

1 Présentation d'un système standard à microprocesseur

1.1 Introduction.

Comme présentée dans le cours de logique, la donnée fondamentale sur laquelle travaille un ordinateur est le bit. Les deux états 0 et 1 que peuvent prendre une variable binaire peuvent être associés à l'absence ou à la présence d'un phénomène physique qui pourrait être une tension électrique:

'1' présence d'une tension. Exemple 5volt.

'0' absence d'une tension. Exemple 0volt.

L'ensemble des composants nécessaires à la réalisation d'un système standard à microprocesseur vont donc pouvoir s'échanger des données binaires sous forme de signaux électriques. Ces signaux transitent sur de petits fils de cuivre que l'on appelle des pistes, celles-ci étant fixées sur un support que l'on appelle le circuit imprimé.

L'ensemble composé du circuit imprimé et de ses divers composants s'appelle la carte mère. Nous allons dans le cadre de ce chapitre aborder les composants les plus standard que l'on retrouve sur les cartes mère des ordinateurs.

1.2 Le microprocesseur.

1.2.1 Ses éléments internes.

Il est à la base de toutes les opérations arithmétiques et logiques que le micro-ordinateur va effectuer sur les données binaires. Les principaux éléments constitutifs de ce composant sont:

- Une horloge qui rythme la cadence de travail du composant. Plus la fréquence de l'horloge sera élevée, plus le processeur effectuera un nombre d'instructions par seconde important (MIPS: Millions d'instruction par seconde).
- Une unité arithmétique et logique (ALU ou UAL) responsable des calculs sur les données. Une des caractéristiques de cette unité est la taille de la donnée sur laquelle ces calculs vont porter. Une addition de deux entiers codés sur 32 bits prendrait un seul cycle pour une unité arithmétique et logique de 32 bits tandis qu'elle prendrait 4 cycles pour une unité ne pouvant traiter que 8bits à la fois.

On parlera, en fonction de cette taille, de microprocesseur 16bits, 32bits ou 64bits

- Les registres internes. Lorsque le microprocesseur traite les données (lorsqu'il exécute des instructions), le microprocesseur stocke temporairement les données dans de petites mémoires internes que l'on appelle les registres. Parmi les registres les plus importants, on retrouvera:
 - le registre accumulateur qui stocke les données et les résultats des opérations arithmétiques et logiques.
 - le registre d'état contenant des indicateurs sur le résultat de la dernière instruction exécutée: nombre positif, négatif, nul...
 - le registre ordinal encore appelé PC (programm counter). Ce registre contient l'adresse de la prochaine instruction à traiter.
- Les bus externes. Les données que le microprocesseur doit traiter ainsi que les instructions qu'il doit exécuter se trouvent dans des circuits externes que l'on appelle mémoire. Pour lire une instruction ou une donnée dans de tels circuits, il doit fournir le numéro de la case mémoire dans laquelle elle est stockée. Ce numéro s'appelle une adresse et est également représentée par un ensemble de bits. Plus le nombre de bits est important, plus le nombre de case mémoire que le microprocesseur pourra accéder sera important. On parle de la capacité mémoire ou de l'espace adressable.

Taille de l'adresse

capacité mémoire

16 bits	65536 emplacements mémoire
20 bits	1048576 emplacements mémoire
32 bits	4294967296 emplacements mémoire

Dans tous les PC, la capacité mémoire est donnée en octets. La taille est exprimée suivant une convention d'écriture en permettant sa simplification: 1ko = 1024 octets.

65536 octets équivaudra donc à 64ko

1048576 octets équivaudra à 1Mo (1024 x 1024)

4294967296 octets équivaudra à 4Go

1.2.2 Les bus externes.

Sur une carte mère, l'ensemble des pistes servant à faire transiter des informations binaires de même nature s'appelle un BUS. On parlera dans ce cas ci du bus d'adresses. On dira que la taille du bus d'adresses est de 16bits, 20 bits ou 32 bits.

L'échange de la donnée entre la mémoire et le microprocesseur se fera par le bus de données. La taille de ce bus est en général identique à la taille de la donnée que l'unité arithmétique et logique est capable de traiter en un seul cycle. Pour des raisons philosophiques, ce n'est pas toujours le cas:

Pour augmenter la vitesse de transferts des données entre le microprocesseur et la mémoire, la taille du bus est parfois plus importante:

Pour une cadence d'horloge de 100Mhz avec une taille de bus de données de 8 bits, on obtiendra un transfert de $100\text{M} \times 8\text{bits} = 800\text{Mbits/sec}$ ou 100Mo/sec

Pour une cadence d'horloge de 100Mhz avec une taille de bus de données de 32bits, on obtiendra un transfert de $100\text{M} \times 32\text{bits} = 3,2\text{Gbits/sec}$ ou 400Mo/sec

Ces calculs tiennent compte d'un transfert par cycle d'horloge.

Pour diminuer le coût du microprocesseur, la taille du bus de données est parfois plus petite. Nec V20 a sorti dans les années 1980 un processeur compatible Intel8086 (avec une UAL de 16 bits) mais dont la taille du bus de données n'était que de 8 bits. Il fallait donc deux cycles de lecture en mémoire pour charger les registres internes du microprocesseur.

Lorsque le microprocesseur fournit une adresse pour accéder à une case mémoire, il est impératif que la mémoire sache s'il s'agit d'un cycle de lecture ou d'un cycle d'écriture. Ce renseignement est généralement fourni par le microprocesseur par un signal que l'on appelle read/write. Ce signal de contrôle se retrouve dans un bus que l'on appelle bus de commande ou bus de contrôle.

1.2.3 L'architecture.

1.2.3.1 Le pipelining.

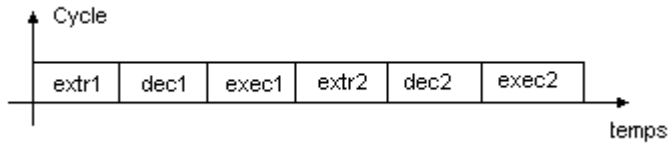
Un pipeline est un ensemble d'unités effectuant un travail à la chaîne dans le but final de traiter les instructions. Lorsque l'on augmente la profondeur du pipeline, on augmente le nombre d'unités, et on réduit donc le travail que chacune doit effectuer.

Le Pentium utilise un pipeline entier de cinq étages similaire à celui du i486 tandis que le Pentium 4 utilise une architecture similaire appelée NetBurst se distinguant notamment par une profondeur de pipeline jamais atteinte à ce jour par un processeur x86, puisqu'elle est de 20 niveaux, contre 10 par exemple pour une architecture du Pentium Pro.

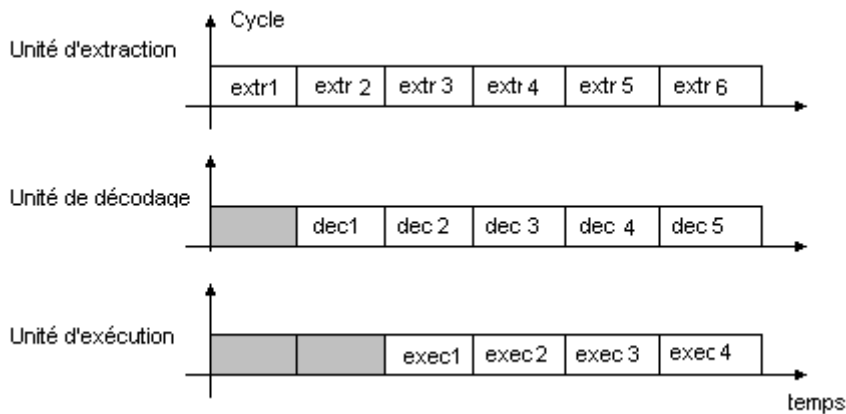
Pour bien comprendre l'utilité de cette architecture, considérant un processeur hypothétique composé de trois unités:

- L'unité d'extraction qui est responsable de la lecture en mémoire de l'instruction à exécuter.
- L'unité de décodage qui est responsable du décodage de l'instruction lue par l'unité précédente.
- L'unité d'exécution.

Pour un processeur simple, nous obtenons le chronogramme suivant:



Pour un processeur disposant d'une architecture pipeline à trois unités, nous aurons le chronogramme suivant:



En analysant les deux chronogrammes, on peut s'apercevoir que sur un laps de temps identique, un processeur simple a exécuté deux instructions tandis qu'un processeur en architecture pipeline cadencé avec la même horloge en a déjà exécuté 4.

Le problème majeur est bien évidemment de remplir le pipeline de façon optimale. En effet, une instruction de saut ou l'attente d'une donnée non disponible arrêtent le fonctionnement du pipeline. Cette tâche d'organisation échoit à l'unité de contrôle du processeur, mais aussi au compilateur. Le processeur Pentium Pro dispose ainsi de deux pipelines entiers (qui effectuent des calculs sur des nombres entiers) et d'un pipeline flottant, qui se greffe en fait sur l'étage de décodage de l'un des pipelines entiers.

Un processeur scalaire regroupe, pour sa part, plusieurs unités d'exécution. Il est de cette façon en mesure d'appliquer la même opération à un ensemble de données variables. Les constructeurs parlent dès lors d'architecture SIMD (Single Instruction Multiple Data). C'est sur ce principe que repose la technologie MMX d'Intel.

Dans les microprocesseurs à partir du 486, on trouve autour de cette architecture la possibilité de travailler avec une cadence d'horloge interne, cadencant les échanges entre unités, différente de la cadence permettant d'assurer les échanges sur le bus extérieur. Cet artifice est possible en utilisant une unité d'interface de bus permettant d'adapter les différences de cadence.

On retrouve comme premiers processeurs ayant cette particularité:

- Le 486DX2 (50Mhz sur le bus interne 25Mhz sur le bus externe).
- Le 486DX4 100 (100Mhz en interne et 25Mhz en externe).

Cette technique se retrouve encore actuellement sur les microprocesseurs: ils utilisent une horloge externe de 100Mhz ou 133Mhz tandis que l'horloge interne peut dépasser le 1Ghz.

1.2.3.2 L'exécution dynamique.

L'exécution dynamique repose sur l'utilisation simultanée de plusieurs techniques :

1. La prédiction de branchement consiste à deviner l'emplacement de la prochaine instruction à exécuter et donc à la charger dans le pipeline. En effet, un saut oblige le processeur à recharger ses pipelines avec des données pertinentes. Pendant ce temps, le programme perd l'accélération apportée par le pipeline. Concernant le Pentium II, Intel annonce une capacité de prédiction record de 90%.
2. L'analyse de flux permet au processeur de modifier l'ordre d'entrée des instructions d'un programme dans le pipeline pour en optimiser le taux d'occupation. Par exemple, sur les Pentium, il s'agit de choisir entre les deux pipelines entiers en fonction d'un éventuel calcul flottant - obtenu par la dérivation du pipeline entier. Le tout, bien sûr, en tenant compte du temps nécessaire à l'exécution des instructions. Le nouvel ordre établi peut être complètement différent de celui imaginé par un piètre compilateur. L'exécution spéculative, enfin, fonctionne de concert avec la prédiction de branchement.
Ainsi, l'architecture scalaire du Pentium II exécute les instructions des différentes portions de code envisageable (après un saut, par exemple).

1.2.3.3 L'architecture CISC.

L'architecture CISC (*Complex Instruction Set Computer*, ce qui signifie "ordinateur avec jeu d'instructions complexes") est utilisée par tous les processeurs de type x86, c'est-à-dire les processeurs fabriqués par Intel, AMD, Cyrix, ...

Les processeurs basés sur l'architecture CISC peuvent traiter des instructions complexes, qui sont directement câblées sur leurs circuits électroniques, c'est-à-dire que certaines instructions difficiles à créer à partir des instructions de base sont directement imprimées sur le silicium de la puce afin de gagner en rapidité d'exécution sur ces commandes.

L'inconvénient de ce type d'architecture provient justement du fait que des fonctions supplémentaires sont imprimées sur le silicium, d'où un coût élevé.

D'autre part, les instructions sont de longueurs variables et peuvent parfois prendre plus d'un cycle d'horloge ce qui les rend lentes à l'exécution étant donné qu'un processeur basé sur l'architecture CISC ne peut traiter qu'une instruction à la fois!

1.2.3.4 L'architecture RISC

Contrairement à l'architecture CISC, un processeur utilisant la technologie RISC (*Reduced Instruction Set Computer*, dont la traduction est "ordinateur à jeu d'instructions réduit") n'a pas de fonctions supplémentaires câblées. Cela impose donc des programmes ayant des instructions simples interprétables par le processeur. Cela se traduit par une programmation plus difficile et un compilateur plus puissant. Cependant vous vous dites qu'il peut exister des instructions qui ne peuvent pas être décrites à partir des instructions simples...

En fait ces instructions sont tellement peu nombreuses qu'il est possible de les câbler directement sur le circuit imprimer sans alourdir de manière dramatique leur fabrication.

L'avantage d'une telle architecture est bien évidemment le coût réduit au niveau de la fabrication des processeurs l'utilisant. De plus, les instructions, étant simples, sont exécutées en un cycle d'horloge, ce qui rend l'exécution des programmes plus rapides qu'avec des processeurs basés sur une architecture CISC.

De plus, de tels processeurs sont capables de traiter plusieurs instructions simultanément en les traitant en parallèle.

Bien sûr, les implications sont nombreuses, mais s'il ne faut en retenir qu'une ce doit être que, sur une machine RISC, la diminution de la complexité matérielle est compensée par un compilateur très évolué.

1.2.4 Les interruptions matérielles.

Prenons le cas d'un ordinateur et réfléchissons à ce qui se passe lorsque l'utilisateur enfonce une touche du clavier provoquant une réaction au niveau de votre application ou de votre système d'exploitation. Deux cas de figure peuvent être envisagés:

1. Le microprocesseur scrute à intervalle régulier votre clavier et l'ensemble des autres périphériques, par l'intermédiaire en général d'un registre d'état, afin de déterminer si ces derniers sont demandeur d'un service. Cette solution est simple d'un point de vue matériel mais si aucun périphérique n'est demandeur, cette scrutation représente un gaspillage de temps.
2. Le périphérique demandeur d'un service envoie une demande vers le microprocesseur qui interrompt son travail en terminant l'instruction en cours d'exécution et exécute le code en fonction du périphérique demandeur. Cette solution est plus complexe d'un point de vue matériel car elle nécessite la présence d'une ou plusieurs bornes spéciales sur le microprocesseur permettant aux périphériques d'envoyer les demandes. Ces bornes s'appellent bornes d'interruptions.

On retrouve deux grandes familles:

- La borne d'interruption masquable que l'on peut désactiver par programmation. Elle est souvent identifiée par le nom INT.
- La borne d'interruption non masquable qui est identifiée par le nom NMI (non maskable interrupt).

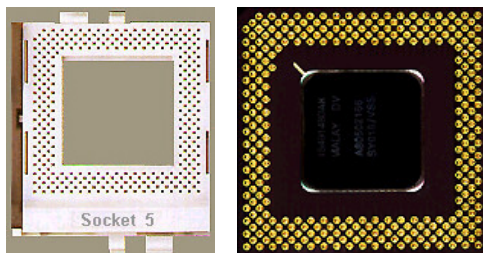
Comme il existe plus d'un périphérique demandeur mais généralement une seule borne INT sur le microprocesseur, on centralise les demandes avec une gestion de priorité sur un circuit externe que l'on appelle PIC (programmable interrupt controller) qui sera étudié dans le cadre d'un paragraphe ultérieur.

1.2.5 Les sockets.

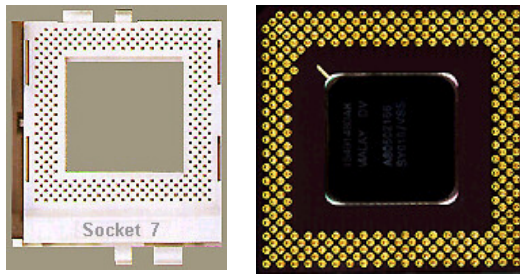
On appelle par socket le type de support du microprocesseur. Initialement, les microprocesseurs étaient montés sur des supports d'où ils ne pouvaient être extraits que difficilement au moyen de pinces spéciales. Dès l'apparition des processeurs 486, apparaissent aussi les supports appelés ZIF (zero insertion force). Ces supports disposent d'un levier latéral permettant de pouvoir ôter facilement le microprocesseur par un système de verrouillage/déverrouillage.

En fonction de la taille des bus, de la cadence des horloges et quelques fois des tensions utilisées, les supports évoluent. Voici une liste non exhaustive des différents supports des microprocesseurs depuis le Pentium pour lesquels ils étaient prévus et dans certains cas d'une image le représentant.

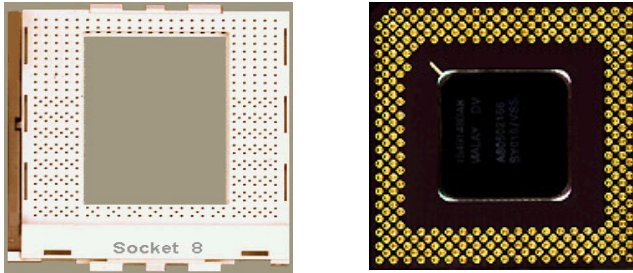
- **Le socket 5** est un support 320 broches pour les processeurs P54C ou P54CS cadencés entre 75Mhz et 166Mhz. Ces processeurs sont alimentés en 3,3Volt et ont un boîtier de type SPGA (Staggered Pin Grid Array) . On va donc retrouver les Pentium 75,90,100,120,133,150,166,200



- **Le socket 7** est un support 321 broches pour les microprocesseurs P54C or P54CS cadencés entre 75Mhz to 200Mhz et alimentés soient en 3,3volt ou 2,5volt.



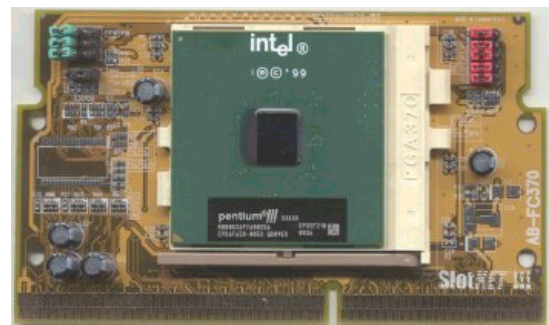
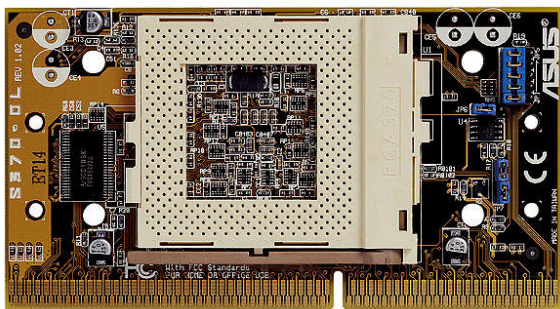
- **Le socket 8** est un support 387 broches pour les microprocesseurs Pentium Pro cadencés de 150 à 200Mhz et alimentés en 3,1Volt ou 3,3Volt. On va donc retrouver les Pentium Pro 150,166,180,200



- **Slot 1** est un support 242 contacts pour les microprocesseurs Pentium II, III(Katmai) suivant un format de boîtier SECC (Single Edge Contact Cartridge) et celeron slot 1 suivant un format SEPP (Single Edge Processor Package). Ces processeurs sont alimentés en 2,8Volt ou 3,3Volt.



- **Le socket 370.**



La photo ci dessus représente une adaptateur permettant aux propriétaires d'une carte mère équipée d'un slot1 de pouvoir utiliser la dernière génération des processeurs Pentium III baptisés

Coppermine et alimentés uniquement en 1,5volt. Il est clair qu'une telle possibilité ne s'offrirait qu'aux propriétaires d'une carte mère permettant d'ajuster la tension d'alimentation du processeur à une valeur inférieure à 1,65volt. Des cartes mère ont été vendue directement équipées du support 370, permettant de descendre en deçà de 1,3volt pour la tension d'alimentation du processeur.

- **Les sockets 423/478**



Le Pentium 4 pour le support 478 se trouve à gauche de l'image tandis que le Pentium 4 pour le support 423 se trouve à droite.

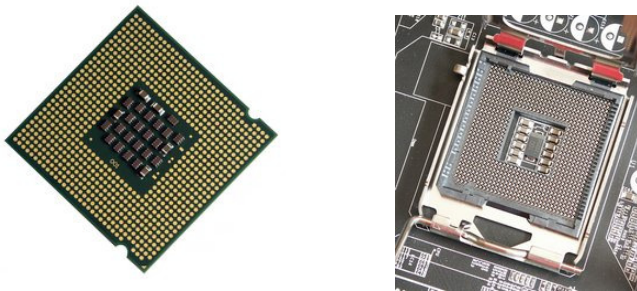
Les performances des Pentium 4 sont les mêmes pour les deux types de boîtier. Ils supportent tous les deux l'architecture NetBurst incluant la technologie hyper pipeline, un bus système à 100Mhz. Ils utilisent un procédé de fabrication à 0.18 et incluent une cache intégrée L2 variant de 256K. La raison d'utiliser un boîtier différent est d'être compatible électriquement avec les futurs processeurs Pentium4. Le Pentium 4 est alimenté en 1,5volt.

Cette première version du processeur appelée Willamette offre des cadences allant de 1.3GHz à 2GHz.

La seconde évolution pour le pentium 4, le Northwood, est gravé en 0.13 μ , il de ce fait, peut intégrer une cache L2 de 512 Ko. Les modèles sortis vont de 1.6 GHz à 3.2 GHz et pour différencier certains modèles de leur prédécesseurs, un A a été rajouté à leur noms. Le Northwood a lui aussi connu quelques évolutions, tout d'abord le passage du FSB de 100 MHz à 133 MHz (533 MHz virtuels) à partir du modèle à 2.27 GHz a fait passer la version de A à B. Pour le modèle 3.06 GHz, une innovation a fait son apparition et s'appelle HyperThreading. La dernière évolution fut le passage du FSB de 133 à 200 MHz (800 MHz virtuels) à partir du modèle 2.4 GHz, ce qui a fait passer leur version de B à C.

Pour des raisons un peu étranges, Intel mets sur le marché un processeur en technologie 13 micron présentant une cache de niveau 2 de 512 Ko et une cache de niveau 3 de 2 Mo avec un FSB permettant d'atteindre des cadences virtuelles de 800 MHz et 1066 Mhz.

- **Les sockets LGA775**



Le dernier né de la famille Intel est le processeur Pentium 4 HT (pour HyperThreading) nom de code Prescott. Ce processeur utilise un procédé de fabrication à 0.09 micron et intègre une cache L2 de 1Mo. Le bus système de ce processeur est cadencé à 200 Mhz qui avec la technologie

quadpumped permet d'obtenir un Front Side Bus de 800 Mhz. Ce processeur peut s'obtenir dans des versions d'horloge allant de 2,8GHz à 3,8 GHz.

Ce processeur intègre un jeu d'instructions supplémentaire SSE3 et voit son niveau de pipeline passer de 20 à 31. La tension d'alimentation de ce processeur peut varier entre 1.25 et 1.4Volts.

