour connaître la quantité de mémoire disponible.

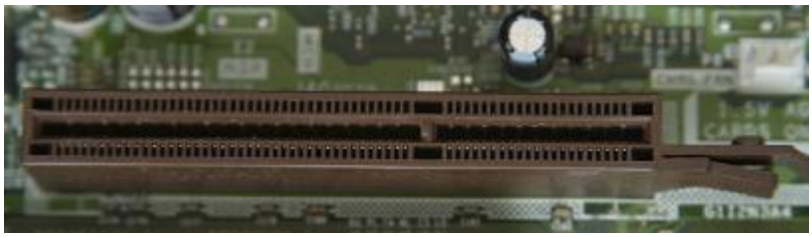

Figure 7.7 Détail d'un connecteur AGP

Habituellement, les cartes mères ne comportent qu'un seul connecteur AGP puisqu'elles ne sont connectées qu'à un seul moniteur (Figure 7.7). Le connecteur est généralement brun, placé parallèlement à côté des connecteurs PCI et légèrement en retrait.

Figure 7.8 Détail d'une carte graphique AGP

La carte AGP possède à l'arrière, un petit ergot caractéristique qui rend son insertion dans le connecteur quelque peu délicate (Figure 7.8). Elle présente une ou deux encoches dont la position définit la tension d'alimentation de la carte (1.5V ou 3.3V).

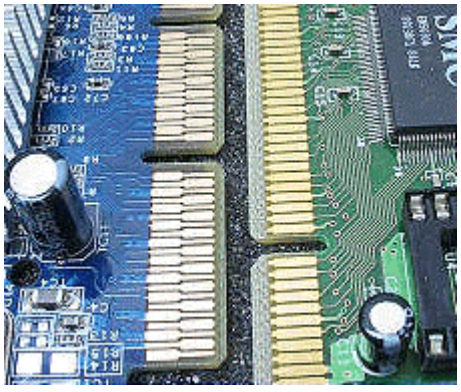

Figure 7.9 Détail des contacts d'une carte AGP (gauche) et PCI (droite)

Les contacts d'une carte AGP sont alignés sur deux étages contrairement à ceux d'une carte PCI (Figure 7.9).

8. Les bus "série"

(à développer)

9. Sources

9.1. Ouvrages

- **Upgrading and Repairing PC's**
  Scott Mueller
  QUE. 19th edition
  ISBN 0-7897-3954-2

Probablement la meilleure référence et la plus complète pour tout ce qui concerne les caractéristiques techniques des composnts et l'architecture des PCs