Lorsque l'on analyse les caractéristiques des différents processeurs Pentium, on peut en tirer les conclusions suivantes:

1. La fréquence du bus externe n'a que très peu augmenté. Elle était de 100Mhz pour le Pentium I, elle est de 100Mhz pour les premiers Pentium IV pour atteindre maintenant 200MHz pour les dernières versions (266Mhz pour le pentium 4 Extrême Edition). Les principales évolutions sont l'augmentation de la taille du bus de données permettant des débits plus importants et le fait que dans un transfert en rafale entre le processeur et la mémoire, on utilise soit les deux flancs d'horloge pour ainsi doubler le débit soit un artifice de chez Intel appelé Quad pumped et permettant avec la même horloge de quadrupler le débit (4 mots de 64bits) .
2. La fréquence du bus interne a fortement augmenté. D'un Pentium I cadencé à 60Mhz, on arrive à un Pentium IV cadencé à 3.8Ghz. Cette augmentation de la fréquence se traduit par une diminution de la tension d'alimentation. Lorsque l'on s'intéresse aux circuits logiques, on constate qu'à fréquence élevée, une grande partie de la puissance est dissipée par les capacités parasites; on retrouve dans le calcul de la puissance la formule suivante: $C_p \times V_{cc}^2 \times \text{Freq}$. Si l'on désire limiter la puissance dissipée tout en augmentant la fréquence, on peut agir sur la tension d'alimentation.
3. L'intégration. Avec ses 42 millions de transistors, le Pentium IV occupe une surface de 217 mm² contre 170 mm² pour le Pentium III. A titre d'information, le 80386 comprenait 275000 transistors pour une surface identique à celle du Pentium IV. La technologie de gravure était de 1.5µ pour le 80386 alors qu'elle est de 0.13µ pour le Pentium IV. Les derniers pentium sortis sur le marché ont actuellement une gravure à 0.09µ.

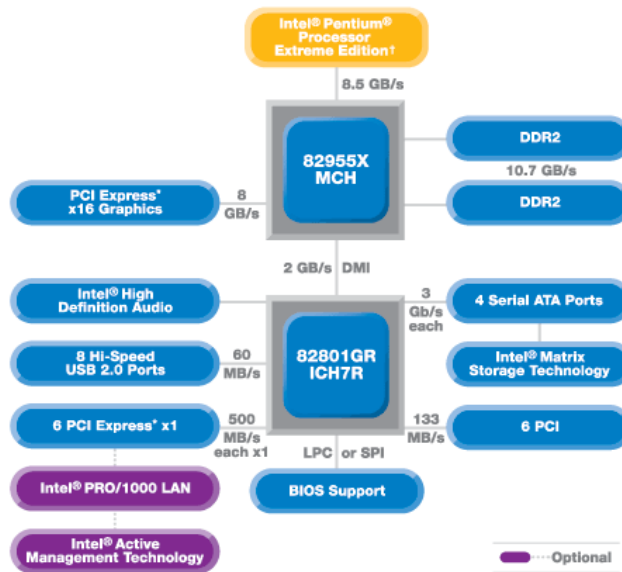
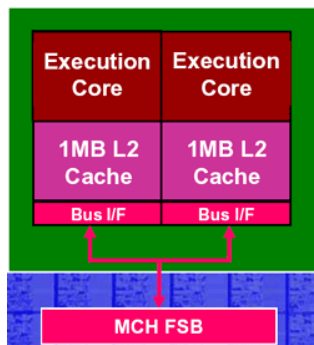


Question intégration, on atteint 55 Millions de transistors pour une gravure à 0.13 µ et pour le dernier né, 125 millions de transistors pour une surface de 112 mm² dont l'analyse de la structure interne laisse apparaître des zones du processeur non activées dans la version actuelle du processeur. L'arrivée du 0.065µ chez Intel est prévue pour fin 2005 / début 2006.

2005 voit l'apparition chez Intel du processeur Intel Pentium D, D pour Double Cœur. Le principe du Dual Core est très simple : disposer deux processeurs complets sur un seul socket, ce qui permettra de doubler la puissance théorique du processeur sans avoir à augmenter la fréquence ou à augmenter la complexité de l'architecture de celui-ci. Les deux processeurs sont gravés néanmoins sur la même puce de silicium. En plus de ce processeur, Intel propose sa version Extrême Edition 840 qui intègre en plus des deux cœurs, le système Multi Threading, ce qui porte en fait le processeur à 4 processeurs logiques. Le nom de code de ces deux architectures : SmithField.

Pour compléter sa gamme, la version Extrême Edition se voit doter d'un nouveau chipset dont la représentation graphique suivante, nous donne une idée des performances et des modifications apportées par rapport à l'ancienne version.

Nous pouvons remarquer la possibilité d'utilisation des mémoires DDR de type DDR2-667. Nous y reviendrons dans la partie de cours traitant des mémoires. Nous retrouvons aussi la gestion du SATA 2 c.-à-d. proposant des débits maximum théoriques pouvant atteindre 300Mo/sec (contre 150 Mo/sec pour les modèles actuels)



1.3 Le contrôleur d'interruptions.

1.3.1 Introduction.

Comme indiqué dans le paragraphe précédent, le microprocesseur dispose de une ou plusieurs bornes d'interruption. Ces bornes permettent à tout périphérique de demander un 'service' au microprocesseur: lorsque celui ci reçoit un tel signal, il termine son instruction en cours et se branche sur une partie de code appelée 'routine d'interruption' (ISR: Interrupt Service Routine) permettant de servir le périphérique demandeur.

On retrouve généralement plus de périphériques potentiellement demandeurs que d'entrées sur le microprocesseur. C'est la raison pour laquelle on retrouve sur les cartes mère contrôleur d'interruption permettant d'effectuer une gestion centralisée des demandes avant de les transmettre vers le microprocesseur. Cette gestion comprend notamment la gestion des priorités: si deux demandes arrivent en même temps sur le contrôleur d'interruption, laquelle fera l'objet de la première demande vers le microprocesseur?

1.3.2 Interruptions et architecture Intel.

Les interruptions dans l'architecture Intel ne sont pas exclusivement associées à des périphériques. La famille des microprocesseurs Intel x86 utilisent 256 interruptions, la plupart d'entre elles étant des interruptions logicielles (non abordées dans ce cours).

Les microprocesseurs Intel x86 ont une table des vecteurs d'interruptions située à l'adresse 0000:0000 qui comprend 1024 octets. Cette table comprend les adresses de toutes les routines de service d'interruption codées sur 4octets (32 bits). Cela donne donc un ensemble de 256 vecteurs d'interruption. En mode réel, la table des vecteurs d'interruption a la structure suivante :

Numéro (hex)	IRQ	Description
00-01	Exception	
02	NMI	Erreur de mémoire (parité pour les mémoires ECC)
03-07	Exception	

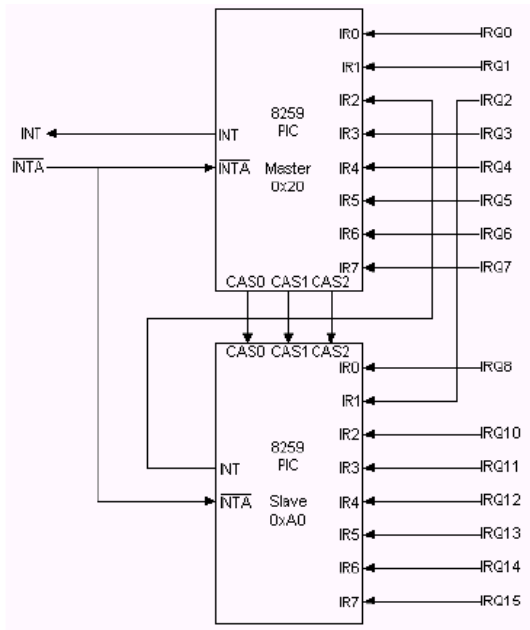
08	IRQ matérielle 0	Horloge (générée 18.2 x /seconde)
09	IRQ matérielle 1	Clavier
0A	IRQ matérielle 2	Mise en cascade du deuxième contrôleur
0B	IRQ matérielle 3	Port série COM2/COM4
0C	IRQ matérielle 4	Port série COM1/COM3
0D	IRQ matérielle 5	Libre
0E	IRQ matérielle 6	Contrôleur de disquette
0F	IRQ matérielle 7	Port parallèle.
10-6F	Interruptions logicielles	
70	IRQ matérielle 8	Horloge temps réel
71	IRQ matérielle 9	Redirection IRQ2
72	IRQ matérielle 10	Libre
73	IRQ matérielle 11	Libre
74	IRQ matérielle 12	Souris PS/2
75	IRQ matérielle 13	Coprocasseur mathématique
76	IRQ matérielle 14	Contrôleur IDE primaire
77	IRQ matérielle 15	Contrôleur IDE secondaire
78-FF	Interruptions logicielles	

Les ordinateurs de type XT (jusqu'aux 286) étaient équipés d'un seul contrôleur d'interruption permettant la gestion de seulement 7 interruptions matérielles. L'IRQ2 était déjà réservée pour la mise en cascade future d'un deuxième contrôleur. Celui-ci fut intégré sur les cartes mère des ordinateurs de type AT (Advanced Technologie).

Dans le tableau ci-dessus, nous retrouvons ce que l'on appelle des exceptions. Celles-ci sont appelées lorsque une erreur se produit: l'interruption logicielle 0 est générée lorsque une division par zéro se produit dans le microprocesseur provoquant son branchement dans la table des vecteurs à l'adresse 0000:0000. A cet emplacement, on retrouve l'adresse de la première ligne de code appartenant à la routine à exécuter.

1.3.3 Le PIC (programmable Interrupt Controller).

Les premiers ordinateurs de type PC XT étaient équipés du contrôleur d'interruption programmable 8259 de chez Intel. Ce composant permettait la gestion de huit niveaux d'interruption tout en offrant une possibilité de mise en cascade permettant la gestion de 64 niveaux: on retrouvait alors un contrôleur maître et sur chacune de ses entrées était connecté la sortie d'un contrôleur esclave. Ce contrôleur permettait la sélection d'un mode de priorité permettant à tout moment une reconfiguration dynamique adaptée aux besoins du système.



Le composant prévoit deux modes de gestion des priorités:

- 1 La rotation automatique: dans ce mode, lorsque un périphérique a été servi, il reçoit la priorité la plus basse de façon à ce que, dans le pire des cas, un périphérique demandeur n'ait qu'à attendre que les sept autres périphériques aient été servis au plus une seule fois.
- 2 La rotation spécifique: dans ce mode, le programmeur peut changer l'entrée affectée à la priorité la plus basse, les autres priorités étant fixées en fonction de cette dernière.

Dans les PC, c'est la deuxième solution qui a été retenue. Ce mode est programmé lors de l'initialisation de votre carte mère par le BIOS. C'est l'entrée IRQ7 qui reçoit la priorité la plus basse et IRQ0 la priorité la plus élevée. Le fait que le second contrôleur soit mis en cascade sur l'entrée IRQ2 du premier contrôleur, nous aurons dans l'ordre décroissant de priorité les entrées suivantes: IRQ0, IRQ1, IRQ8, IRQ9, IRQ10, IRQ11, IRQ12, IRQ13, IRQ14, IRQ15, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7.

Le PIC dispose de registres internes permettant la gestion des interruptions. Nous retiendrons uniquement les registres suivants:

IRR: (Interrupt request Register)

IMR: (Interrupt Mask Register)

ISR: (In Service Register)

Les huit lignes d'entrée passent au travers du registre de masquage qui va déterminer si une entrée est masquée ou pas. Si elle est masquée, la demande arrivant sur cette ligne ne sera pas prise en compte. Si elle n'est pas masquée, la demande sera enregistrée dans le registre de demande d'interruption qui la maintiendra jusqu'au moment où elle sera servie. En fonction des priorités programmées, le PIC pourra déterminer de quelle interruption il doit s'occuper. Il envoie donc une demande INT vers le processeur en activant la borne INT de ce dernier. Le microprocesseur termine son instruction en cours et va renseigner au PIC qu'il accepte la demande en activant sa borne INTA (Interrupt Acknowledge). Dès la réception de ce signal, le PIC va positionner le bit correspondant à l'interruption servie dans le registre ISR à '1' et le bit correspondant dans le registre IRR à '0'.

Le microprocesseur va alors envoyer un deuxième signal INTA indiquant au PIC qu'il doit déposer sur le bus de données le numéro de l'interruption logicielle qui doit être servie. Ce numéro est construit de la façon suivante par le PIC: les trois bits de poids faible correspondent au numéro de l'interruption matérielle tandis que les cinq bits de poids fort correspondent à une valeur préprogrammée lors de l'initialisation du PIC. Pour les PC, le PIC maître reçoit la valeur 00001 et le PIC esclave la valeur

01110. Si l'on prend l'IRQ matérielle 3 (PIC maître IRQ 3), cela correspondra donc à 00001011 (0B) tandis que si l'on prend l'IRQ matérielle 10 (PIC esclave IRQ 2), cela correspondra à 01110010 (72). La routine d'interruption doit prévoir l'envoi d'un signal (EOI) vers le PIC qui replacera le bit correspondant à l'interruption servie à 0.

Analysons maintenant la gestion des priorités:

Partons d'une condition initiale où l'interruption matérielle 4 est servie. Nous retrouverons alors les données suivantes dans les différents registres:

IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
0	0	0	0	0	0	0	0

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	1	0	0	0	0

Supposons que deux demandes d'interruptions se présentent sur les entrées IR6 et IR2 du PIC. L'IRQ6 étant moins prioritaire que l'interruption 4 en cours de traitement, la demande restera en attente. L'IRQ2 étant plus prioritaire que l'interruption 4 en cours de traitement, elle sera prise en compte par le PIC.

Celui ci enverra donc une demande INT vers le microprocesseur qui répondra par la succession de deux signaux INTA. Les registres contiendront alors les données suivantes:

IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
0	1	0	0	0	0	0	0

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	1	0	1	0	0

Il est important de signaler que la routine de l'IRQ4 en cours d'exécution a été interrompue au bénéfice de la routine de l'IRQ2. Lorsque cette routine se termine, elle envoie un EOI (End Of Interrupt) vers le PIC. Celui ci, en analysant son registre ISS sait qu'il s'agit d'un signal de fin d'interruption pour l'IRQ2. Voici le contenu des registres:

IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
0	1	0	0	0	0	0	0

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	1	0	0	0	0

Le processeur ayant terminé l'exécution de la routine de l'IRQ2, il revient à l'exécution du code avant interruption: il s'agissait de la routine de l'IRQ4. Lorsque cette routine se termine, elle envoie un EOI (End Of Interrupt) vers le PIC. Celui ci, en analysant son registre ISS sait qu'il s'agit d'un signal de fin d'interruption pour l'IRQ4. Voici le contenu des registres:

IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
0	1	0	0	0	0	0	0

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	0	0	0	0	0

Le PIC peut alors prendre en compte la demande de l'IRQ6 puisqu'il n'y a plus aucune autre demande plus prioritaire en cours de traitement.

Redirection IRQ2/IRQ9: malgré le fait que cette interruption ait été réservée dès les premiers PC XT pour une mise en cascade d'un deuxième contrôleur, certains constructeurs ont utilisé cette interruption pour une toute autre fonction en l'utilisant pour certains périphériques. Lorsque les PC AT sont apparus sur le marché, il a fallu jouer d'artifice pour maintenir une compatibilité avec de tels périphériques. C'est l'IRQ9 qui fut alors utilisée pour remplacer l'IRQ2 mais le problème était que l'IRQ9 était présente sur le contrôleur esclave alors que l'IRQ2 se trouvait initialement sur le contrôleur maître. Le problème fut réglé de la façon suivante: au niveau de la routine associée à l'IRQ9, une EOI est envoyé vers le PIC esclave et ensuite un ISR est appelé pour l'IRQ2. Le code assembleur associé à l'IRQ9 serait alors le suivant:

```
MOV AL,20
OUT A0,AL ;envoi d'un EOI vers le PIC esclave
INT 0A ;appel de l'interruption correspondant à l'IRQ2
IRET
```

1.3.4 Partage des interruptions.

L'ajout d'un contrôleur secondaire dans les PC n'aura malheureusement pas assouvi le besoin d'entrées supplémentaires pour la gestion des périphériques demandeurs de plus en plus nombreux (carte vidéo, carte réseau, carte son, contrôleur USB, carte d'acquisition, carte SCSI...)

Pour y faire face, Windows a proposé une gestion logicielle: si plusieurs périphériques se partagent la même interruption, il est possible par logiciel de passer en revue leur registre d'état pour déterminer quel est le périphérique demandeur.

L'apparition du bus d'extension PCI a donné un peu de répit car il permet de façon matérielle une possibilité de partage des interruptions. Cette solution sera décrite dans le paragraphe traitant des bus d'extension.

1.3.5 L'IO APIC (IO Advanced Programmable Interrupt Controller)

Tandis que le contrôleur d'interruption standard est conçu pour une utilisation dans des systèmes mono processeur, le contrôleur IOAPIC peut être utilisé indifféremment dans les systèmes mono et multi processeurs. Pour une question de compatibilité, les deux contrôleurs se retrouvent sur la carte mère, les interruptions pouvant être contrôlées par le PIC standard ou par l'IOAPIC dépendant de la façon dont les contrôleurs ont été programmés. Ce sera de la responsabilité du programmeur que le même signal d'interruption ne soit pas pris en charge par les deux contrôleurs. Cette possibilité nous laisse entrevoir l'installation d'anciens systèmes d'exploitation tels que Windows98 sur des cartes mères récentes qui ne gèrera que le PIC tandis que des systèmes d'exploitation tels que Windows 2000 ou XP prendront en charge sur la même carte mère le contrôleur avancé.

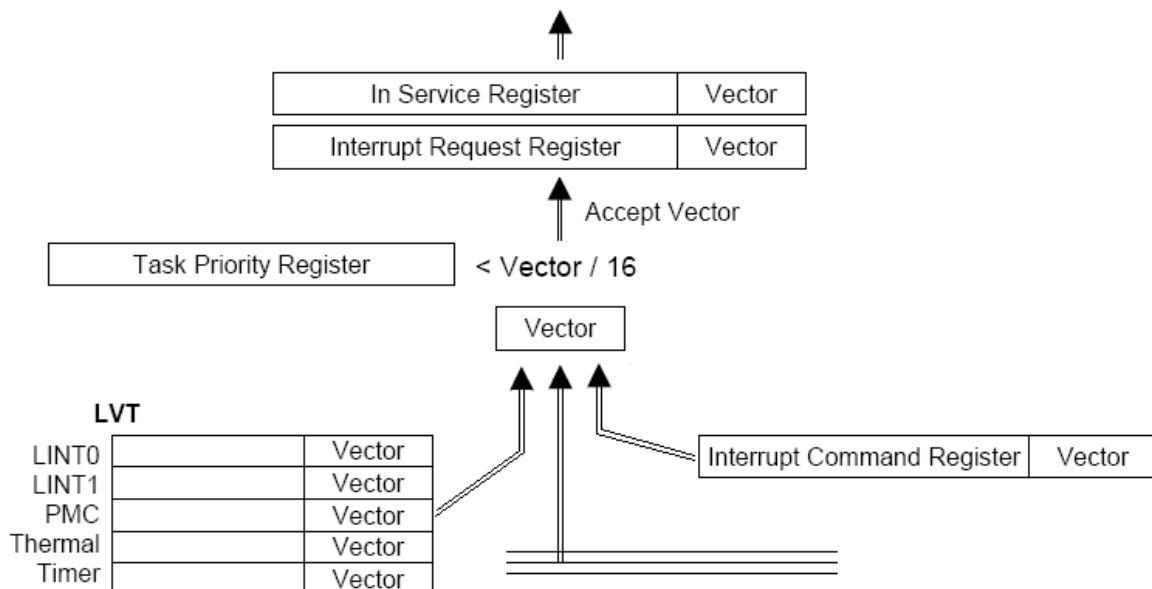
Au niveau système, l'APIC est constitué de deux parties : l'une résidant dans le microprocesseur et appelée Local APIC tandis que l'autre se trouve dans le sous système IO et est appelée IOAPIC. Ces deux parties communiquent à travers un bus dédié appelé bus APIC.

Bien que le microprocesseur soit capable de pouvoir gérer 256 sources d'interruption, la partie IO ne prévoit au niveau matériel que la gestion de 24 interruptions matérielles. A chacune de ces 24 entrées est associé un registre de 64 bits permettant de spécifier la polarité du signal (actif à l'état haut ou à l'état bas), la sensibilité du signal sur un flanc ou sur un niveau, la destination et le mode dans lequel l'interruption sera délivrée. Ces registres se trouvent dans une table de redirection et sont utilisés pour translater l'interruption correspondante dans un message inter APIC qui sera délivré sur le bus APIC. Bien qu'au niveau matériel on ne retrouve que 24 interruptions matérielles, l'APIC peut gérer jusque 240 vecteurs d'interruptions différents allant du vecteur 16 jusqu'au vecteur 255. Si un vecteur d'interruption était envoyé ou reçu au travers de l'APIC local, celui-ci indiquerait un statut de vecteur illégal au travers de son registre de statut d'erreur. Bien que les vecteurs d'interruptions entre 16 et 31 soient également utilisés pour la gestion des erreurs internes du micro-processeur, l'APIC accepte quand même de tels vecteurs.

Les priorités: pour les interruptions qui sont délivrées au processeur à travers l'APIC local, chaque interruption a une priorité liée à son numéro de vecteur. L'APIC local utilise cette priorité pour déterminer quand servir cette demande en rapport avec les autres activités du processeur incluant notamment le service des autres interruptions. Pour les vecteurs d'interruption dans la gamme comprise entre 16 et 255, la priorité de l'interruption est déterminée en utilisant la relation suivante : $\text{Priorité} = \text{vecteur} / 16$. Le quotient est arrondi à la valeur entière avec la valeur 0 comme priorité la plus basse et 15 la plus élevée. Du fait que les vecteurs 0 à 15 sont interdits et que les vecteurs de 16 à 31 sont réservés pour les exceptions, les priorités des interruptions définies par l'utilisateur s'étendent de 2 à 15.

Dans chacun de ces 16 groupes, les quatre bits de poids faible du vecteur seront utilisés pour classer la priorité des interruptions dans un même groupe.

La priorité de tâche: c'est une valeur comprise entre 0 et 15 présente dans le registre de priorité de tâche (TPR) de l'APIC. Cette priorité est configurable par logiciel et permet de définir un seuil de priorité pour les interruptions du processeur. Le processeur servira seulement les interruptions qui ont une priorité plus élevée que celle spécifiée dans le registre TPR : la valeur 0 permettra d'autoriser toutes les interruptions tandis que la valeur 15 les interdira toutes, exceptées les interruptions de type NMI, SMI, INIT, ExtINT.



L'APIC local peut recevoir les interruptions des sources suivantes :

- Périphériques I/O connectés localement aux bornes d'interruptions du processeur (LINT0 et LINT 1). Les périphériques peuvent être connectés à un contrôleur de type 8259 qui est à son tour connecté à une de ses bornes locales.
- Périphériques I/O connectés aux bornes d'interruption d'un APIC de type I/O, les interruptions étant envoyées sur le bus APIC au moyen de messages d'interruption I/O.
- Interruptions inter processeurs. Un processeur peut utiliser le mécanisme IPI pour interrompre un autre processeur ou un autre groupe de processeur sur le bus système. Les IPI's peuvent être utilisés pour de l'auto interruption logicielle, du transfert d'interruptions ou de la planification préemptive.
- Les interruptions générées par une horloge APIC.
- Les interruptions liées à une erreur interne de l'APIC.
- Les interruptions liées à un capteur de température

- Les interruptions liées à un compteur de performance (PMC : Performance monitoring counter)

L'IOAPIC et le multi processeurs. Lorsqu'une interruption arrive sur une des bornes de l'IOAPIC, celui-ci peut transmettre cette demande par le bus APIC vers un processeur spécifique, vers un groupe de processeur ou vers l'ensemble de tous les processeurs. Cette possibilité de configuration est possible grâce aux 64 bits de chaque registre associé à chacune des entrées.

Nous retrouverons pour l'IOAPIC, le champ de destination et le mode de destination. Si le mode de destination est le mode physique, alors le champ de destination contiendra l'identificateur APIC (4 bits) qui permettra de sélectionner un seul composant sur le bus APIC. Si le mode de destination est le mode logique, alors le champ de destination (8 bits) définira un jeu de processeurs, chaque APIC local des processeurs utilisant ses registres DFR (Destination Format Register) et LDR (Logical Destination Register) pour déterminer si il fait partie ou pas du jeu.



Dans le modèle Flat, chaque APIC local effectue un ET logique entre l'identificateur logique APIC et l'adresse de destination du message envoyé par l'IOAPIC. Si une condition logique est vraie, alors l'APIC local va accepter le message. Un broadcast vers tous les APIC locaux peut être effectué en positionnant tous les bits de l'adresse de destination à 1.

Dans le modèle Flat Cluster, l'identificateur logique et l'adresse de destination sont décomposés en deux groupes : les quatre bits de poids fort permettant de coder le numéro du cluster et les quatre bits de poids faible permettant de représenter quatre APIC locaux dans le même cluster, un par bit.

1.4 Le contrôleur de DMA.

1.4.1 Introduction.

Sur une carte mère, bon nombre de périphériques doivent transférer des données que le microprocesseur doit ensuite traiter: prenons à titre d'exemple le disque dur, le lecteur de disquette ou la carte réseau. Ces données, avant d'être traitées, sont placées en mémoire via une des méthodes suivantes: I/O programmé (PIO) ou accès direct à la mémoire (DMA).

1.4.2 Entrées/sorties programmées.

Les entrées/sorties programmées se réfèrent à l'utilisation d'instructions d'entrées/sorties qui seront exécutées par le microprocesseur pour effectuer le transfert des données entre la mémoire et les registres du périphérique. L'avantage de cette méthode est la simplicité de sa mise en œuvre: le microprocesseur est suffisamment rapide pour assurer un transfert aussi rapide que la fourniture des données par le périphérique et le seul matériel additionnel nécessaire sur la carte mère est bien souvent un circuit de Wait State pour ralentir le processeur lors de ses accès ou pour le synchroniser avec le périphérique. Le désavantage de ce mode est que le microprocesseur est occupé pendant toute la durée du transfert pour n'effectuer qu'une tâche relativement simple.

1.4.3 L'accès direct à la mémoire.

On peut aisément comprendre qu'il serait intéressant que le microprocesseur puisse effectuer d'autres tâches alors que d'une façon indépendante le périphérique transférerait ses données vers la mémoire. Pour effectuer un tel transfert, le périphérique doit utiliser les bus du microprocesseur. Le contrôle du bus est assuré pour les périphériques par un circuit supplémentaire que l'on appelle contrôleur de DMA ou encore DMAC. Ce circuit peut effectuer toutes les opérations nécessaires pour le transfert des données (incrémenter de l'adresse mémoire, décrémenter du compteur de transfert, lecture sur le périphérique (INPUT), écriture en mémoire) en un seul cycle de bus. Donc le bus n'est seulement occupé que pendant un cycle pour le transfert d'un octet et le microprocesseur peut continuer à exécuter ses codes.

Le contrôleur de DMA peut travailler suivant deux modes:

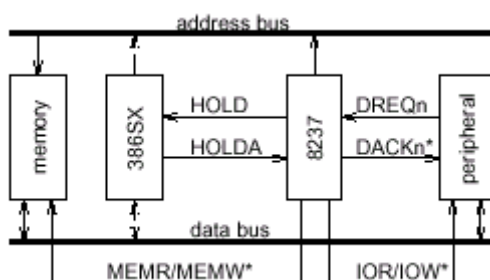
- mode par vol de cycle dans lequel le contrôleur prend le contrôle des bus pour chaque octet transféré et ensuite rend le contrôle des bus au microprocesseur.
- mode en rafale dans lequel des blocs de données sont transférés avant de rendre le contrôle des bus au microprocesseur.

Le choix dépend de la rapidité avec laquelle les données sont fournies par le périphérique en regard de la bande passante du bus et du fait qu'une application particulière puisse accepter que le microprocesseur soit déconnecté de ses bus pendant la durée du transfert.

Les transferts peuvent être effectués en une étape ou en deux étapes. Dans un transfert en deux étapes, la première étape consiste en la lecture des données sur un port d'un périphérique lors d'un cycle et la deuxième étape en l'écriture de la donnée vers la mémoire lors d'un deuxième cycle. Il est aussi possible pour le contrôleur d'effectuer cette opération simultanément. C'est le mode de fonctionnement utilisé sur les PC.

1.4.4 DMA sur les PC XT.

Le premier contrôleur utilisé sur les PC XT était un circuit de chez Intel: le 8237. Il fournissait quatre canaux DMA, chacun des canaux pouvant être programmé et caractérisé par une adresse indépendante, un registre de comptage et des lignes de contrôle de périphérique. Le circuit prévoit des priorités sur les entrées de sorte qu'un seul transfert puisse être actif à la fois.



Nous allons nous baser sur le schéma précédent pour décrire les différentes phases caractérisant un transfert DMA.

- Chaque fois qu'un périphérique doit effectuer un transfert, il va activer la borne DREQn (n représentant le numéro du canal utilisé) du contrôleur DMA.
- Dès réception de cette demande, le contrôleur va activer la borne HOLD du microprocesseur pour lui indiquer qu'il désire utiliser ses bus.
- Lorsque le microprocesseur reçoit un tel signal, il termine son instruction en cours et active sa sortie HOLDA pour indiquer au contrôleur de DMA qu'il va placer ses bus dans un état haute impédance (voir notion tri-states pour les sorties des circuits logiques).

- Le contrôleur va alors déposer l'adresse mémoire sur le bus d'adresse, activer MEMR (Memory Read) et IOW (Input/Output Write) ou MEMW (Memory Write) et IOR (Input/Output Read) et activer le signal d'acquiescement de DMA vers le périphérique.
- Le périphérique répond alors au signal DACKn en lisant ou en écrivant la donnée sur le bus de données.
- La borne HOLD est maintenue active vers le microprocesseur tant que le transfert n'est pas terminé. Dès le relâchement du signal, le microprocesseur reprend son travail jusqu'à la requête suivante.

1.4.5 DMA sur les PC AT.

L'arrivée des PC AT (les premiers furent équipés d'un microprocesseur 80286 – 16bits) fut marquée par l'apparition d'un deuxième contrôleur de DMA mis en cascade sur le premier. Alors que dans les PC XT, les slots d'extensions ISA ne permettaient que des transferts de données de 8 bits, les pc AT furent équipés d'un petit connecteur additionnel permettant de porter les slots d'extensions à 16 bits. Il est clair que les contrôleurs DMA évoluèrent: le premier permettait toujours des transferts 8 bits pour une question de compatibilité tout en acceptant les transferts sur 16 bits alors que le deuxième contrôleur ne permettait que des transferts sur 16 bits.

Toujours pour une question de compatibilité, la vitesse maximale avec laquelle ce type de contrôleur pouvait travailler était liée à la cadence d'horloge du premier PC XT. Voici comment s'organisait l'utilisation des ressources des deux contrôleurs:

Canal DMA	Bus	Usage par défaut	Autres usages communs
0	-	Rafraîchissement de la mémoire	-
1	8/16 bits	Carte son	Adaptateur SCSI, port parallèle ECP, cartes réseau, modems vocaux
2	8/16 bits	Lecteur de disquette	Cartes accélératrices pour lecteur de bandes
3	8/16 bits		Contrôleur de disque dur pour les PC XT (contrôleur ST506), Adaptateur SCSI, port parallèle ECP, cartes réseau, modems vocaux, cartes son
4	-	Cascade du deuxième contrôleur	-
5	16 bits	Carte son	Adaptateurs SCSI, cartes réseau.
6	16 bits	-	Cartes son, cartes réseau.
7	16 bits	-	Cartes son, cartes réseau.

Les PC AT ont également vu l'apparition d'un nouveau contrôleur de disque dur. Précédemment, le contrôleur se trouvait sur la carte mère ou sur une carte d'extension: on parlait de contrôleur ST506. Pour les PC AT, le contrôleur se trouve sur le disque dur et on voit le câble qui le relie à la carte mère comme étant une extension du bus système: on parle alors de bus AT ou d'IDE (Integrated Drive Electronics).

Alors que les vitesses de rotation des disques ne faisant qu'augmenter et du fait la vitesse de transfert, l'utilisation d'un canal de DMA bridé pour une question de compatibilité ne fait plus l'affaire. Les nouveaux ordinateurs ont des disques dont les transferts sont assurés en utilisant un mode PIO ou un transfert DMA appelé 'first-party' sur le bus PCI. On parle alors de bus mastering. Ce mode de transfert offre de meilleures performances du fait que les périphériques modernes possèdent des circuits DMA intégrés plus rapides que le contrôleur DMA/ISA standard.

1.4.6 Le busmastering ou DMA 'first-party'.

Etant donné que ce mode de transfert est lié fortement aux caractéristiques des disques durs, nous allons reprendre dans un tableau l'ensemble des différentes normes avec la vitesse de transfert correspondante et un comparatif avec le mode PIO correspondant.

PIO mode 0	ATA (ATA-1 ou IDE)	3.33 MB/sec	1994
DMA Mode 0 ou DMA Multiword Mode 0	ATA	2.1-4.16 MB/sec	
PIO Mode 1	ATA	5.22 MB/sec	
PIO Mode 2	ATA	8.33 MB/sec	
PIO Mode 3	ATA-2 (EIDE ou FASTATA)	11.1 MB/sec	1996
DMA Mode 1 ou DMA multiword Mode 1	ATA-2	4.2-13.3 MB/sec	
PIO Mode 4	ATA-2	16.6 MB/sec	
DMA Mode 2 ou DMA multiword Mode 2	ATA-2	8.3-16.6 MB/sec	
Ultra DMA Mode 0	ATA/ATAPI 4	16.6 MB/sec	1998
Ultra DMA Mode 1	ATA/ATAPI 4	25.0 MB/sec	
Ultra DMA Mode 2, UDMA 33 ou ATA/33	ATA/ATAPI 4	33.3 MB/sec	1999
Ultra DMA Mode 3	ATA/ATAPI 5	44.4 MB/sec	2000
Ultra DMA Mode 4, UDMA 66 ou ATA/66	ATA/ATAPI 5	66.6 MB/sec	2000
Ultra DMA Mode 5, UDMA 100 ou ATA/100	ATA/ATAPI 6	100 MB/sec	2000
Ultra DMA Mode 6, UDMA 133 Ou ATA/133	ATA/ATAPI 7	133 MB/sec	2001

L'augmentation du débit des différentes normes s'explique par une diminution du temps de cycle nécessaire pour effectuer un transfert. L'utilisation des deux flancs d'horloge pour effectuer un transfert permet également d'augmenter le débit. C'est ce qui explique que l'on ait pu passer à la norme UDMA 33 présentant un débit de 33.3 MB/sec alors que l'horloge présentait un temps de cycle identique à celle utilisée pour la norme DMA Mode 0 (multiword) qui était limitée à un débit de 16.6 MB/sec. Le passage aux normes UDMA 66 et UDMA 100 s'est effectué en augmentant la cadence d'horloge ce qui a d'ailleurs justifié le passage d'un câble standard IDE à 40 conducteurs à un câble IDE à 80 conducteurs. L'UDMA 100 présente un temps de cycle de 40 ns tandis que l'UDMA 33 de 120 nsec.

Attention: un cycle de transfert n'est pas nécessairement égal à un cycle d'horloge. Il faut en effet parfois plusieurs cycles d'horloge pour effectuer un transfert complet (le temps nécessaire pour ce transfert est également appelé cycle).

2 Classification des mémoires.

2.1 Introduction.

Les mémoires peuvent être classées en fonction de la volatilité des données qu'elles contiennent et du fait qu'elles nécessitent ou pas un signal d'horloge pour assurer la synchronisation des transferts. On parlera dans ce dernier point de mémoires asynchrones ou de mémoires synchrones. La volatilité dépendra du fait que les mémoires puissent ou non conserver les données même si la tension d'alimentation disparaît. On parlera de mémoires vives et de mémoires mortes.

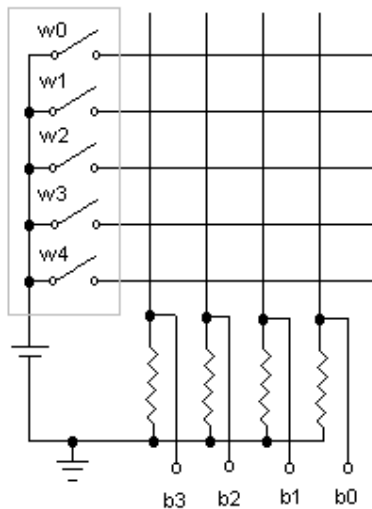
2.2 Les mémoires asynchrones.

2.2.1 Les mémoires mortes.

a) Les mémoires ROM.

ROM signifie Read Only Memory c-à-d mémoire en lecture seule. Ces mémoires sont dites non volatiles c-à-d que l'absence de tension d'alimentation ne provoque pas la perte des données comprises dans ces mémoires. Elles ne sont pas programmable directement par l'utilisateur final mais les données ou codes doivent être y placés lors de la fabrication du composant.

Prenons le schéma de principe suivant pour comprendre le fonctionnement d'une telle mémoire:

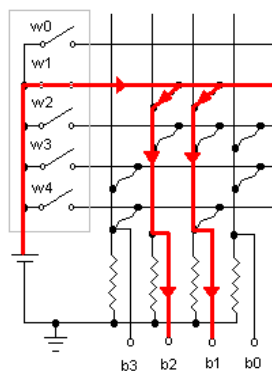


La zone encadrée d'un trait gris correspond au décodeur d'adresses. Lorsqu'une adresse sera fournie à la mémoire pour en récupérer une donnée, le décodeur interne fermera l'interrupteur correspondant à cette adresse.

Ce schéma correspond à une mémoire de 5x4bits. Imaginons que l'on veuille obtenir la table de vérité suivante:

w4	w3	w2	w1	w0	b3	b2	b1	b0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	1	1	0
0	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	0	1
1	0	0	0	0	1	0	1	0

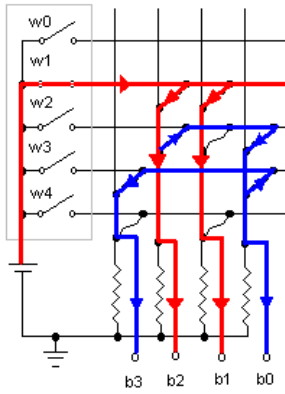
Sachant que l'on va associer une tension nulle à un bit 0 et une tension 5 volt à un bit 1, on peut modifier le schéma précédent pour le faire correspondre à notre table de vérité.



Si nous supposons l'interrupteur w1 fermé et en ne considérant que cette seule ligne où w1 se situe, nous retrouveront bien une tension nulle sur b3 et sur b0 ainsi que la tension de 5 volts sur b2 et sur b1.

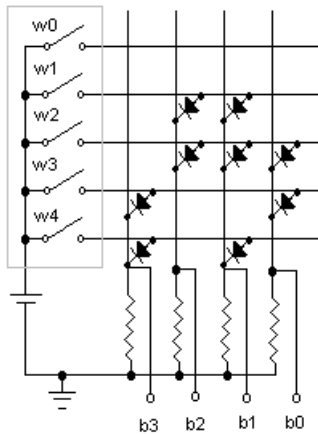
Mais si nous analysons attentivement notre schéma, nous pouvons nous rendre compte que certains autres liens placés sur d'autres lignes vont interférer et provoquer une erreur dans la donnée qui sera fournie par cette mémoire.

Le résultat pour w1 sera donc 1111. Le schéma suivant reprend certaines de ces connexions parasites.

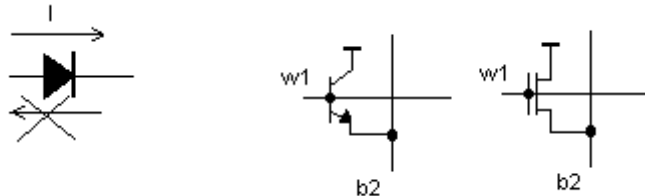


Nous pouvons remarquer dans notre schéma que les liens parasites qui sont à l'origine du problème ont un sens de courant qui parcourt le lien différent de ceux parcourant les liens qui assurent la fourniture correcte de la tension correspondant à l'état du bit souhaité.

On peut remplacer alors ces liens par un composant électronique autorisant le passage du courant dans un sens et pas dans l'autre. Ce composant est la diode dont voici le symbole:



Nous pouvons également retrouver d'autres semi-conducteurs à la place des diodes. Nous pouvons placer des transistors bipolaires ou des transistors MOSFET. Voici alors la version simplifiée d'une seule cellule:



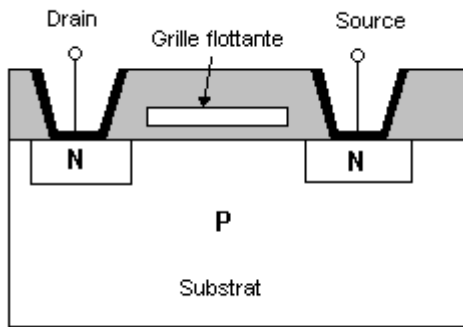
L'ensemble de ces semi-conducteurs est placé lors de la fabrication de la mémoire. Celle-ci peut donc être lue mais ne peut être modifiée par la suite.

b) Les mémoires PROM.

Ces mémoires sont également des mémoires mortes (non volatiles) dont le contenu ne s'efface pas si la tension d'alimentation disparaît. Ces mémoires peuvent être lues mais ne sont programmables qu'une seule fois par l'utilisateur. Le principe est simple: lors de la construction de ces mémoires, on va utiliser le principe des mémoires ROM avec diodes excepté que ces diodes sont placées à toutes les intersections. La lecture d'une telle mémoire non programmée nous fera apparaître tous les bits à l'état logique 1. La programmation consiste donc à détruire les diodes là où l'on désire qu'un 0 logique soit placé dans la cellule mémoire correspondante. Ces mémoires portent également le nom de OTP pour l'abréviation de On Time Programmable.

c) Les mémoires EPROM.

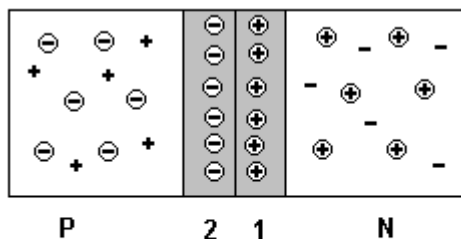
Ces mémoires sont également des mémoires mortes mais elles sont effaçables et reprogrammables. Pour en comprendre la technique, nous allons analyser l'élément constitutif principal d'une cellule mémoire élémentaire qui est le transistor FAMOS (Floating Gate Avalanche Metal Oxyde Silicium). Le schéma de principe d'un transistor FAMOS est le suivant:



Dans une zone N, on retrouve des donneurs d'ion (charges fixes) et des électrons (porteurs majoritaires mobiles) tandis que dans une zone P, on retrouve des accepteurs d'ion et des trous (porteurs majoritaires mobiles).

Pour qu'un courant puisse circuler entre le drain et la source, il faut qu'il n'y ait entre ces deux zones que des porteurs majoritaires mobiles de même type. Nous allons décrire le procédé par lequel nous pouvons faire en sorte qu'un courant puisse circuler entre le drain et la source.

La jonction (frontière entre les deux zones) Drain/Substrat va être polarisée en sens bloquant de sorte de pouvoir provoquer un effet d'avalanche. Pour bien comprendre cet effet d'avalanche, nous reprendrons la théorie de la diode en nous basant pour une bonne compréhension du phénomène sur le principe simple que des charges de même signe se repoussent tandis que des charges de signes contraires s'attirent. Reprenons le schéma de la diode suivant:

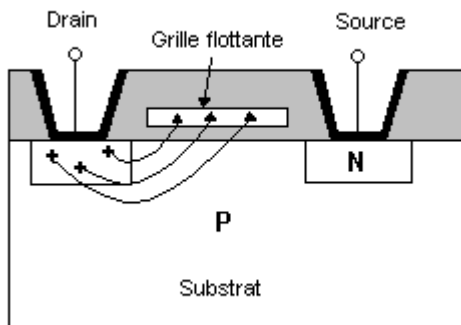


- ⊖ Charges fixes négatives
- ⊕ Charges fixes positives
- + Porteurs mobiles positifs
- Porteurs mobiles négatifs

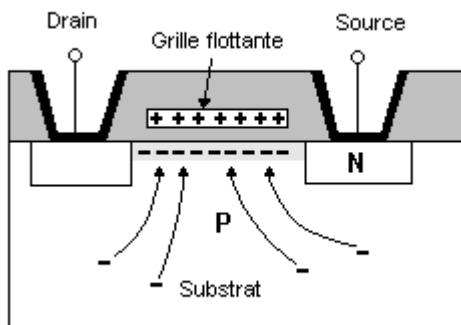
On va retrouver également dans chacune des zones des porteurs mobiles minoritaires qui seront de signe + pour la zone N et de signe - pour la zone P. On appelle ces porteurs des porteurs minoritaires car ils sont en nombre plus réduits que les autres; ils résultent principalement de la création de paires électrons/trous dans le réseau cristallin dû à l'agitation thermique.

Dans la zone N, un électron ne peut approcher de la jonction car même si elle parvenait à pénétrer dans la zone 1, il serait automatiquement repoussé par les charges négatives présents dans la zone 2. Par contre, si un porteur minoritaire positif dans cette même zone parvenait du fait de son énergie, à entrer légèrement dans la zone 1, ce porteur serait attiré par les charges de la zone 2 et ensuite rejetées dans la zone P avec une certaine énergie. Plus la zone 1 et 2 sont larges, plus l'énergie qu'aura le porteur sera importante et son entrée dans la zone N provoquera dans le réseau cristallin la création de paires électron/trou avec augmentation de la température. Dans la zone P, ces porteurs minoritaires ainsi créés pourraient à leur tour suffisamment d'énergie pour s'approcher de la zone 2, être happés et rejetés dans la zone N... C'est ce que l'on appelle l'effet d'avalanche.

Dans le cas de notre transistor FAMOS, les électrons pourraient avoir suffisamment d'énergie que pour pouvoir franchir la couche très mince de SiO₂ et se retrouvés emprisonnés dans la grille flottante.



Les charges positives prisonnières dans la grille flottante vont pouvoir attirer les porteurs minoritaires présents dans le substrat et l'on va retrouver alors dans la zone P un endroit où la concentration en porteurs positifs sera plus grande que la concentration en porteurs négatifs. On parlera alors de la création d'un canal d'inversion.



Comme on retrouve alors le même type de porteurs mobiles entre le drain et la source, on pourra alors avoir un courant et le transistor est alors considéré comme passant.

Pour rendre le transistor à nouveau bloqué, il suffit de donner aux charges présentes dans la grille flottante suffisamment d'énergie que pour retourner dans le substrat. Cette énergie sera donnée en exposant le transistor à une lumière dans la gamme de l'ultra violet. C'est la raison pour laquelle on retrouve dans les mémoires de type EPROM une fenêtre en quartz présente sur le dessus du boîtier. Cette fenêtre doit être occultée lorsque la mémoire a été programmée pour ainsi éviter tout effacement accidentel.

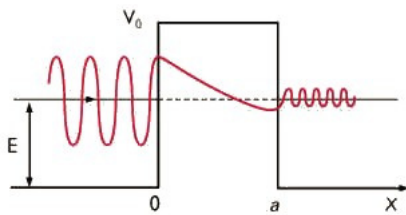
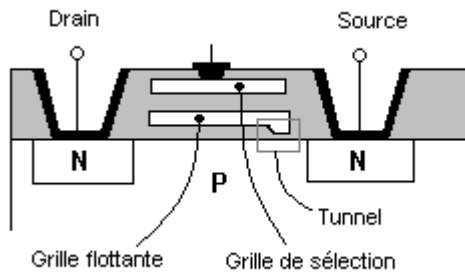
Il faut compter plusieurs minutes pour effacer une EPROM tandis que sa programmation nécessite une tension de l'ordre de 15 à 20 volts pour l'effet d'avalanche et il faut compter quelques μ sec par mots pour la programmation. Il est important de signaler que le nombre de cycles (effacements / reprogrammations) est limité et avoisine les 1000 unités.

d) Les mémoires EEPROM ou E²PROM

Un des aspects négatifs des mémoires EPROM est le fait que pour en effacer leur contenu, il faut les extraire de leur support en vue de les placer dans un effaceur d'EPROM constitué d'une lampe à ultra violet. La programmation nécessite également l'utilisation d'un niveau de tension que l'on ne retrouve pas sur les cartes mères d'ordinateur, et de ce fait, il faut donc disposer d'un programmeur d'EPROM externe pour cette opération.

L'idée est donc de pouvoir diminuer le niveau de tension nécessaire à la programmation et permettre une autre technique d'effacement.

EEPROM signifie "*Electrically Erasable Programmable ROM*". L'effacement d'une telle mémoire est électrique. Ces mémoires reposent sur l'utilisation d'un transistor FLOTOX. On y retrouve une grille flottante, une grille de sélection et le principe de l'effacement repose sur un effet tunnel que l'on appelle (Fowler-Nordheim tunneling).



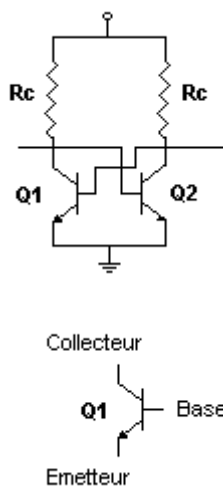
Si l'on considère un électron non plus en tant que particule mais en tant qu'onde, on peut expliquer que même si cet électron n'a pas l'énergie suffisante que pour franchir une barrière de potentiel, il pourra la traverser si cette barrière est mince. Le principe est donné par le schéma suivant:

2.2.2 Les mémoires vives (mémoires volatiles).

1 Les mémoires statiques (SRAM).

SRAM signifie Static Random Access Memory c-à-d mémoire à accès aléatoire statique. Ces mémoires sont dites volatiles c-à-d que l'absence de tension d'alimentation provoque la perte des données comprises dans ces mémoires. Ces mémoires sont accessibles en lecture/écriture.

Voici le schéma de principe d'une cellule élémentaire d'une telle mémoire.



Ce schéma repose sur l'utilisation de transistors bipolaires. Ces transistors peuvent être vus comme des interrupteurs statiques entre leurs bornes d'émetteur et de collecteur. La borne de base permet de commander cet interrupteur en courant. Si aucun courant ne circule entre la base et l'émetteur, l'interrupteur est considéré comme ouvert, c-à-d qu'aucun courant ne peut circuler entre le collecteur et l'émetteur. Si un courant circule entre la base et l'émetteur, l'interrupteur est considéré comme fermé et un courant peut alors circuler entre le collecteur et l'émetteur du transistor.

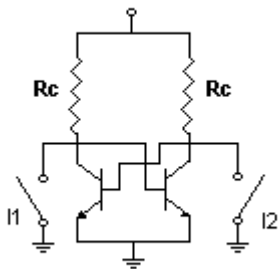
Analysons le fonctionnement d'un tel schéma. Supposons initialement que le transistor Q1 soit bloqué. Cela signifie qu'un courant peut alors circuler entre la base et l'émetteur du transistor Q2 et de ce fait le

rendre passant. Si le transistor Q2 est passant, le potentiel de l'émetteur de Q2 est tiré à la masse et de ce fait aucun courant ne peut circuler dans la base de Q1, ce qui le maintient bloqué. C'est donc un état de fonctionnement stable.

Vue la symétrie verticale du schéma, on peut en conclure que si nous avions émis l'hypothèse de départ que Q2 soit bloqué on aurait eu comme situation finale Q1 passant et Q2 reste bloqué avec à ce stade un nouvel état stable.

Nous allons pouvoir associer à chaque état stable une valeur logique. Le tout est maintenant de savoir comment nous allons pouvoir passer d'un état stable à l'autre pour pouvoir ainsi changer le contenu de la cellule mémoire.

Reprenons le même schéma et ajoutons y des interrupteurs montés en parallèle sur chaque transistor.

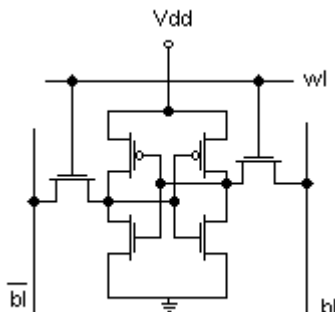


Prenons la situation de départ suivante: Q1 est passant, Q2 est bloqué et les deux interrupteurs I1 et I2 sont ouverts.

Fermons alors l'interrupteur placé en parallèle sur le transistor bloqué. Le potentiel du collecteur de Q2 est tiré à la masse par cet interrupteur et aucun courant ne peut plus circuler dans la base de Q1 qui se bloque. Si Q1 se bloque, un courant peut alors circuler dans la base de Q2 qui devient alors passant ce qui nous autorise alors à rouvrir l'interrupteur I2 et le montage se retrouve alors dans le second état stable.

Les interrupteurs I1 et I2 sont eux mêmes des transistors.

Pour une question d'intégration, les transistors bipolaires et les résistances sont remplacés par des transistors à effet de champs de type MOS. Nous obtenons alors le schéma suivant:

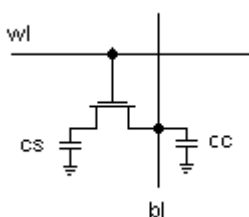


Cellule SRAM CMOS à 6 transistors

2 Les mémoires dynamiques (DRAM)

DRAM signifie Dynamic Random Access Memory c-à-d mémoire à accès aléatoire dynamique. Ces mémoires sont dites volatiles c-à-d que l'absence de tension d'alimentation provoque la perte des données comprises dans ces mémoires. Ces mémoires sont accessibles en lecture/écriture.

La cellule mémoire élémentaire repose sur la capacité parasite grille substrat d'un transistor MOS.



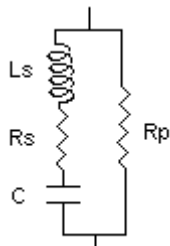
C_s est la capacité de stockage 10pFd

C_c est la capacité parasite de colonne 150 pFd

Cette cellule mémoire est confrontée au problème de la perte de charge de la capacité de stockage. On peut voir cette perte sous deux aspects:

- 1) Une capacité n'est pas parfaite et elle va se décharger naturellement plus ou moins rapidement en fonction de la technologie utilisée pour la fabriquer.

Nous pouvons donner le schéma électrique d'une capacité:



La capacité réelle présente une capacité idéale C qui se trouve en parallèle sur une résistance Rp qui est responsable de sa décharge naturelle.

La résistance Rs aura comme conséquence que la charge de la capacité idéale ne sera pas instantanée.

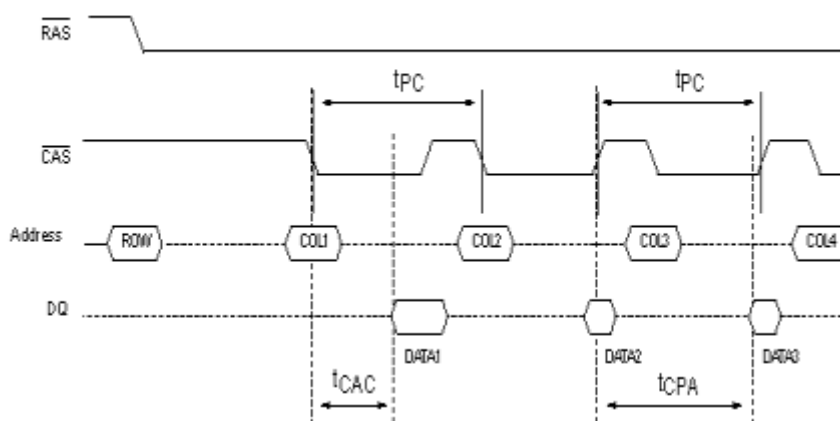
- 2) Lorsque l'on va accéder à la cellule mémoire pour en lire le contenu, la capacité Cs va se décharger dans la capacité de colonne Cc et de ce fait, comme la capacité Cc est grande par rapport à Cs, la charge de Cs va s'en trouver réduite.
Si on veut calculer la tension résultante aux bornes de Cs, nous aurons la formule suivante:
 $V_{cs} = V_s * C_s / (C_s + C_c)$. Valable si nous supposons que la charge initiale dans la capacité Cc est nulle.

Les différentes techniques de rafraîchissement seront étudiées dans un paragraphe ultérieur. Les mémoires DRAM se présentent sous la forme d'un tableau de cellules. Cette table est adressée par l'intermédiaire d'un décodeur de ligne et d'un décodeur de colonne. Pour minimiser la taille des composants, le bus d'adresse est multiplexé c-à-d que les mêmes fils serviront pour recevoir soit l'adresse de la ligne, soit l'adresse de la colonne. Chaque adresse sera validée par les deux bornes suivantes: RAS (Row Address Strobe) et CAS (Column Address Strobe).

Un accès typique à une cellule mémoire se fera de la façon suivante. Premièrement l'adresse de la ligne est fournie et après une certaine période de temps le signal RAS est activé. Cette activation provoquera le chargement de la ligne sélectionnée dans les amplificateurs sensitifs et la mémorisation de l'adresse dans une petite mémoire appelée "buffer". Ensuite, l'adresse de la colonne est fournie et lorsque le signal CAS est activé, l'amplificateur sensitif ainsi sélectionné fournira sa donnée sur le buffer de sortie de la mémoire.

Sachant que tous les amplificateurs sensitifs contiennent l'ensemble des colonnes d'une même ligne, on peut envisager un mode d'accès dans les mémoires dynamiques nécessitant la seule fourniture du numéro de colonne si celle ci appartient à la même ligne que la cellule préalablement accédée. C'est ce que l'on appelle les mémoires FPM c-à-d Fast Page Mode dont voici le chronogramme.

Fast Page Mode:



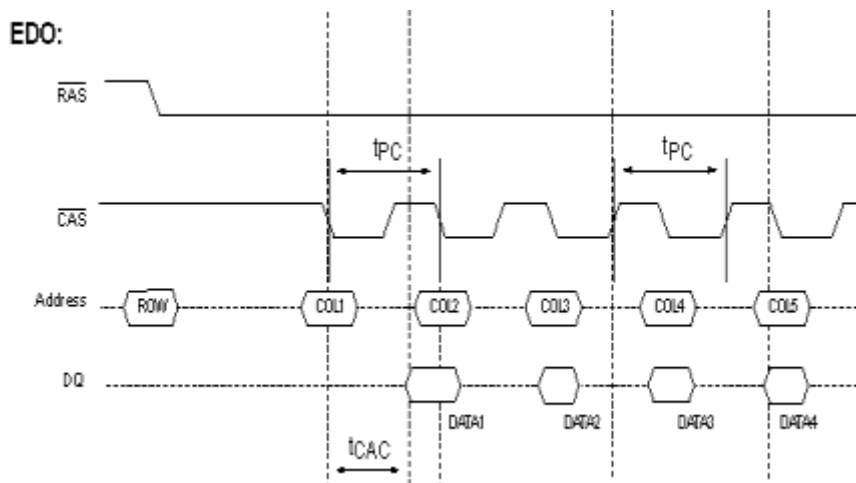
Prenons comme exemple la mémoire MT4LC4M4B1-6.

Cette mémoire présente un temps d'accès de 60nsec. C'est le temps qui sépare la réception par la mémoire du signal RAS et la fourniture de la donnée dans le cas d'un accès en lecture.

Pour les données suivantes appartenant à la même ligne, le temps d'accès est celui correspondant au temps qui sépare la fourniture de la donnée de la réception du signal CAS. On retrouve dans les caractéristiques de cette mémoire 30nsec.

Une façon particulière de noter les mémoires FPM était 6-3-3-3 pour un bus cadencé à 66MHz c-à-d que pour un accès à la première donnée, il fallait compter 6 cycles d'horloge tandis que pour les suivantes, uniquement 3 cycles d'horloge processeur (ce qui correspond bien à la moitié comme renseigné dans la mémoire ci-dessus). Ces mémoires disparaissent avec l'arrivée sur le marché des processeurs Pentium 1 au profit des mémoires EDO (Enhanced Data Output). Dans ces mémoires, on peut fournir l'adresse de la colonne suivante à laquelle on désire accéder tandis que la donnée correspondant à la cellule précédente se trouve toujours sur le buffer de sortie de la mémoire. Cela crée donc un chevauchement des accès permettant de gagner du temps sur chaque cycle.

On retrouve alors le chronogramme suivant:

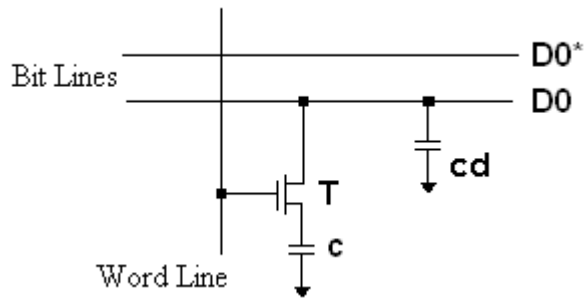


Pour les mémoires FPM, pour une cadence d'horloge de 66Mhz, nous retrouvons une caractéristique de temps de cycle de 6-3-3-3 ou 5-3-3-3. Pour les mémoires EDO, du fait que l'on puisse déposer l'adresse de la cellule suivante alors que la donnée de la cellule précédente est toujours dans le buffer de sortie, nous retrouvons un temps de cycle de 5-2-2-2, soit un gain de quatre cycles sur l'accès des quatre données.

2.3 Le rafraîchissement des mémoires dynamiques.

2.3.1 Aspects techniques.

Comme renseigné dans le paragraphe précédent, une mémoire dynamique est composée d'une capacité comme cellule mémoire élémentaire et de ce fait, cette capacité va se décharger naturellement mais également se décharger lors d'un cycle de lecture.



Supposons que la capacité C soit chargée à la tension V_0 et la capacité C_d chargée à la tension V_{d0} . Lorsque le transistor T sera conducteur, la tension résultante aux bornes de la capacité C sera alors :

$$V_{\text{final}} \cdot (C + C_d) = V_0 \cdot C + V_{d0} \cdot C_d.$$

a) Imaginons la cellule mémoire devant contenir l'équivalent d'une valeur logique 0, nous obtiendrons alors les données suivantes :

$$V_0 = \text{Gnd}$$

$$V_{d0} = \frac{1}{2} V_{cc} = 1.65\text{V}$$

En partant du principe que la capacité C_d est dix fois plus grande que la capacité C, nous obtiendrons le calcul suivant :

$$V_{\text{final}} \cdot 11C = 0 + 1.65 \cdot 10C.$$

$$V_{\text{final}} = 1.65 \cdot (10/11) = 1.5\text{V}$$

b) Imaginons la cellule mémoire devant contenir l'équivalent d'une valeur logique 1, nous obtiendrons alors les données suivantes :

$$V_0 = V_{cc} = 3.3\text{V}$$

$$V_{d0} = \frac{1}{2} V_{cc} = 1.65\text{V}$$

En partant du principe que la capacité C_d est dix fois plus grande que la capacité C, nous obtiendrons le calcul suivant :

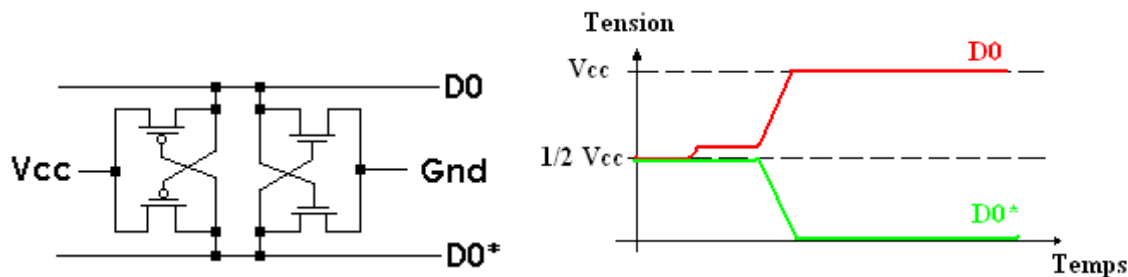
$$V_{\text{final}} \cdot 11C = 3.3 \cdot C + 1.65 \cdot 10C.$$

$$V_{\text{final}} = 1.8\text{V}$$

Si nous considérons la tension sur la ligne D0 comme étant de 1.65, nous pouvons établir le tableau suivant :

Etat logique	Tension V_{d0}	Tension V_{d0}^*	Delta $V_{d0} - V_{d0}^*$
1	1.8V	1.65V	0.15V
0	1.5V	1.65V	-0.15V

Il est évident que la lecture du contenu de la cellule mémoire modifie la charge de la capacité C. Il sera donc important, à la suite de cette phase de lecture, de restaurer les charges dans la capacité de stockage en utilisant un amplificateur différentiel sensitif (differential sense amplifier). Cet amplificateur aura la structure simplifiée suivante :



Le chronogramme de droite représente l'évolution des tensions sur les bornes D0 et D0* de l'amplificateur sensitif lorsque une cellule mémoire correspondant à un état logique '1' est lue. La charge de la capacité est donc restaurée à son état d'origine. Lors d'un accès en lecture, nous avons cependant supposé que la tension initiale sur les bornes D0 et D0* était de $\frac{1}{2} V_{cc}$. Il est important que ces lignes de la DRAM soient maintenues à un potentiel de $\frac{1}{2} V_{cc}$ avant toute opération de lecture. C'est ce que l'on appelle le pré chargement de la DRAM. Ce temps de pré chargement apparaît clairement dans le chronogramme d'un cycle de lecture d'une mémoire dynamique (voir paragraphe suivant).

2.3.2 Les différents types de rafraîchissement.

Comme renseigné dans le paragraphe précédent, le rafraîchissement consiste simplement à accéder à une ligne et automatiquement, toutes les colonnes de cette ligne sont rafraîchies.

Prenons par exemple une mémoire FPM de chez Micron, la MT4LC4M4B1. Cette mémoire de 4Meg x 4, présente une organisation matricielle de 2048 lignes x 2048 colonnes, les 2048 lignes devant être rafraîchies toutes les 32 msec. Pour y arriver, plusieurs solutions sont envisageables :

- 1) Rafraîchir toutes les lignes en une seule fois et attendre ensuite 32 msec avant de recommencer. C'est ce que l'on appelle le rafraîchissement en rafale (Burst Refresh).
- 2) Rafraîchir une ligne toutes les $\frac{32\text{msec}}{2048\text{lignes}} = 15.63\mu\text{sec}$. C'est ce que l'on appelle le rafraîchissement en coup par coup (One Shot Refresh).

Du côté du microprocesseur, le rafraîchissement nécessite que ses bus soient disponibles. Cette disponibilité est possible pendant les cycles de décodage des instructions (cycles fetch) mais également en demandant au microprocesseur l'utilisation de ses bus en utilisant les bornes Hold et Hold Acknowledge. C'est que l'on appellera le rafraîchissement caché (Hidden Refresh) et le rafraîchissement forcé (Forced refresh).

Le rafraîchissement en rafale sera généralement lié au rafraîchissement en rafale tandis que le rafraîchissement en coup par coup sera lié au rafraîchissement caché.

2.4 Les temps d'accès, de cycle et de décodage.

2.4.1 Les mémoires statiques.

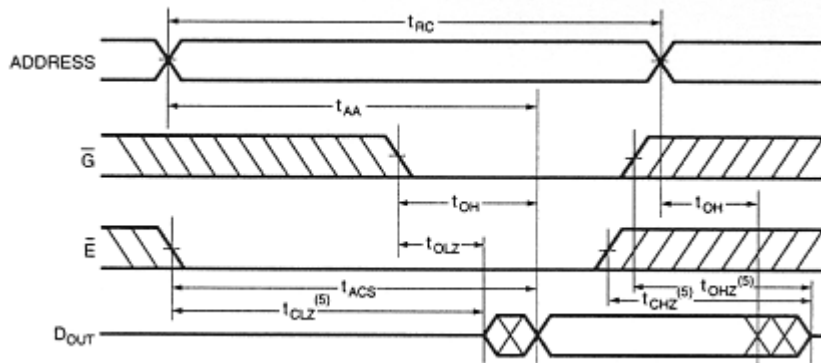
Nous prendrons comme mémoire, une mémoire statique 32k x 8 de chez Mosel dont voici le chronogramme correspondant à un accès en lecture.

Le temps d'accès d'une mémoire est toujours donné en rapport à un cycle de lecture et correspond au temps séparant la réception de l'adresse par la mémoire et la fourniture de la donnée par cette dernière. Dans le chronogramme, nous retrouvons le temps t_{AA} ou $t_{a(A)}$ qui dans le cas de notre mémoire, correspond à 70 nsec et qui va figurer sur le boîtier sous la forme MS62256L-70.

Le temps de cycle d'une mémoire correspond au temps que prend dans le cas de notre chronogramme un cycle complet de lecture et que l'on puisse alors démarrer le cycle suivant. Dans le cas de la mémoire choisie, le temps de cycle est de 70 nsec.

Nous pouvons en déduire que dans le cas des mémoires statiques, le temps de cycle est identique au temps d'accès.

Paramètre	Paramètre	Min	Max	Unité
t_{RC}	Read Cycle Time	70		ns
t_{AA}	Address Access Time		70	ns
t_{ACS}	Chip Enable Access Time		70	ns
t_{OH}	Output Hold from Address Change	10		ns



E: Chip Enable Input
G: Output Enable Input

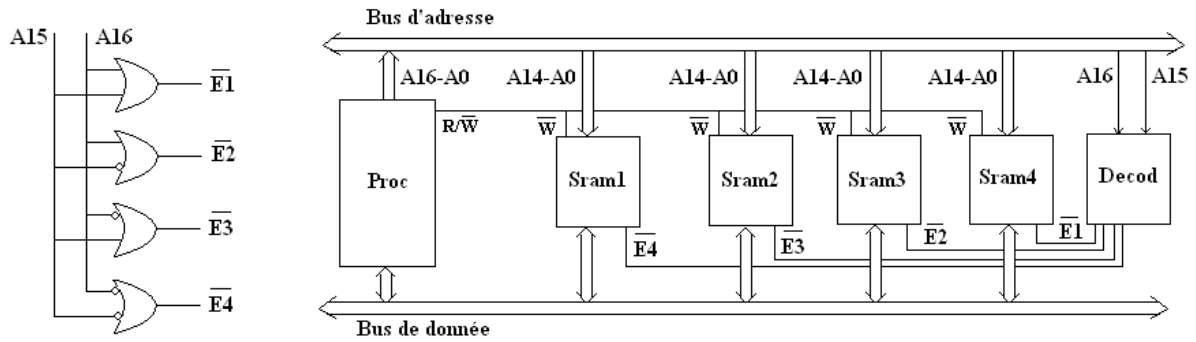
Pour bien comprendre l'utilité de la borne 'Chip Select', nous allons considérer un microprocesseur ayant un bus d'adresse de 17 bits pour un espace adressable de 128Ko (2^{18}). Si nous limitons à l'utilisation des mémoires de 32K x 8, il nous faudra donc 4 boîtiers pour couvrir les 128Ko.

Adresse du microprocesseur (17 bits) $A_{16}-A_0$	Adresse de la mémoire (15 bits)	N° boîtier
00000000000000000 jusque 00111111111111111	000000000000000 jusque 111111111111111	1
01000000000000000 jusque 01111111111111111	000000000000000 jusque 111111111111111	2
10000000000000000 jusque 10111111111111111	000000000000000 jusque 111111111111111	3
11000000000000000 jusque 11111111111111111	000000000000000 jusque 111111111111111	4

Soit l'adresse suivante déposée sur le bus d'adresse par le microprocesseur : **10100111100100101**. L'ensemble des mémoires connectées sur le bus d'adresse ne voient que les 15 bits de poids faible de cette adresse, à savoir **100111100100101** bien que cette demande soit adressée au boîtier numéro 3 et pas aux autres. Il faudra donc sélectionner le boîtier 3 et pas les autres grâce à la borne E de chacune des mémoires.

Adresse du microprocesseur (17 bits)	E boîtier 1	E boîtier 2	E boîtier 3	E boîtier 4
00xxxxxxxxxxxxxxxxx	0	1	1	1
01xxxxxxxxxxxxxxxxx	1	0	1	1
10xxxxxxxxxxxxxxxxx	1	1	0	1
11xxxxxxxxxxxxxxxxx	1	1	1	0

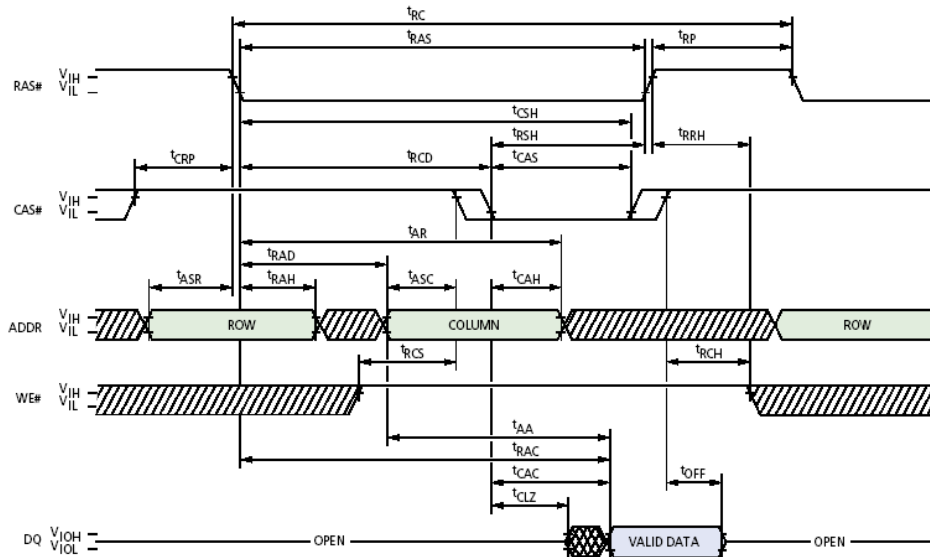
Nous pouvons donc envisager le schéma logique suivant :



Il est clair que si l'on tient compte du temps de propagation des signaux dans le décodeur d'adresse, le signal d'activation du boîtier arrivera après l'adresse. Nous allons analyser le chronogramme pour déterminer s'il y a des limitations dans le temps que le décodeur peut mettre pour traiter l'adresse. Dans le cas la mémoire choisie, le temps t_{ACS} est également de 70 nsec, ce qui nous laisse penser que si le décodage d'adresse prend x nsec pour traiter l'adresse, la donnée ne sera disponible sur le bus de données que $(70+x)$ nsec. Si nous avons choisi de sélectionner les boîtiers en utilisant la borne G au lieu de la borne E, nous aurions alors pu avoir un temps de décodage possible de $t_{AA} - t_{OH}$ c-à-d $70 - 10 = 60$ nsec. Si nous choisissons des portes logiques ayant un temps de propagation de 10 nsec, nous pourrions nous permettre d'avoir 6 portes en cascade entre le signal de sortie et les signaux d'entrée.

2.4.2 Les mémoires dynamiques.

Le temps d'accès pour les mémoires dynamiques est un peu plus compliqué que celui des mémoires statiques. Nous nous baserons pour nos explications sur le chronogramme d'un cycle de lecture d'une mémoire Micron MT4LC4M4B1DJ-6.



Nous retrouverons les temps d'accès suivants :

t_{AA} : temps d'accès à partir de l'adresse de la colonne

t_{RAC} : temps d'accès à partir du signal RAS

t_{CAC} : temps d'accès à partir du signal CAC

Généralement, c'est le temps t_{RAC} qui caractérise la mémoire. Le modèle MT4LC4M4B1DJ-6 correspond à une mémoire ayant un temps d'accès t_{RAC} de 60 nsec (temps maximum).

Si nous nous intéressons au temps de cycle t_{RC} , celui-ci correspond à la somme du temps t_{RAS} et du temps de pré chargement t_{RP} . Le temps de cycle est pour cette mémoire de $40 \text{ nsec} + 60 \text{ nsec} = 100 \text{ nsec}$.

Le fait que le temps de cycle soit plus grand que le temps d'accès peut poser des problèmes : en effet, le processeur ne pourra démarrer un nouveau cycle dans la même mémoire que si le cycle précédent est terminé. Lors d'un cycle de lecture, le microprocesseur pourra obtenir la donnée 60 nsec après avoir déposé l'adresse mais il devra quand même attendre 40 nsec de plus pour démarrer un nouveau cycle. Pour limiter l'effet néfaste de ce temps de pré chargement sur les caractéristiques générales du système à micro processeur, nous allons envisager l'utilisation de plusieurs boîtiers avec un décodage tel que deux adresses successives se trouveront dans des boîtiers différents. Nous parlons alors de banques de mémoire avec la technique du croisement des banques : l'accès à l'une des banques permettra à l'autre de récupérer.

Afin d'éviter l'utilisation de plusieurs boîtiers, les fabricants de mémoires ont intégré dans le même boîtier plusieurs banques, chaque banque pouvant être accédée indépendamment l'une de l'autre. Du fait que cette mémoire soit une mémoire FPM, nous pouvons donc effectuer des transferts par page (en rafale), la première donnée s'obtient au bout d'un temps minimum de 45 nsec ($t_{RAC} \text{ max } 60\text{nsec}$) tandis que les données suivantes s'obtiennent toutes les 25 nsec.

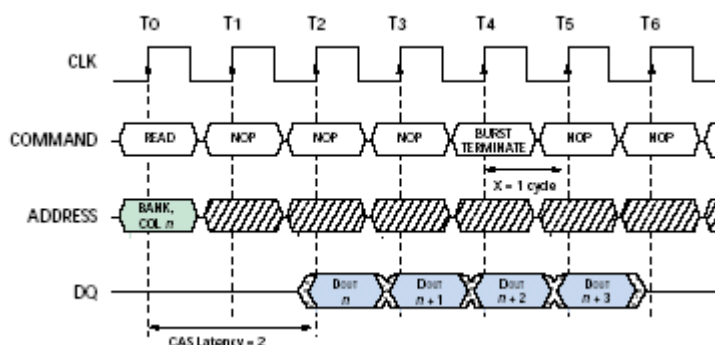
2.5 Les mémoires synchrones.

Comme son nom l'indique, cette catégorie de mémoire se base sur un fonctionnement synchrone c-à-d que l'on va retrouver un signal d'horloge. Seules les mémoires vives subissent cette évolution. On retrouve alors des mémoires dynamiques synchrones et des mémoires statiques synchrones, nous nous intéresserons dans ce chapitre aux mémoires dynamiques synchrones utilisées comme mémoire vive sur nos cartes mère.

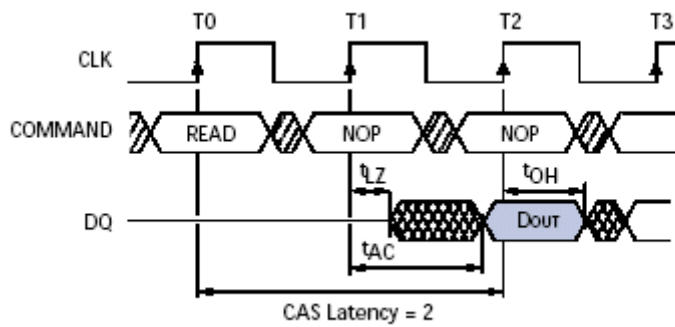
Pour les mémoires synchrones, nous retrouvons comme première mémoire la mémoire SDRAM (RAM dynamique synchrone) c à d que parmi les bornes de la mémoire, nous retrouvons un signal d'horloge sur lequel les différents accès à la mémoire seront synchronisés.

Lorsque l'on caractérise la mémoire, on n'évoque plus le temps d'accès mais la fréquence du bus avec laquelle nous pouvons travailler sur ces mémoires. On parle alors de mémoire SDRAM de type PC100 ou PC133. On évoque également le cycle correspondant à cette horloge en pensant naïvement qu'il s'agit du temps d'accès. On parle de mémoire à 10nsec, 8nsec...

Reprenons le chronogramme d'une telle mémoire afin d'en calculer le temps d'accès: nous retrouvons le temps t_{RCD} (RAS to CAS Delay time), le temps t_{RP} (RAS Precharge time) et le temps CAS Latency qui joueront un rôle important pour déterminer la vitesse maximale du bus à laquelle le module mémoire peut travailler. On atteint généralement de timings du type 4-1-1-1. Il faut quatre cycles pour accéder à la première donnée, ce qui correspond pour une horloge cadencée à 100Mhz à un temps d'accès de 40nsec. Les données suivantes sont accédées avec une cadence de l'ordre de 1 donnée toutes les 10nsec.



Le temps CAS est le délai en cycles d'horloges entre l'enregistrement de la commande READ échantillonné sur le flanc montant de l'horloge et la disponibilité de la donnée sur la sortie.

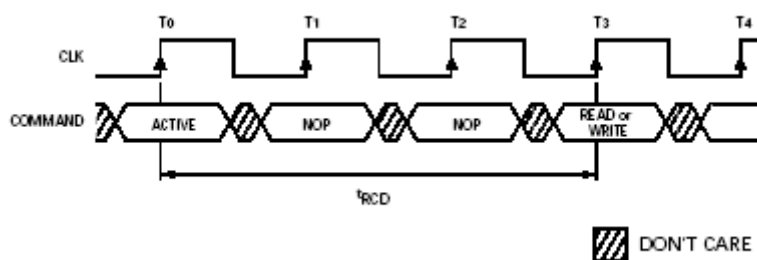


La mémoire fonctionnant toujours de part sa nature en mode asynchrone, si nous travaillons avec une horloge ayant un temps de cycle plus petit, il faudra augmenter le nombre de coup d'horloge correspondant au temps CAS Latency. Si nous nous référons au tableau de la mémoire, nous obtiendrons les données suivantes:

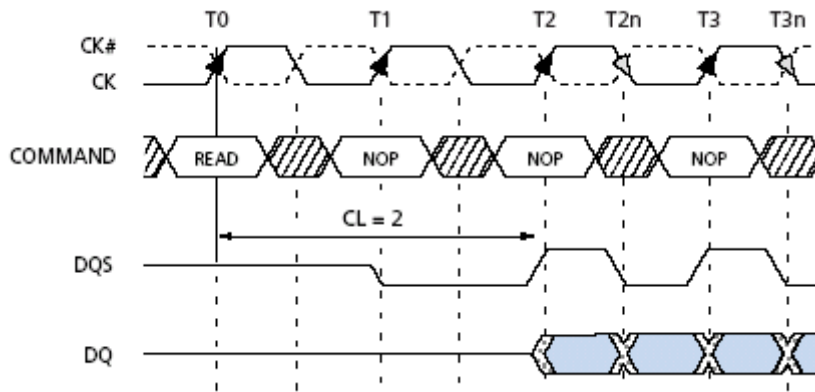
SPEED	ALLOWABLE OPERATING FREQUENCY (MHz)	
	CAS LATENCY = 2	CAS LATENCY = 3
-6	-	≤ 166
-7E	≤ 133	≤ 143
-75	≤ 100	≤ 133
-8E	≤ 100	≤ 125

Pour une mémoire de type -8E avec une fréquence de bus de 100Mhz, nous devons régler au niveau du BIOS de notre carte mère le temps CAS Latency Time à 2 (c à d deux cycles d'horloge correspondant alors à 20 nsec). Si nous voulons changer la fréquence de notre bus système sur notre carte mère et la faire passer à 125MHz, nous devons régler ce paramètre à 3 (c à d un temps de $3 \times 8 \text{ nsec} = 24 \text{ nsec}$).

Si nous considérons le temps tRCD, nous obtenons un temps constant de 20nsec quel que soit le modèle considéré. Nous obtenons alors un temps d'accès de l'ordre de 40nsec et 44nsec.



Les mémoires DDR se basent sur le même concept que les mémoires SDRAM excepté que nous envisageons lors des rafales un transfert sur le flanc montant et sur le flanc descendant ce qui permet de doubler la vitesse de transfert, d'où la dénomination donnée à cette technologie: DDR SDRAM (Double Data Rate). Nous parlons alors de DDR 200, DDR266, DDR333 et DDR400.



Nous retrouverons le même paramétrage pour le temps CAS Latency Time suivant le tableau suivant:

SPEED	ALLOWABLE OPERATING FREQUENCY (MHz)	
	CL = 2	CL = 3
-5	$83 \leq f \leq 125$	$83 \leq f \leq 200$
-55	$83 \leq f \leq 100$	$83 \leq f \leq 183$
-6	$83 \leq f \leq 100$	$83 \leq f \leq 166$
-65	$83 \leq f \leq 100$	$83 \leq f \leq 150$

On retrouve une autre façon de pouvoir identifier un module mémoire, non pas sur la fréquence d'horloge mais sur le débit de données que cette mémoire peut avoir. Si nous envisageons une mémoire dont la fréquence de fonctionnement serait de 100MHz, nous pouvons, lors des rafales, espérer le débit suivant: $200 * 8 = 1600$ $100\text{MHz} * 2$ (2 flancs) $* 8$ (nombre d'octets transférés sur le bus de données). On parle alors de mémoire PC1600

Pour la DDR266, nous obtenons le calcul suivant: $133 * 2 * 8 = 2100$. On parle de PC2100.

Pour la DDR333, nous obtenons le calcul suivant: $166 * 2 * 8 = 2700$. On parle de PC2700.

Pour la DDR400, nous obtenons le calcul suivant: $200 * 2 * 8 = 3200$. On parle de PC3200.

Un des problèmes est le besoin de bande passante de plus en plus important du côté microprocesseur. En effet, le QuadPumped permet d'effectuer 4 transferts par cycle, ce qui nous donne les caractéristiques suivantes pour les processeurs Intel P4:

P4 fréquence d'horloge 100Mhz (FSB $100 * 4 = 400$): $400 * 8 = 3200$ ce qui correspond à la moitié de la bande passante offerte par le mémoires de type PC1600.

Du fait que la demande de transfert du microprocesseur est chaque fois le double de la vitesse de transfert des mémoires, il y a, à ce niveau, un goulot d'étranglement. C'est une des raisons pour laquelle les concepteurs de cartes mère ont imaginé la mise en place de deux canaux capables de fournir simultanément des données sur 64 bits fournissant ainsi 128 bits au Chipset avec la même cadence d'horloge. Nous obtenons alors les débits suivants:

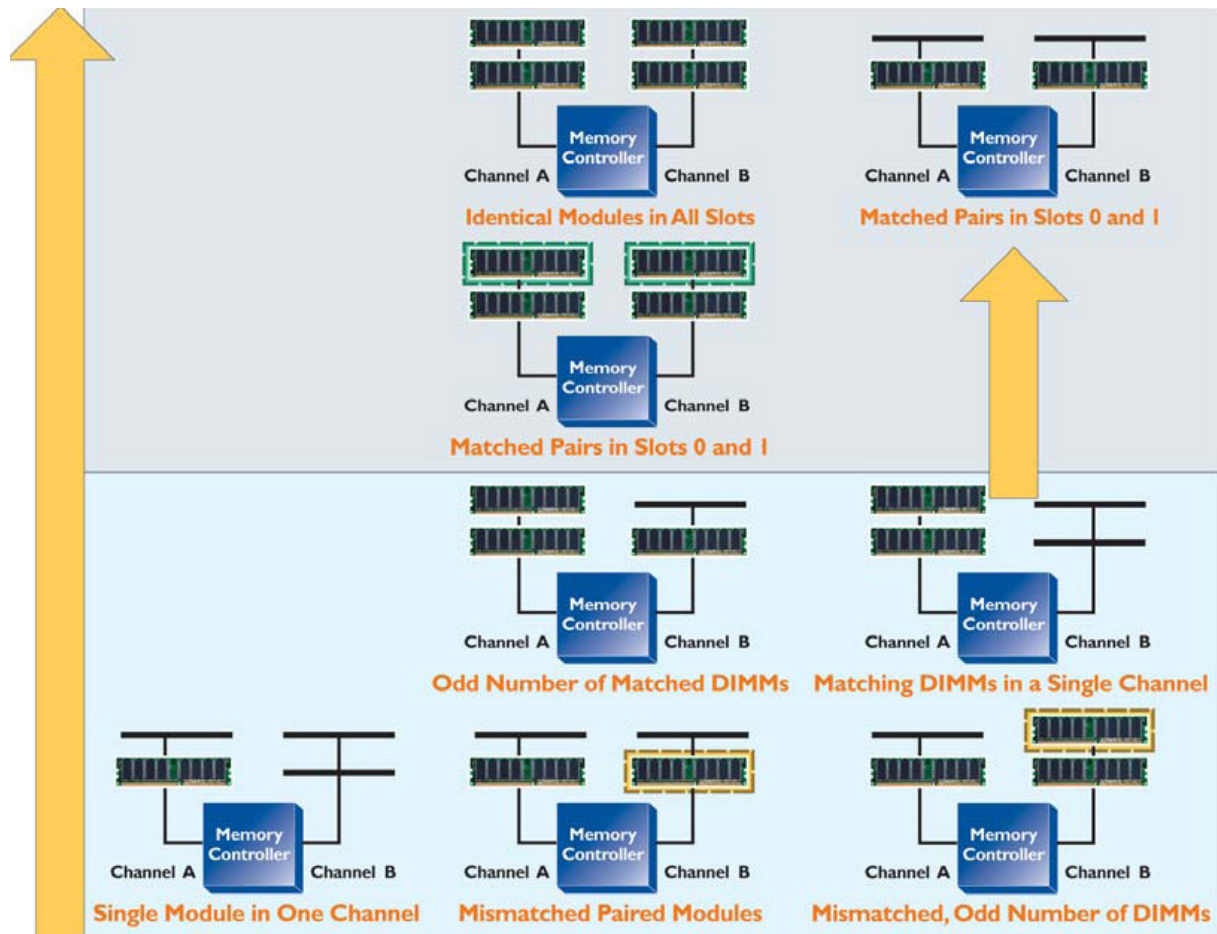
DDR266 \rightarrow PC2100 $\rightarrow 2100 * 2 = 4200$ (4,2Go/sec)

DDR333 \rightarrow PC2700 $\rightarrow 2700 * 2 = 5400$ (5.4Go/sec)

DDR400 \rightarrow PC3200 $\rightarrow 3200 * 2 = 6400$ (6.4Go/sec).

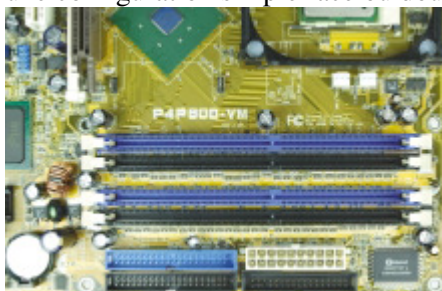
Si nous considérons les processeurs P4 cadencés à 200Mhz avec un FSB correspondant de 800, nous obtiendrons une bande passante souhaitée par le microprocesseur de $800 * 8 = 6,4$ Go/sec, ce que la mémoire est capable de fournir.

Pour les plateformes Intel basées sur des Chipset 865 et 875, les règles à respecter pour que la carte mère puisse fonctionner sur un seul canal ou sur les deux canaux sont les suivantes:



Les règles générales étant:

- 1- Les deux modules doivent avoir la même capacité
- 2- Les deux modules doivent avoir la même vitesse (PC2700 ou PC3300)
- 2- Les deux modules doivent avoir le même nombre de boîtiers et doivent correspondre tous les deux à une configuration simple face ou double face.



La dernière technologie en matière de mémoire est la DDRII. Pour bien comprendre l'évolution, nous devons intégrer une nouvelle caractéristique dans les mémoires qui est le prefetch. Pour bien comprendre, nous allons reprendre l'architecture d'une mémoire comme composée de trois éléments fondamentaux : la cellule mémoire, le buffer d'entrée/sortie et le bus de données. Dans le cas d'une mémoire de type PC100, l'ensemble de ces trois groupes fonctionne à une cadence de 100 MHz et la cellule mémoire fournit un seul bit.

Pour la mémoire de type DDR I, le fait de pouvoir travailler pour la DDR PC 1600 à une fréquence de 200 MHz en effectuant des transferts sur le flanc montant et sur le flanc descendant (2x100MHz), oblige les constructeurs, du fait des limitations technologiques propres aux mémoires, de travailler avec une double cellule mémoire capable de fournir 2 bits en conservant une cadence de fonctionnement de 100 MHz. Ce nombre de bits transférés par la cellule mémoire élémentaire s'appelle le prefetch.

Comme nous l'avons vu précédemment, plus la fréquence de fonctionnement augmente, plus la dissipation thermique du composant est importante. C'est pour cela que pour la technologie DDR II, l'on passe à un prefetch de 4, ce qui nous permet de pouvoir travailler avec une fréquence de fonctionnement moindre et de ce fait pouvoir diminuer la tension d'alimentation qui passe de 2,5V à 1,8V

Type	Fréquence interne	Prefetch	Vitesse	Débit
PC-3200	200MHz	2	400MHz	400MHz*8
PC2-3200	100MHz	4	400MHz	400MHz*8
PC2-5300	133MHz	4	533MHz	533MHz*8

D'autres améliorations sont apportées à ces mémoires, nous citerons en premier l'insertion de la terminaison résistive sur la barrette mémoire et non plus sur la carte mère, nous retrouverons le terme OTD (On Die Termination). Cette résistance est activable par le BIOS.

Nous citerons en deuxième lieu l'OCD ou Off Chip Driver Calibration.

Début février 2005, Samsung annonce qu'il est le premier à avoir produit un prototype de mémoire DDR3. Fonctionnant en 1.5V, cette puce de 512Mb gravée en 80nm est de type DDR3-1066. A titre de comparaison, la DDR fonctionne en 2.5V et la DDR2 fonctionne en 1.8V. Pour que cela soit possible, le prefetch, qui était passé de $2n$ à $4n$ bits lors du passage de la DDR à la DDR-2, passe désormais à $8n$ bits. Nous conservons donc pour cette nouvelle génération de mémoire une fréquence interne de 133 Mhz, tandis qu'avec un prefetch de 8, nous pouvons atteindre une vitesse de $133*8=1064$ Mhz ce qui nous permettra donc d'atteindre un débit de $1064\text{Mhz}*8=8\text{Go/sec}$. Samsung annonce la commercialisation de cette mémoire pour début 2006.

Selon un autre constructeur de mémoire Micron, les prévisions sont les suivantes: pour début 2007, nous pourrions retrouver sur le marché la DDR3-1333 et pour 2008, la DDR3-1600.

En octobre 2005, Samsung annonce la sortie de la DDR4 pour les cartes graphiques. Ces modules portent le nom de GDDR4 et permettent des débits de 10Go/sec.

2.6 Les différents types de mémoire.

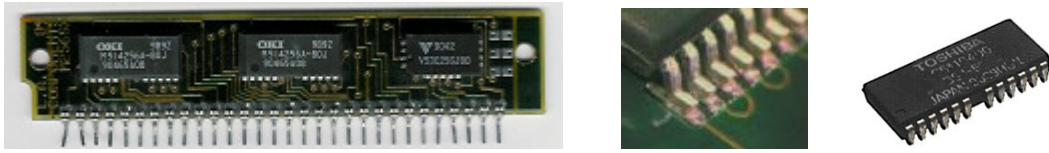
a) Les boîtiers DIP (Dual In Line Package).

A l'origine, les boîtiers mémoires sont semblables aux composants électroniques classiques sous forme de boîtiers de type DIP. Tout boîtier de ce type est identifié par le nombre de bornes ainsi que la largeur du boîtier : on parlera par exemple de boîtiers 14 DIP 300 ou 14 DIP 600.

14 DIP 300 signifie que l'on retrouve 14 bornes en deux rangées parallèles de 7 bornes espacées chacune de 1/10 de pouce. Pour la valeur 300, nous aurons une largeur de 3/10 de pouce. Dans les premiers ordinateurs, ces boîtiers étaient directement intégrés sur la carte mère.



b) Les barrettes mémoire SIP (Single In Line Package)



Nous retrouvons pour ces barrettes, des boîtiers mémoire montés en surface avec des modèles de boîtiers de type SOJ ou TSOP. Les bornes des composants sont espacées de 1/20 de pouce.

Fort semblable aux barrettes mémoire de type SIMM, elles comprennent 30 contacts avec un bus de données de 8 bits. Elles sont en général soudées sur la carte mère ce qui ne permet pas au commun des mortels de pouvoir facilement effectuer une mise à jour mémoire de son ordinateur.

c) Les barrettes mémoire SIMM (Single In Line Memory Module) 30 contacts et 72 contacts

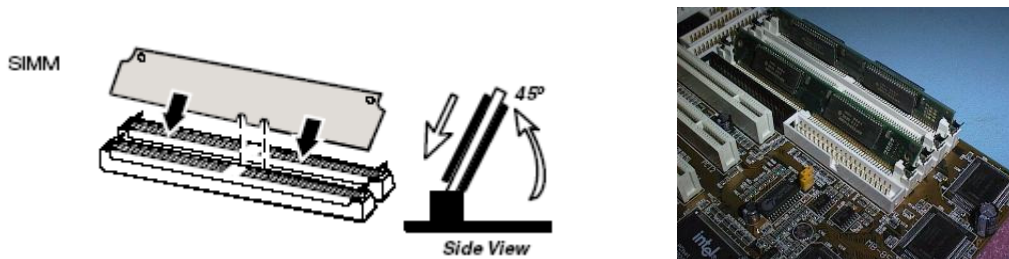


Les barrettes SIMM 30 contacts permettent des transferts sur 8 bits. Dans le cas par exemple des ordinateurs équipés d'un processeur de type 486, processeur 32 bits avec un bus de données de 32 bits, il faut prévoir d'installer les mémoires par groupe de quatre barrettes pour couvrir ainsi les 32 bits du bus de données.



Les barrettes SIMM 72 contacts apparaissent permettant ainsi de n'utiliser qu'une seule barrette avec ce type de processeur mais avec l'apparition des premiers PENTIUM, processeur 32 bits avec un bus de données de 64 bits il faut installer les barrettes par groupe de deux.

Contrairement aux mémoires SIP, celles-ci peuvent être ajoutées facilement sur la carte mère grâce à des connecteurs.



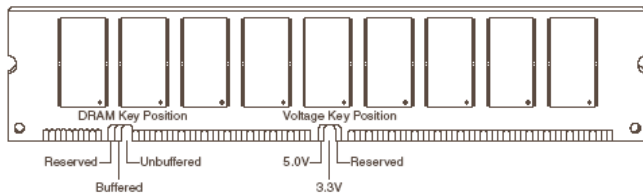
d) Les barrettes mémoire DIMM (Dual In Line Memory Module)

La principale différence entre les barrettes SIMM et DIMM est que, sur la SIMM, les broches situées à l'opposé de la carte, sont "liées" pour former un seul contact électrique, sur la DIMM, les broches opposées demeurent électriquement isolées et forment deux contacts séparés. Parmi les

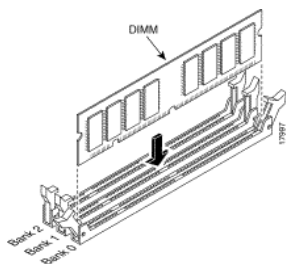
autres différences physiques entre la DIMM et la SIMM à 72 broches, il faut citer la longueur du module, son nombre d'encoches, ainsi que la manière dont il est inséré dans le connecteur.

Outre la tension d'alimentation qui peut différer, nous retrouvons deux modèles distincts de modules : ceux que l'on appelle les modules mémoire UNBUFFERED et les modules mémoire REGISTERED/BUFFERED. Si l'on veut comprendre la différence entre ces deux modèles, nous devons parler de la sortance des composants électroniques c à d du nombre d'entrées de composants d'une même technologie que l'on peut mettre sur une même sortie. Dans le cas des modules UNBUFFERED, leur nombre sur une carte mère est limité à quelques uns tandis que dans le cas des modèles REGISTERED/BUFFERED, nous pourrions atteindre une vingtaine de modules. On retrouvera donc ces derniers principalement utilisés sur les cartes mères de serveur où la capacité mémoire offerte est plus importante.

Pour ne pas faire le mauvais choix d'un modèle de mémoire pour une carte mère donnée, ces mémoires sont pourvues de deux encoches dont la position diffère en fonction de la tension d'alimentation et en fonction qu'elle soit ou pas REGISTERED/BUFFERED.

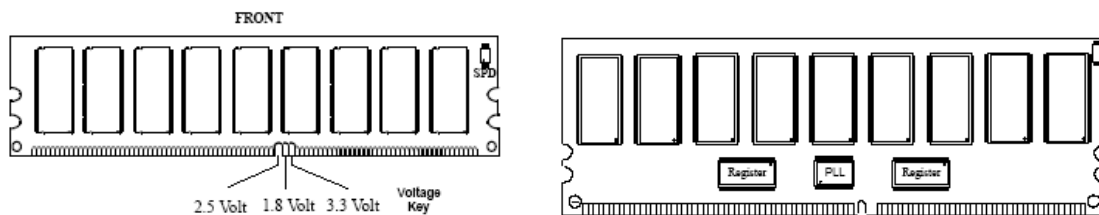


L'insertion du module est différente de l'insertion d'un module SIMM. Celui-ci doit simplement s'insérer verticalement



Bien que portant le même nom, les barrettes mémoire DIMM évoluent en même temps que les différentes technologies mémoires :

- DIMM 168 broches pour les mémoires de type SDRAM
- DIMM 184 broches pour les mémoires de type DDR I. Outre la différence résultant d'un nombre de bornes différent pour une dimension presque identique, les modules DIMM 184 ne disposent que d'une seule encoche dont la position dépend de la tension d'alimentation du module.



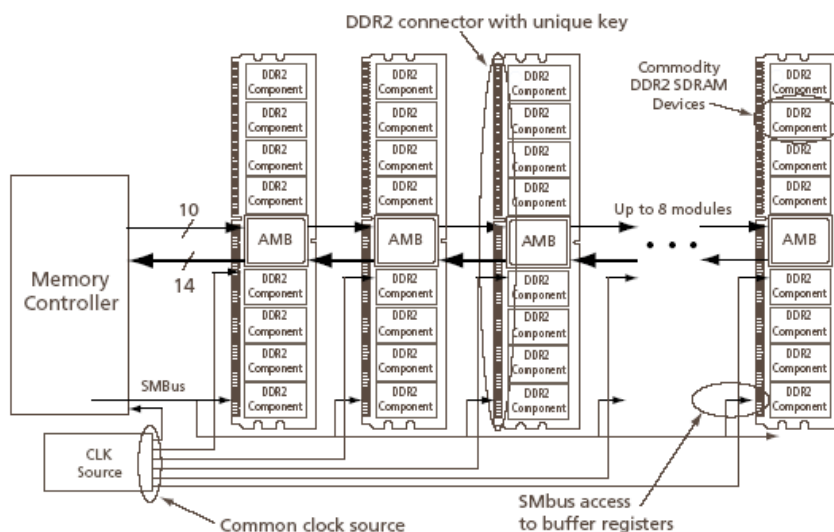
Le fait que le module soit prévu avec un tampon ou pas se marquera par la présence sur le module de composants supplémentaires.

- DIMM 240 broches pour les mémoires de type DDR II

e) Les modules FBDIMM (Fully-Buffered DIMM)

Une des limitations des mémoires DDR2 est liée aux nombres de composants qu'un canal est capable d'accepter. Sur les cartes mères actuelles, nous retrouvons au maximum deux modules par canal ce qui pour les serveurs, peut brider la capacité mémoire. Ces deux modules correspondent à 72 composants

Pour augmenter le nombre de modules sans pour autant augmenter le nombre de broches sur les modules, les commandes, adresses et données sont sérialisées et permettent alors d'adresser 8 buffers par canal, chaque canal pouvant supporter 32 composants, ce qui nous donne 288 composants au total. Nous pouvons donc retrouver un total de 8 modules par canal.



Comparée aux DIMM de type "registred" existants, la technologie FB-DIMM sera capable de supporter 24x plus de quantité de mémoire (192 Go contre 8 Go), tout en proposant une bande passante multipliée par 4 (40 Gbps contre 10 Gbps). De plus le FB-DIMM ne nécessitera que 69 pins, contre 240 actuellement, de quoi proposer des gains de place non négligeable dans les serveurs en racks.

f) Les modules RIMM (Rambus In Line Memory Module)

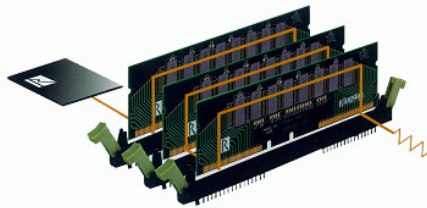
Les modules RIMM se basent sur des boîtiers mémoire appelés RDRAM (Rambus DRAM). Rambus est une société ayant mis au point cette technologie basée sur un bus mémoire propriétaire cadencé dans sa première version à 400Mhz pour une largeur de 16 bits, on parle alors de RDRAM 400. A cette fréquence, la mémoire est capable de travailler sur les deux flancs d'horloge et permet d'atteindre un débit de $400\text{Mhz} \times 2\text{flancs} \times 2\text{ octets} = 1.6\text{Go/sec}$ (à la même époque, la mémoire PC100 en technologie SDRAM permet d'atteindre des débits de 800Mo/sec).

Dans des versions ultérieures nous retrouverons la RDRAM 600 avec un débit de 2.4Go/sec et la RDRAM 800 avec 3.2Go/sec, la RDRAM 1200 avec 4.8Go/sec.

Suite à un accord avec la société Intel, celle-ci est la première à intégrer cette technologie dans son Chipset 820 qui est lié aux processeurs Pentium III fonctionnant avec une cadence de bus externe de 133Mhz. Nous retrouvons les Pentium de la série B (533B et 600B). Ce Chipset ne permet pas de prendre en charge les mémoires SDRAM de type PC100.

Avec la sortie de son processeur Pentium IV, Intel continue sur la même lancée et sort le Chipset 850 qui à son tour ne prend en charge que la Rambus.

Confronté aux prix de cette mémoire, INTEL sort fin décembre 2001 un chipset équivalent au 850 pour Pentium IV mais gérant les mémoires PC133/PC100, l'i845. La version i845D, sortie ensuite, est identique mais gère les mémoires DDR.



Malgré son échec, RAMBUS sort une nouvelle évolution de sa mémoire qui est la XDR. La technologie XDR a des caractéristiques techniques impressionnantes : elle utilise une architecture permettant une interconnexion "point-to-point differential data" qui permet d'atteindre des fréquences de fonctionnement pouvant atteindre 6,4 GHz ! Les interfaces utilisées peuvent de plus aller jusqu'à 128 bits, ce qui donne un débit maximal théorique de 100 Go/s... Pour sa première version, la XDR est proposée dans une configuration permettant des débits allant de 12.8 à 25.6 Go/sec sous la forme d'un packaging classique analogue à la DDR2. Actuellement, aucun constructeur ne s'est aventuré à développer une nouvelle technologie de chipset prenant en charge ces mémoires. Seul Sony intègre la XDR dans ses consoles de jeux PS2.

2.7 Les techniques de détection et de correction des erreurs.

2.7.1 La technique de la parité simple.

Un bit supplémentaire sera ajouté pour chaque groupe de 8 bits de donnée. Donc, pour une organisation des données sur 8 bits, 1 bit supplémentaire et pour une organisation des données sur 32 bits, il faudra compter 4 bits supplémentaires.

Comment calculer la valeur que l'on va donner à ce bit supplémentaire que l'on appelle bit de parité ? En fait, il existe deux façons de calculer ce bit suivant que l'on travaille avec une parité paire ou une parité impaire.

Dans la parité paire, on va choisir la valeur du bit de tel façon que le nombre total de bits à 1 y compris le bit de parité soit pair.

Dans la parité impaire, on va choisir la valeur du bit de tel façon que l'on nombre total de bits à 1 y compris le bit de parité soit impair.

Exemple : soit la donnée suivante : 10010010 le bit de parité ajouté pour la parité paire sera 1

Cette parité s'obtient en effectuant un ou exclusif sur l'ensemble des bits de la donnée.

Il faudra donc alors mémoriser 10010010 1.

En cas d'erreur sur la donnée, exemple 11010010 1, lors de la relecture de la donnée, la parité sera recalculée et l'on remarque alors que la parité paire n'est plus respectée. On peut détecter l'erreur mais il n'est pas possible de la corriger.

L'inconvénient de cette méthode est que si nous avons un nombre pair d'erreurs, aucune erreur ne sera détectée. Reprenons l'exemple précédent et imaginons deux erreurs :

11010010 0. Lors de la relecture de cette donnée erronée, la parité est respectée puisque le nombre de bits à 1 est toujours pair.

Un autre inconvénient de cette méthode est que pour 8 bits de données utiles, il faut ajouter 1 bit supplémentaire (1/8) qui ne sera pas porteur d'une donnée utile. Pour 256 Mo de mémoire, il faut 32Mo de bits supplémentaires...

Attention : il est clair que si une donnée est transmise entre deux équipements (ex : deux pc), il faudra être vigilant que le choix de la parité soit identique. Dans le cas contraire, l'équipement recevant la donnée ainsi que son bit de parité calculé par l'équipement émetteur, considèrera toujours la donnée comme erronée.

2.7.2 La technique de parité verticale et horizontale

Nous retrouvons également l'utilisation d'un calcul de parité basé sur la parité paire ou impaire. Imaginons que l'on doit transmettre un groupe de 8 octets dont un exemple de configuration serait le suivant :

1	0	0	1	0	1	0	1	0
1	1	0	0	0	1	0	1	0
1	0	0	1	1	0	1	0	0
1	1	1	0	0	0	1	1	1
1	0	1	0	1	0	1	0	0
0	0	0	0	0	1	1	1	1
1	1	0	0	1	1	1	1	0
1	0	1	1	0	0	0	0	1

0 1 1 1 1 0 1 1

La parité va donc se calculer de façon horizontale comme dans le cas de la parité simple mais va aussi se calculer sur chacune des colonnes. Nous avons choisi une parité paire pour le calcul. Si nous envisageons des erreurs sur ces données :

1	0	0	1	0	1	0	1	0
1	1	0	0	0	1	0	1	0
1	0	0	0	1	0	1	0	0
1	1	1	0	0	0	0	1	1
1	0	1	0	1	0	1	0	0
0	0	0	0	0	1	1	1	1
1	1	0	0	1	1	1	1	0
1	0	1	1	0	0	0	0	1

0 1 1 1 1 0 1 1

Dans cet exemple, la relecture de la donnée permettra au niveau des lignes de détecter une erreur sur la troisième et sur la quatrième ligne. Cette même relecture permettra au niveau des colonnes, de détecter une erreur sur la quatrième colonne et sur la septième. Le système sera donc capable de détecter les erreurs mais également de pouvoir les corriger puisque l'on sait exactement où se trouvent les bits en erreur (numéro de ligne et numéro de colonne).

Un nombre pair d'erreurs sur une même ligne ou sur une même colonne permettra de détecter les erreurs mais pas de les corriger. Prenons le cas extrême où deux erreurs sur la même ligne correspondent aux mêmes colonnes que deux autres erreurs sur une autre ligne.

1	0	0	1	0	1	0	1	0
1	1	0	0	0	1	0	1	0
1	0	0	1	1	0	1	0	0
1	1	0	0	0	0	0	1	1
1	0	1	0	1	0	1	0	0
0	0	0	0	0	1	1	1	1
1	1	1	0	1	1	0	1	0
1	0	1	1	0	0	0	0	1

0 1 1 1 1 0 1 1

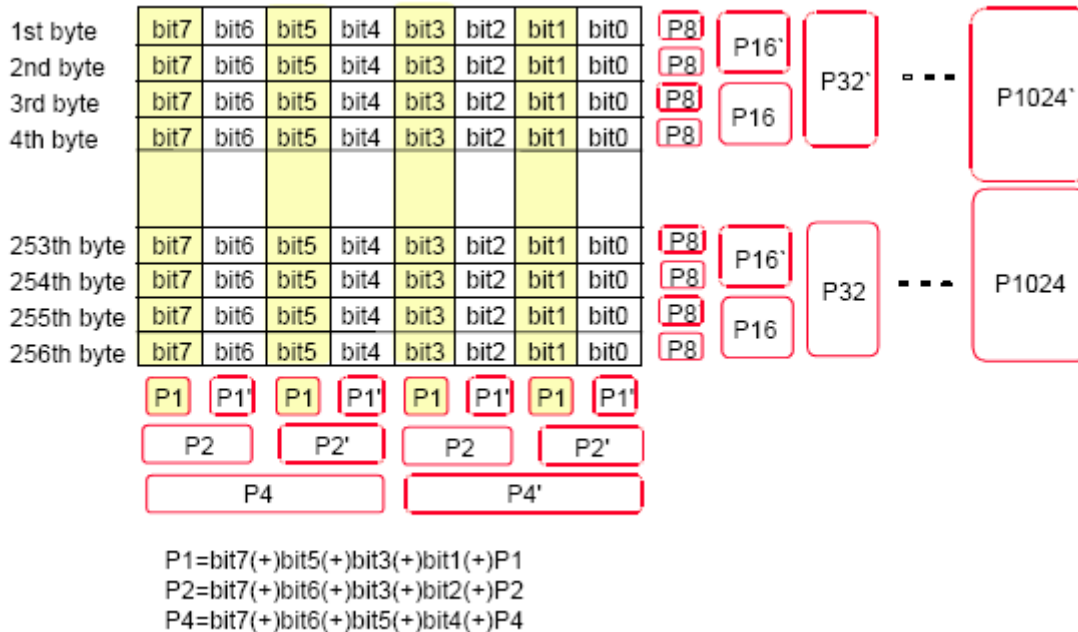
Dans ce cas de figure, aucune erreur ne sera détectée ni corrigée.

L'inconvénient de cette technique, si on calcule la parité verticale sur un groupe de 8 octets, est l'aggravation de la situation par rapport à la parité simple, sur le nombre de bits redondants utilisés en comparaison du nombre de bits utiles sur lesquels ils portent. Pour 64 bits de données utiles, il faut 16 bits supplémentaires (donc ¼).

Pour une configuration mémoire de 256 Mo, il faudra compter 64Mo pour la parité.

2.7.3 La technique ECC (Error checking and correction).

La technique ECC consiste à ajouter 3 octets de contrôle pour 256 octets de données utiles. Actuellement, nous retrouvons 22 bits ECC pour 2048 bits de données utiles. Ces 22 bits comprennent 16 bits de parité de ligne et 6 bits de parité de colonne.



Les 6 bits de parité verticale sont répartis en deux groupes (2x3bits) : P1, P2 et P4 ainsi que P1', P2' et P4' et les 16 bits de parité horizontale sont répartis eux aussi en deux groupes (2x8bits) : P8, P16, P32, P64, P128, P256, P512, P1024 ainsi que P8', P16', P32', P64', P128', P256', P512', P1024'.

Nous prendrons comme exemple un seul octet où nous allons nous limiter au calcul de la parité verticale.

b7 b6 b5 b4 b3 b2 b1 b0
 1 0 1 1 0 1 1 0

$P1 = 1 + 1 + 0 + 1 = 1$ (résultat du ou exclusif entre les différents bits)
 $P1' = 0 + 1 + 1 + 0 = 0$
 $P2 = 1 + 0 + 0 + 1 = 0$
 $P2' = 1 + 1 + 1 + 0 = 1$
 $P4 = 1 + 0 + 1 + 1 = 1$
 $P4' = 0 + 1 + 1 + 0 = 0$

Supposons une erreur dans la donnée comme suit :

b7 b6 b5 b4 b3 b2 b1 b0
 1 0 1 1 0 1 1 **1**

$P1 = 1 + 1 + 0 + 1 = 1$
 $P1' = 0 + 1 + 1 + **1** = 1$
 $P2 = 1 + 0 + 0 + 1 = 0$
 $P2' = 1 + 1 + 1 + **1** = 0$
 $P4 = 1 + 0 + 1 + 1 = 1$
 $P4' = 0 + 1 + 1 + **1** = 1$

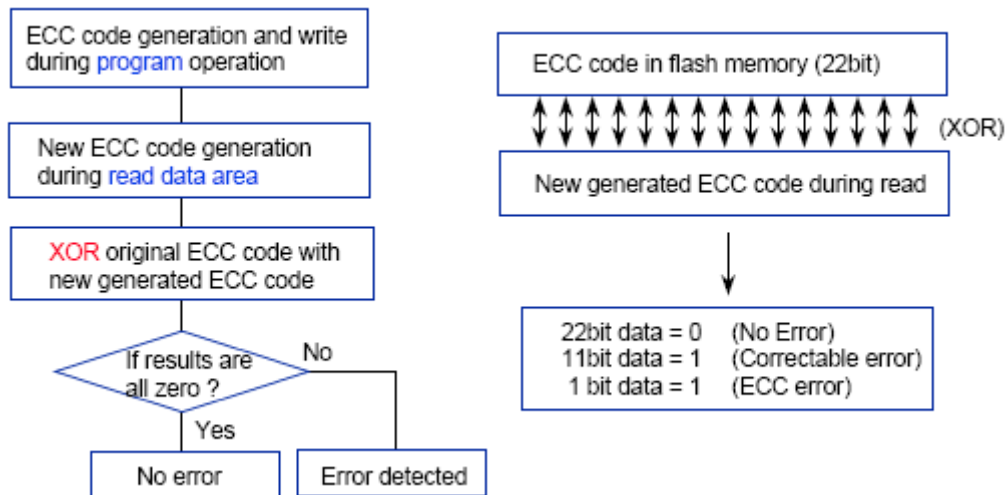
Nous pouvons maintenant effectuer un OU exclusif entre les codes ECC obtenus.

	P4'	P4	P2'	P2	P1'	P1
Donnée d'origine	0	1	1	0	0	1
Donnée erronée	1	1	0	0	1	1
OU exclusif	1	0	1	0	1	0

Pour déterminer s'il y a une erreur, et l'endroit éventuel où cette erreur se trouve, il faudra tenir compte du résultat du OU exclusif en tenant compte des informations suivantes :

- Tous les bits dans le OU exclusif sont à 0. Il n'y a pas d'erreur
- Les 11 bits en P' sont tous à 1.
Dans le XOR, tous les couples de bits (ex : P1 et P1') sont complémentaires.
Il y a une erreur qui est corrigible
- Dans les autres cas, l'erreur n'est pas corrigible.

Voici la séquence de détection de l'erreur sous forme d'un organigramme :



Dans notre exemple, nous remarquons que tous les bits P' sont à 1 et que les couples de bits présentent une complémentarité. Nous avons donc une seule erreur et cette erreur est corrigible. Pour obtenir le numéro de la colonne en erreur, il suffit de se baser sur la configuration binaire des bits.

Nous avons en effet la configuration suivante pour P3 P2 P1 :

	p3	p2	p1
OU exclusif	0	0	0

Cela correspond bien au bit 0 qui est en erreur dans notre donnée. Nous pouvons alors le corriger. Nous pourrions étendre notre exemple en prenant la parité horizontale dont les bits P permettront de fournir le numéro de la ligne où l'erreur se trouve. Nous pouvons obtenir alors l'information suivante :

(P1024, P512, P256, P128, P64, P32, P16, P8, P4, P2, P1)

P1024, P512, P256, P128, P64, P32, P16, P8 = numéro de la ligne contenant l'erreur
P4, P2, P1 = numéro de la colonne contenant l'erreur

Pour 2048 bits (256 octets), nous utilisons 22 bits supplémentaires. Si nous reprenons notre exemple d'une capacité de mémoire vive de 256Mo, il faudra donc compter 22Mbits (environ 8Mo)

3 Etude de la structure d'un disque dur

3.1 Structure physique.

Tout disque dur est constitué d'un ensemble de plateaux rigides recouverts d'un matériau magnétique caractérisés par leur rémanence. Ces plateaux sont divisés en pistes, sortes de cercles concentriques répartis à intervalles réguliers. Ces pistes sont numérotées de 0 jusque N, nombre variant suivant la capacité du disque dur. La piste la plus extérieure porte toujours le numéro 0.

Chaque piste est divisée en secteurs de tailles égales (à l'image de quartiers de tarte) numérotés de 1 jusque N. Chacun des secteurs permet le stockage d'un nombre identique d'octets: 512 octets.

L'ensemble des pistes de l'ensemble des plateaux forment ce que l'on appelle un cylindre et pour identifier dans ce cylindre la piste d'une des faces d'un plateau donné, on utilisera le numéro de la tête de lecture/enregistrement. On les numérote du plateau supérieur vers le plateau inférieur en commençant par le numéro 0. Cette façon de représenter la structure du disque s'appelle CHS.

En connaissant les caractéristiques du disque dur, on peut facilement en déduire sa capacité:

Capacité = Nombre de cylindres x Nombre de secteurs x Nombre de têtes x 512 octets.

Exemple: 16708 cylindres 16 têtes 63 secteurs

Capacité = 16708 x 16 x 63 = 8 Go

Le nombre de secteurs par piste ainsi que la vitesse du disque dur influent directement sur le débit de transfert des données. Bien que dans les disques durs IDE, la valeur du nombre de secteurs configurés dans le BIOS pour un disque dur donné ne corresponde pas nécessairement à la réalité physique du disque, on pourrait en connaissant ce nombre poser le calcul suivant:

Débit = Nombre de secteurs sur une piste x 512 octets x 7200 tours / 60 secondes (Exprimé en général en Mo/sec et pour une seule face)

Actuellement, le CHS peut être sur 28 bits ou 48 bits (introduit dans ATA6): 16 bits pour le nombre de cylindres (0-65535), 4 bits pour le nombre de têtes(0-15), et 8 bits pour le nombre de secteurs par piste (1-255), correspondant donc à une capacité de 137 GB. Avec 48 bits, la limite est donc de 144 petabytes (144,000,000 gigabytes).

Une autre façon d'accéder à un secteur de 512 octets est d'utiliser le mode LBA (Logic Block Array) qui permet d'accéder par une adresse linéaire à un secteur donné.

3.2 La gestion du système de fichiers.

3.2.1 Introduction.

L'ensemble des données présentes sur le disque va être répartie dans l'ensemble des différents secteurs. Il est important que ces secteurs soient numérotés et que l'on puisse à tout moment retrouver l'ensemble des secteurs sous lesquels une donnée est enregistrée. On retrouve pour cela une table d'allocation de fichier et un répertoire racine.

3.2.2 La table d'allocation de fichiers FAT.

La plus connue et la plus ancienne est appelée FAT (File Allocation Table). On y joint un nombre qui représente le nombre de bits sur lesquels est représenté le numéro d'un cluster, celui-ci étant le regroupement logique d'un certain nombre de secteurs. On parle de FAT12, FAT16 et FAT32.

Une FAT est une table présente sur le disque dur et contenant les numéros de clusters qui sont occupés par des données.

La FAT12 permet de coder un numéro de cluster sur 12 bits c-à-d que l'on peut numéroter au plus 4096 clusters. La FAT16 permet quant à elle de pouvoir coder 65536 clusters.

En sachant qu'un secteur contient 512 octets, on adaptera le nombre de secteurs/cluster en fonction de la taille du disque dur.

Taille du disque jusque	128Mo	256Mo	512Mo	1028Mo	2048Mo
Taille d'un cluster	2Ko	4Ko	8Ko	16Ko	32Ko
Secteurs/cluster	4	8	16	32	64

Exemple: FAT16 65536 entrées, 32 secteurs par cluster.
 $65536 \text{ clusters} \times 32 \text{ secteurs} \times 512 \text{ octets} = 1028\text{Mo}$

Même si un cluster est partiellement occupé par une donnée, dès qu'il est marqué comme occupé dans la table d'allocation de fichier, la totalité de sa place est réservée. On comprend aisément que plus la taille d'un cluster est grande, plus on risque de perdre de la place sur le disque. C'est la raison pour laquelle on n'a pas été plus loin que 64 secteurs/cluster.

Avec l'apparition des disques durs de plus grosses capacités, Microsoft sort en même temps que son système d'exploitation Windows 95 le système FAT32 (version OSR2). Chaque entrée dans la FAT est codée sur 32 bits mais le nombre de clusters est limité à 2^{28} car sur les 32 bits, 4 bits sont réservés, ce qui nous permet d'envisager une taille de disque maximale de $2^{28} * 32\text{ko}$ (taille maximale d'un cluster) = 8To.

Pour un disque dur de 8To, la taille de la FAT est de $2^{28} * 4\text{octets} = 1\text{Go}$. Pour les disques de petites tailles, le nombre d'entrées dans le FAT est réduit à 2M, ce qui nous donne une place occupée de 8Mo ($32\text{Go}/16\text{Ko} = 2\text{M}$). Au-delà des 32Go, la taille des clusters reste identique c-à-d de 32Ko et c'est la taille de la FAT qui augmente. Au niveau des systèmes d'exploitation, les choses ne sont pas aussi simples: les outils de vérification de disques inclus avec Windows 95 et Windows 98 ne peuvent pas analyser de disques dont le système de fichiers FAT32 aurait une FAT supérieure à 16Mo c-à-d 4M d'entrées. 4M de clusters occupant chacun 32Ko nous donne donc un disque dur analysable dont la taille serait de 128Go. Windows 98 permet donc de formater un disque de 8To mais ses outils de vérifications n'admettent que 128Mo. Si nous choisissons Windows 2K, la taille des disques que ce système d'exploitation est capable de gérer sous le système de fichier FAT32 est de 32Go. Même un disque dur de plus de 32 Go formaté sous Windows 98 ne pourrait être monté sous Windows 2K. Sous Windows XP, l'utilitaire de formatage proposé lors de l'installation du système d'exploitation ne permet pas d'envisager des disques de plus de 32Go bien que le système d'exploitation, une fois installé, permet de monter des disques de taille supérieure.

Taille du disque jusque	< 8Go	16Go	32Go	>32Go – 2To
Taille d'un cluster	4Ko	8Ko	16Ko	32Ko
Secteurs/cluster	8	16	32	64

Tout système de fichier FAT32 doit contenir au moins 65527 clusters qui composés chacun de 4Ko nous permettrait d'envisager un disque d'une taille minimale de $65527 * 4\text{ko} = 260\text{Mo}$. Nous ne pouvons donc pas envisager de formater une partition d'une taille inférieure à 260Mo.

Microsoft a malgré tout estimé que pour les disques de taille supérieure à 512Mo, il était préférable de choisir le système de fichier FAT32 pour le gain de place qu'il représente lors du stockage des fichiers. En deçà, il vaut mieux conserver un système de fichiers de type FAT16.

Les différentes valeurs que l'on peut retrouver dans chacune des entrées de la FAT sont reprises dans le tableau suivant:

Code cluster FAT 16 bits	Signification
0000h	Le cluster est libre
FFF0h-FFF6h	Le cluster est réservé
FFF7h	Le cluster est défectueux
FFF8h-FFFFh	Dernier cluster d'un fichier
xxxxh	Prochain cluster d'un fichier.

3.2.3 Le répertoire racine.

Alors que la FAT contient les informations sur les clusters qui sont occupés, le répertoire racine comprend les informations concernant les différents fichiers qui sont stockés sur le disque dur. On retrouvera entre autres: le nom du fichier, l'extension du fichier, les attributs, les différentes dates et heures, la taille du fichier ainsi que le numéro du premier cluster occupé.

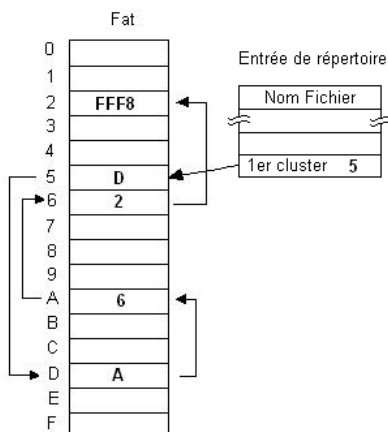
Voici pour la FAT16 la description de l'espace occupé par une entrée dans le répertoire racine:

Adresse	Description	Taille
+00h	Nom du fichier	8 octets
+08h	Extension du fichier	3 octets
+0Bh	Attribut du fichier	1 octet
+0Ch	Réservé	10 octets
+16h	Heure de la dernière modification	1 mot
+18h	Date de la dernière modification	1 mot
+1Ah	Premier cluster du fichier	1 mot
+1Ch	Taille du fichier	1 double mot

La taille du répertoire racine est figée et n'évolue pas en fonction du nombre de fichiers stockés. Pour la FAT16, on retrouve 512 entrées pour les noms représentés sous la convention 8.3. A partir de Windows 95, tout nom représenté sur plus de 8 caractères (maximum 255 caractères) occupe plus d'une entrée et de ce fait le nombre total de fichiers que l'on peut placer dans le répertoire racine se trouve diminué.

Sous le système FAT32, le répertoire racine accepte 65536 entrées avec nombre de fichiers maximum dépendant de la taille de la FAT (au moins 65527 et au plus 2^{28}) fichiers différents sur une même partition (65536 pour le système FAT16).

3.2.4 Lien entre le répertoire racine et la FAT.



Lorsque le système d'exploitation doit charger un fichier en mémoire, l'ensemble des différents clusters vont être accédés suivant l'ordre renseigné par le chaînage. A chaque entrée de la FAT, le cluster correspondant est lu dans la zone de données du disque ainsi que le numéro du cluster suivant et ainsi de suite jusqu'au moment où l'on atteint le dernier cluster identifié par le code FFF8. A force d'enregistrer et d'effacer des fichiers sur votre disque, celui-ci ressemble à un véritable gruyère et les déplacements de têtes que l'on doit effectuer pour lire la totalité des clusters constituant le fichier deviennent importants.

C'est la raison pour laquelle il est important de pouvoir défragmenter le disque pour pouvoir ainsi regrouper les différents clusters d'un même fichier.

3.2.5 Les partitions d'un disque dur (structure MBR).

Pour une question d'organisation de ses données sur un disque dur physique, pour la présence sur votre disque dur de systèmes d'exploitation multiples, chacun disposant de son propre système de fichier mais également parce que un système de fichier donné ne permet toujours la gestion complète du disque du fait que tous les clusters ne peuvent être gérés, il a été rendu possible, au niveau des disques durs, la création de disques virtuels (partitions ou volumes) ayant chacun leur propre système de fichier et leur propre taille.

Les informations caractérisant les différentes partitions sur un disque se trouvent au début du disque dur dans un secteur réservé à cet effet que l'on appelle secteur d'amorçage maître (Master Boot Record). Le fait que tout secteur ne fasse que 512 octets, le nombre d'informations que l'on peut y stocker est limité. En voici la structure :

Adresse	Contenu	Taille
+00h	Code de partition	Code
+1BEh	1 ^{ière} entrée dans la table de partition	16 octets
+1CEh	2 ^{ième} entrée dans la table de partition	16 octets
+1DEh	3 ^{ième} entrée dans la table de partition	16 octets
+1EEh	4 ^{ième} entrée dans la table de partition	16 octets
+1FEh	Code d'identification de ce secteur	2 octets

La première conclusion que l'on peut tirer de ce tableau est que le nombre de partitions est limité à quatre. C'est une limitation physique quel que soit l'utilitaire de partitionnement utilisé. Si on analyse les 16 octets de chacune des entrées, nous retrouverons une structure reprenant les informations suivantes :

- Etat de la partition : non active ou partition amorçable
- Emplacement de la partition : début et fin – tête, secteur et cylindre pour chaque
- Type de partition : notamment primaire, étendue ou non utilisée

Théoriquement, il est possible de créer jusque quatre partitions primaires ou trois partitions primaires et une seule partition étendue étant elle-même subdivisée en disques logiques mais dans la réalité, le nombre de partitions que l'on peut créer dépend de l'outil utilisé.

Une partition étendue est unique et ne peut pas être amorçable, c a d contenir un système d'exploitation sur lequel il serait possible d'amorcer (de démarrer) votre ordinateur.

Une seule partition primaire (sur les trois ou quatre éventuels) peut être rendue active. C'est cette partition active qui sera détectée par le BIOS et qui permettra alors de pouvoir démarrer votre ordinateur sur le système d'exploitation installé sur votre disque dur.

La première chose à faire lorsque vous désirez installer un système d'exploitation sur un disque dur neuf est de partitionner le disque (exemple : utilitaire FDISK). Vous choisissez le nombre de partitions (1 à 4) et leur taille respective. La structure de ce MBR est indépendante du choix de la structure de fichiers choisie, le MBR devant être accédé par le BIOS.

Lorsque le BIOS a trouvé sur votre disque dur la partition amorçable, il effectue la lecture du secteur d'amorçage de la partition. Il y a donc sur un disque dur 1 seul MBR mais autant de secteurs

d'amorçage que de partitions primaires créées. Le secteur d'amorçage du volume est créé lors du formatage haut niveau effectué avec l'outil de votre système d'exploitation. Ce même outil sera à la base du choix du système de fichier utilisé. Voici la structure d'un volume qui serait formaté avec le système de fichier FAT :

Secteur d'amorçage (Boot record) – 512 octets (1 seul secteur)
Première table d'allocation de fichiers
Une ou plusieurs copies de la FAT
Répertoire racine avec nom de volume
Zone de données pour les fichiers et sous répertoires

En dehors des informations concernant la taille et la structure du volume, le secteur d'amorçage comprend ce que l'on appelle le BootStrap Loader qui permet en fait de charger le système d'exploitation.

Adresse	Contenu	Taille
+00H	Instruction de saut à la routine d'amorçage (JUMP)	3 octets
+03H-1CH	Caractéristiques du volume	
+1EH-1FFH	Routine d'amorçage	482 octets

Cette routine d'amorçage doit comprendre les routines de base permettant l'accès au système de fichier pour permettre le chargement complet du système d'exploitation.

3.2.6 Les partitions d'un disque dur (structure GPT).

3.2.7 La table d'allocation de fichiers MFT.

La Master File Table (MFT) est propre au système de fichier NTFS. Même sous NTFS, le cluster représente la plus petite organisation sur le disque capable de contenir un fichier. NTFS a toujours été le système de fichier lié aux systèmes d'exploitation Microsoft de type NT. Pour les versions antérieures à NT 3.51, le système d'exploitation ne supporte pas la compression NTFS et la taille par défaut des clusters lors du formatage peut dépasser 4Ko. Nous retrouverons les correspondances suivantes entre taille de disque dur et taille des clusters:

Taille du disque jusque	4Go-8Go	8Go-16Go	16Go-32Go	>32Go
Taille d'un cluster	8Ko	16Ko	32Ko	64Ko

A partir de la version NT3.51, lié au fait que la compression NTFS n'est pas possible sur des disques dont la taille des clusters serait supérieure à 4Ko, l'utilitaire de formatage n'utilisera jamais une valeur par défaut supérieure à 4Ko (une valeur supérieure peut être forcée manuellement). Nous obtenons alors le tableau suivant:

Taille du disque jusque	<512Mo	512Mo-1024Mo	1Go-2Go	>2Go
Taille d'un cluster	512octets	1Ko	2Ko	4Ko

La MFT est l'endroit où les informations relatives aux fichiers et répertoires présents dans un volume NTFS sont placées. La MFT peut être vue comme étant une table d'une base de données relationnelle contenant des attributs variés sur des fichiers. La MFT peut être vue comme étant analogue à la FAT mais elle ne se limite pas à une liste de clusters disponibles ou utilisés. La taille de chaque enregistrement dans la MFT correspond à la taille d'un cluster mais doit au minimum avoir une taille

de 512 octets tout en ne dépassant pas 4096 octets. Nous pourrions retrouver dans certaines sources que chaque enregistrement dans la MFT est limité aux deux valeurs 1024 et 2048.

Les enregistrements pointant sur les fichiers et répertoires possèdent les attributs suivants:

- **Header (H):** L'en-tête est une donnée permettant au système de fichier NTFS d'assurer la gestion du répertoire. Il inclut des numéros de séquence utilisés en interne par NTFS et des pointeurs vers les attributs des autres fichiers et l'espace libre dans l'enregistrement
- **Standard Information Attribute (SI):** Cet attribut contient les informations standard stockées sur les fichiers et répertoires. Il inclut des propriétés fondamentales telles que les dates/heures de création, de modification et de dernier accès mais aussi les attributs standard de type FAT tels que par exemple Read-Only, Hidden etc...
- **File Name Attribute (FN):** Cet attribut stocke le nom associé avec le fichier. Notons que le fichier peut avoir plusieurs attributs de noms permettant de renseigner un nom court de type MS-DOS mais aussi des liens durs de type POSIX.
- **Data (Data) Attribute:** Cet attribut stocke le contenu actuel du fichier. Nous évoquerons le stockage des petits fichiers par la suite dans le cours.
- **Security Descriptor (SD) Attribute:** Cet attribut contient des informations de sécurité permettant de contrôler les accès aux fichiers. (Access Control List ACL). Nous pouvons donc dire que dans ce type de système de fichier, il est possible de déplacer un fichier dans le même volume sans pour autant perdre les attributs de sécurité qui lui sont liés.

Pour les petits fichiers, nous retrouverons la structure suivante:



H - En tête

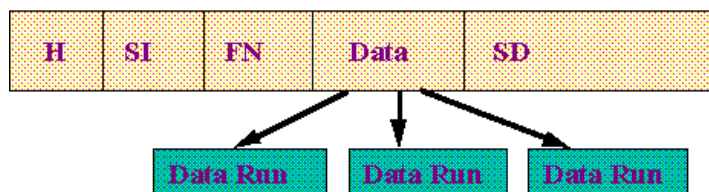
SI- Informations standard

FN - Nom du fichier

Data - Données résidentes

SD - Descripteur de sécurité

Pour les plus grands fichiers qui ne peuvent être contenus dans le seul enregistrement de la table MFT, nous retrouverons alors la structure suivante:



H - En tête

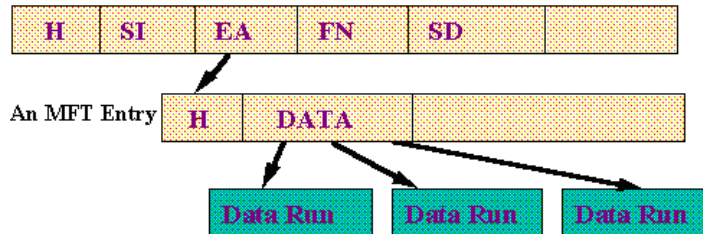
SI- Informations standard

FN - Noms du fichier

Data - Données non résidentes

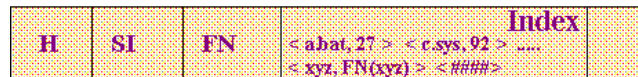
SD - Descripteur de sécurité

Dans ce cas de figure, la donnée contient le numéro de cluster virtuel du premier cluster contenant chaque partie du fichier ainsi que le nombre de clusters contigus de chacun de ces blocs. Si un fichier est trop grand de sorte que même les attributs de données ne peuvent tenir dans l'enregistrement, l'attribut de données deviendra lui-même non résident. Nous retrouverons alors la structure suivante:



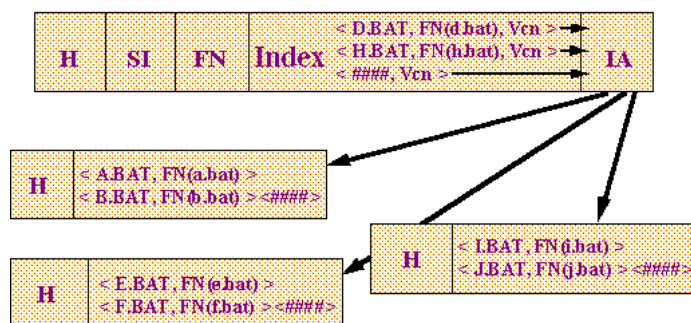
- H - Header** **SI- Standard Information Attribute**
- EA - External Attribute** **FN - File Name Attribute**
- SD - Security Descriptor Attribute**
- Data - Non-Resident Form**

Les répertoires correspondent également à une entrée dans la MFT en reprenant un attribut sous forme d'un index. Cet index permet de trier les fichiers sur un attribut spécifique et de pouvoir ainsi les retrouver plus rapidement. Si le nombre de fichiers dans le répertoire est suffisamment petit, l'index peut être résident dans l'enregistrement de la MFT, on parle alors de Small Index. Celui-ci contient la valeur de l'attribut devant être utilisé comme index (par défaut le nom) mais aussi le numéro de fichier c-à-d le numéro de l'enregistrement dans la MFT correspondant au fichier 'FN(xyz)'.



- H - En tête**
- SI- Informations standard**
- FN - Noms du fichier**
- Index des fichiers dans la MFT**
- ##### - Indication de fin**

Lorsque le nombre de fichiers dans un répertoire augmente, la structure d'index peut devenir non résident. Cependant, la racine de l'index doit toujours rester résident dans l'enregistrement racine du répertoire. Dans ce cas de figure, on parle d'index large (large index) qui est lié à l'enregistrement du répertoire par un attribut d'allocation d'index.

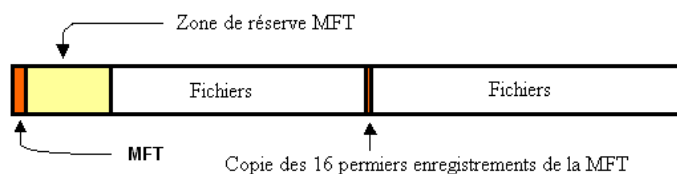


Un index large contient <attribut, Numéro de fichier, Numéro de cluster virtuel> où la valeur de l'attribut est le nom du fichier. En se basant sur le diagramme représenté ci avant, nous retrouvons comme première entrée dans l'index <D.BAT, FN(d.bat), Vcn> pointant dans le buffer Btree dans l'attribut d'allocation d'index qui référence <A.BAT, FN(a.bat)> <B.BAT, FN(b.bat)> <#####>

Par défaut, NTFS réserve une zone correspondant à 12.5% de la taille totale du volume et ne permet pas l'écriture des données utilisateurs dans cette zone. Cependant, lorsque le nombre de fichiers augmente, cette zone peut augmenter au-delà de cette réserve et devenir alors fragmentée. De plus, lorsque des fichiers sont effacés, NTFS n'utilise pas toujours cet espace pour stocker de nouveaux fichiers, il marque juste les entrées comme étant effacées et alloue de nouvelles entrées pour les nouveaux fichiers en forçant la MFT à devenir fragmentée amenant ainsi une perte de performance. Nous pouvons définir la zone de réservation MFT à travers les paramètres renseignés dans la base de registre:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem
 Value *NtfsMftZoneReservation* of DWORD type (1 to 4)

Cette entrée permet de spécifier une zone MFT pour tout volume nouvellement créé (12.5%, 25%, 37.5%, 50%). Si la zone réservée aux fichiers devenait trop petite, NTFS permet que la zone de réserve MFT soit utilisée pour le stockage des fichiers. La structure générale du disque dur est la suivante:



Dans la MFT, les 16 premiers enregistrements sont réservés et correspondent aux valeurs suivantes:

\$MFT	MFT elle même
\$MFT miroir	Copie des 16 premiers enregistrements placée au milieu du disque dur
\$LogFile	Fichier contenant un journal des opérations sur le disque
\$AttrDef	Liste des attributs standard des fichiers sur le volume
\$.	Répertoire racine
\$Bitmap	Chaque cluster est associé à un bit permettant d'indiquer si il est libre ou occupé
\$Boot	Secteur d'amorçage
\$Quota	Fichier où les droits des utilisateurs sur l'espace disque utilisable sont enregistrés (valable à partir de NTFS 5).
\$Upcase	Table assurant la correspondance des noms des fichiers stockés sous NTFS dans le format Unicode.

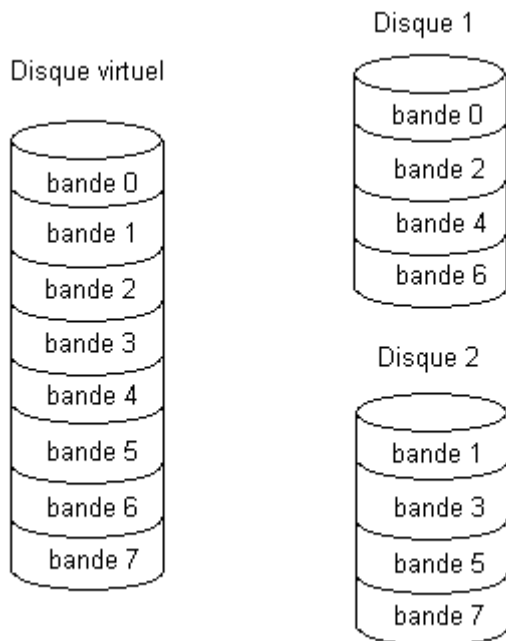
4 Technique des disques en grappe RAID.

4.1 Introduction

C'est en 1987 à l'université de Berkeley qu'un groupe de chercheurs étudie la possibilité de pouvoir faire reconnaître un groupe d'un ou plusieurs disques, appelé une grappe, comme une seule entité. Bien que présentant de bonnes performances, un inconvénient était sans doute le manque de fiabilité puisque un problème matériel sur un des disques de la grappe avait comme conséquence la perte de toutes les données dans la grappe. En 1988, ces mêmes chercheurs intègrent une architecture redondante pour apporter une tolérance de panne à ces grappes. Sont nées les technologies RAID allant du niveau 1 au niveau 5 mais d'autres niveaux vont ensuite naître notamment comme une association des niveaux précédents qui seront abordés dans la suite de ce paragraphe.

RAID est l'abréviation de *Redundant Arrays of Inexpensive Disks*. Au lieu d'imaginer des disques de très grande capacité à coût élevé, on préfère travailler avec une grappe de disques durs de petite taille et moins onéreux. Nous n'aborderons dans le paragraphe que les technologies RAID standard et celles les plus rencontrées.

4.2 RAID 0 (appelé striping ou volume agrégé par bande).

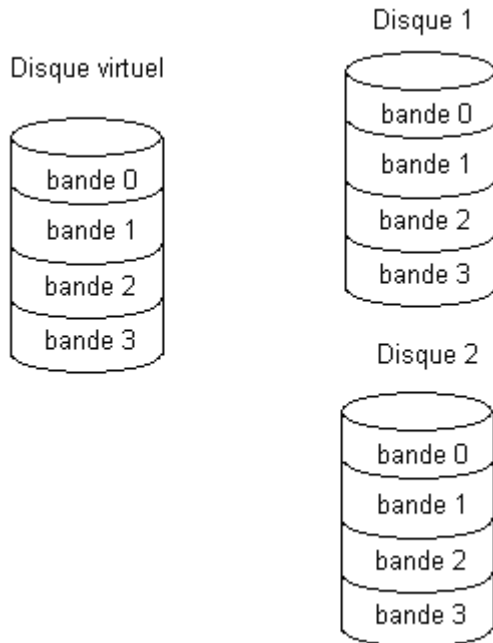


Bien que portant le nom de RAID avec notamment la notion de redondance, le RAID 0 n'inclut aucune tolérance de panne. Les données sont découpées en bandes logiques et réparties de manière entrelacée sur l'ensemble des disques. Une bande peut être associée à un bloc, un secteur physique ou toute autre unité. De ce fait, si un des disques durs est défectueux, les données de la grappe entière sont perdues.

Si un fichier de grande taille est ainsi réparti sur plusieurs disques, le taux de transfert lors d'un accès s'en trouvera accéléré du fait qu'en parallèle, chacun des disques de la grappe comprenant une partie du fichier sera accédé.

Le nombre de requêtes d'E/S sera également augmenté du fait que chaque disque de la grappe peut être accédé individuellement et donc, pour autant par exemple que plusieurs fichiers différents se trouvent sur des disques différents, ceux-ci peuvent être accédés en parallèle.

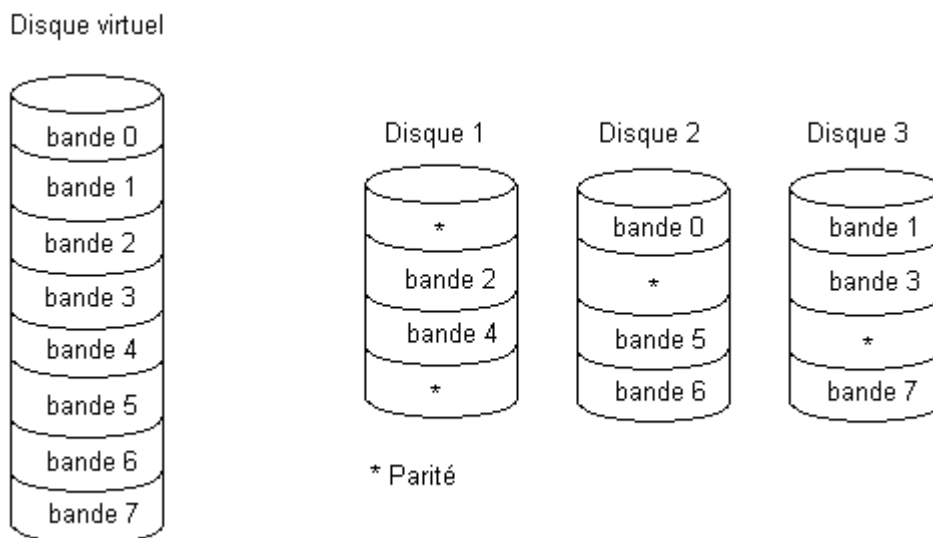
4.3 RAID 1 (appelé mirroring).



Dans ce mode de fonctionnement, les données sont dupliquées sur deux disques. Le RAID 1 présente donc une tolérance à la panne mais cette tolérance a un coût puisque pour un disque de données utiles, il faut un disque pour la redondance.

L'accès en lecture est plus rapide du fait que les deux disques de la grappe peuvent être accédés en parallèle. En écriture, nous n'avons aucune amélioration.

4.4 RAID 5 (stripping avec parité alternée)



Il faut au minimum trois disques pour pouvoir envisager cette technologie. Nous retrouvons une tolérance de panne : si un des disques est défectueux, les données manquantes sont reconstruites sur base de la parité ajoutée. Si deux disques sont défectueux dans la grappe, l'ensemble des données de la grappe sont perdues. Les performances en lecture sont plus élevées car les données réparties sur plusieurs disques peuvent être accédées en parallèle. Les performances en écriture sont moins bonnes que dans le cas du RAID1 car il est nécessaire de calculer la parité. Par contre si pour le RAID1, nous avons uniquement 50% de l'espace total disque occupé par les données utiles, dans le RAID5, plus le nombre de disques est important, plus le pourcentage de données utiles par rapport à la capacité totale des disques de la grappe sera élevé.

Avec trois disques, nous avons un pourcentage de 66% (2/3)

Avec quatre disques, nous avons un pourcentage de 75% (3/4)

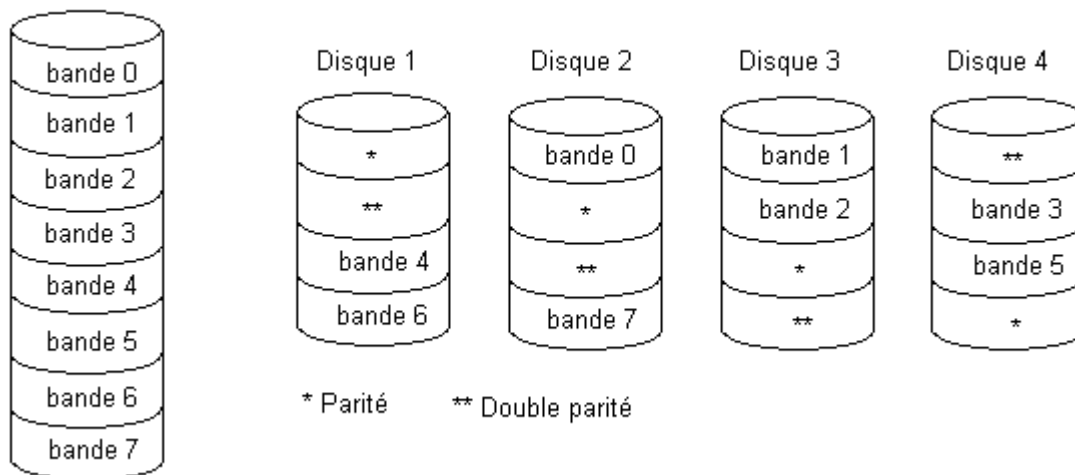
La calcul de la parité est assez simple et repose sur le principe du OU exclusif.

Exemple : * = bande 0 (+) bande 1

Le RAID4 proposait un disque de parité dédié mais cette technique présente un goulot d'étranglement du fait que lors d'accès parallèle sur l'ensemble des disques de données, la parité se retrouve concentrée sur le même disque et de ce fait réduit les performances générales du système.

4.5 RAID 6 (striping avec double parité alternée)

Disque virtuel



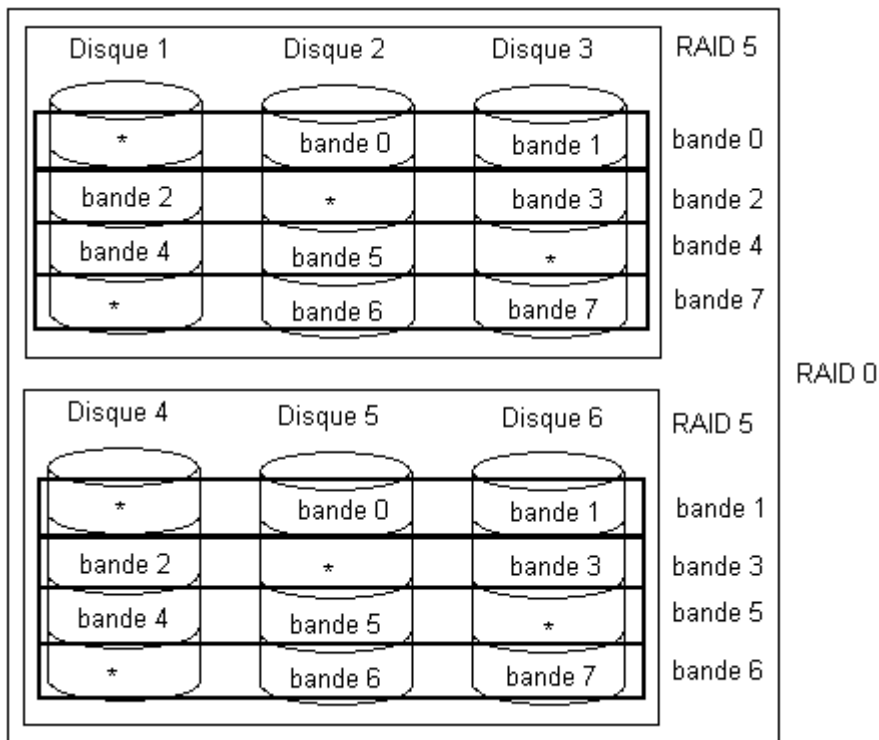
Si la parité * correspond à celle calculée dans le RAID5, le deuxième bloc ** correspond à un algorithme plus compliqué non abordé dans le cadre de ce cours.

Il faut donc quatre disques pour envisager cette technologie. Nous retrouvons une tolérance de panne où le RAID 6 peut accepter sans perte de données la défaillance de deux disques durs.

Cette tolérance à la panne meilleure a un coût en termes de place occupée par les deux blocs de parité mais aussi en termes de performance lors des opérations d'écriture car deux blocs de parité doivent être calculés.

4.6 RAID X0 (Niveaux de RAID combinés)

Nous pouvons également retrouver les combinaisons RAID X1. A titre d'exemple, nous nous limiterons à aborder dans ce paragraphe le mode combiné 50. On peut généralement considérer que le premier chiffre indique le niveau de raid des "grappes" et que le second indique le niveau de raid global



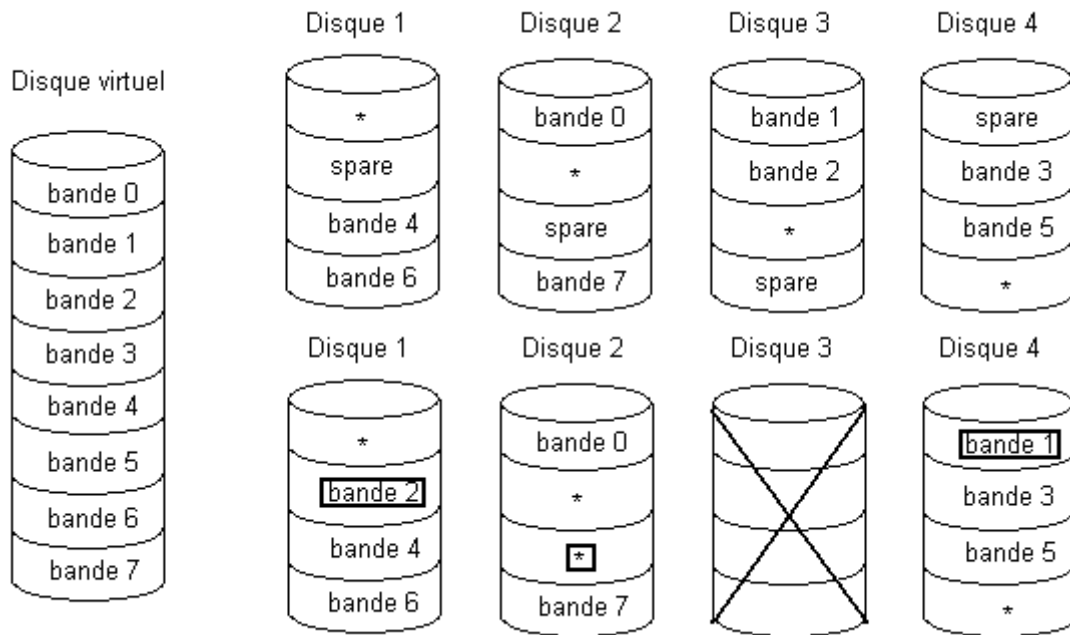
4.7 Les disques de rechange (SPARE)

Nous pouvons affecter à une grappe un disque de rechange appelé disque de spare ou hotspare. Ce disque n'est pas utilisé mais servira de rechange immédiate sans intervention manuelle dans le cas où le disque d'une grappe devient défectueux et pour autant que la technique utilisée offre une tolérance à la panne. Le contenu du disque est alors reconstruit à partir des données présentes sur les autres disques, ce qui peut, en fonction de la taille du disque dur durer plusieurs heures. Le disque défectueux peut alors être remplacé par un nouveau disque qui deviendra le disque de rechange.

Si nous nous basons sur la technologie RAID5 qui réclame au minimum 3 disques et que nous prévoyons du spare, il faudra donc compter au minimum 4 disques. Nous pouvons en effet prévoir plusieurs disques de rechange.

Si nous envisageons certaines évolutions des technologies abordées précédemment, nous retrouvons la technologie RAID 5E qui envisagera la répartition de cette zone de rechange sur l'ensemble des disques de la grappe. De cette façon, le disque de réserve est quand même utilisé comme disque utile et permet d'améliorer ainsi les performances. Il faut prévoir dans cette technologie au minimum 4 disques.

Nous envisageons dans le graphique suivant, la façon dont ces blocs de rechange sont répartis et le cas de figure où le disque trois devient défectueux avec la reconstruction des données sur les autres disques.



4.8 La cache en écriture.

Comme nous avons pu le constater précédemment, les technologies présentant une tolérance à la panne offrent de bonnes performances en lecture mais sont pénalisées par des performances moins bonnes en écriture du fait du calcul de la parité. Pour palier à ce problème, les contrôleurs RAID sont équipés, en plus d'une mémoire cache pour les lectures, d'une mémoire cache additionnelle pour les écritures.

Pour les écritures, il faudra veiller à ce que le contrôleur soit placé sur une alimentation non interruptible pour plusieurs raisons :

Toute interruption de l'alimentation fera perdre des données présentes dans la mémoire cache et non transférées sur le disque.

Si nous envisageons l'écriture en parallèle sur plusieurs disques de la grappe, il se peut que des données aient été transférées sur certains disques plus rapides ou moins chargés alors que l'écriture n'aura été effectuée sur d'autres. Ce peut être le cas pour la parité qui avant son transfert vers le disque doit être calculée.

5 Etude des bus d'extension.

5.1 Le bus ISA (Industry Standard Architecture)

ISA (Industry Standard Architecture) : Ce bus apparaît avec les premiers ordinateurs de type PC mis sur le marché par IBM, on parle alors de pc XT (8088, 8086). Au départ il travaille sur un bus de données de 8 bits avec une vitesse de 4,77 MHz. Parmi les signaux présents sur le connecteur, nous retrouvons:

- 20 bits pour les adresses, (soit 1 Mo de mémoire adressable)
- 8 bits pour les données, vitesse de transfert théorique de 4,7 Mo/s en réalité de 1 Mo/s

- 8 lignes d'interruptions, (IRQ) moyen pour prévenir le système de l'appel d'un périphérique afin d'éviter les conflits.
- 4 DMA (Direct Memory Access), circuit qui effectue les transferts d'information direct entre un périphérique et la mémoire sans passer par le processeur.

Avec l'apparition des pc AT (80286) équipés de processeur 16 bits, le bus ISA passe à 16 bits en recevant une extension de son connecteur 8 bits natif permettant ainsi une compatibilité avec les cartes 8 bits. La fréquence est portée également à 6Mhz et ensuite 8Mhz, l'adressage sur 24 bits permet de pouvoir envisager l'accès à 16Mo de mémoire et le nombre d'interruptions atteint 16 IRQ du fait de la présence de deux contrôleurs d'interruptions sur les cartes mère. Nous pouvons donc envisager des vitesses de transfert de 16Mo/sec.

Dès la sortie des microprocesseurs 32 bits présentant un bus de données de 32 bits, il devient nécessaire d'évoluer vers un nouveau bus d'extension admettant plus de signaux et une cadence d'horloge plus importante. C'est à ce moment que vont apparaître sur le marché plusieurs systèmes essayant de s'imposer comme standard.

5.2 Le bus MCA (Micro Channel Architecture)

En 1987, IBM lance ses ordinateurs de type ps2 (qui vont donner leur nom aux connecteurs utilisés pour l'interconnexion de la souris et du clavier). Ces ordinateurs proposent un bus d'une largeur de 32 bits cadencé à 10Mhz c-à-d permettant des vitesses de transfert de 40Mo/sec. Ce type de bus n'existera que dans ces ordinateurs du fait des points suivants:

- IBM supprime les bus ISA et de ce fait, tout acquéreur de ces ordinateurs doit oublier de pouvoir récupérer toute carte d'extension provenant d'un ancien ordinateur.
- IBM dépose le nom MCA et tout constructeur désirant intégrer cette technologie doit payer une redevance

5.3 Le bus EISA (Extended Industry Standard Architecture)

Une alliance de constructeurs, en réponse au bus MCA, proposent le bus EISA 32 bits sans aucun droits à payer pour son exploitation.

Concurrent mais totalement incompatible avec le MCA !

Ce bus double les contact du bus ISA il garde une fréquence de 8 MHz. L'utilisateur dispose alors de 32 bits d'adresse (4 Go) et les capacités de transfert se font à 8 Mo/s en mode 8 bits 16 Mo/s en mode 16 bits, et 32 Mo/s en 32 bits. De plus il présente des innovations majeures comme le Bus Mastering qui permet aux cartes d'un serveur de communiquer entre elles sans passer par le processeur. Des mécanismes d'arbitrage sont intégrés pour éviter les conflits entre cartes d'autant plus que les lignes d'interruption peuvent être partagées. Les interruptions sont configurées par logiciel. Le bus EISA accepte aussi bien les cartes 8, 16, 32 bits, mais ce bus qui est conçu pour supporter ces trois formats de données différents et assurer les liaisons entre cartes différentes est d'une technologie complexe et chère ce qui va le limiter aux serveurs de réseau. Beaucoup d'ordinateurs basés sur des processeurs 386 et 486 restent équipés uniquement d'un bus d'extension ISA 16bits. Un des périphériques le plus gourmand en vitesse de transfert est la carte graphique et de ce côté-là, la technologie ISA ne répond plus aux besoins. A cette époque le bus système de la carte mère est cadencé par une horloge pouvant aller de 25MHz à 50Mhz avec un bus de données de 32 bits alors que ISA ne propose que 16bits et 8Mhz. En 1989, une association de constructeurs de cartes graphiques propose une nouvelle technologie, il s'agit du VLB (Vesa Local Bus), VESA étant l'abréviation de Video Electronics Standards, Architecture. L'avantage du bus EISA (et du bus MCA) est aussi la facilité d'installation et de configuration par logiciels des cartes installées dans les slots d'extension de ce bus; le bus pouvait détecter automatiquement les paramètres des cartes ajoutées.

Quatre points définissent l'identification et la configuration

1. Chaque carte doit implémenter 4 octets (ID register) à la position 0xnC80 (n est le numéro du slot EISA de 1 à 0xA). Ces registres définissent le fabricant, le type de dispositif, la version.
2. Les concepteurs peuvent utiliser 124 octets (de 0xnC84 à 0xnCFF) pour implémenter des registres de configurations de la carte. Comme par exemple, le numéro de canal DMA que la carte utilise. Un autre pour les options des IRQ. L'enregistrement de ces valeurs dans les registres est équivalent aux jumpers ou DIP des cartes ISA.
3. Il y a un fichier script qui contient la liste des ressources de la carte, qui définit l'utilisation des divers registres et leurs positions. Ce fichier est écrit en un langage standard EISA et son nom contient le ID de la carte. Ce fichier est livré avec la carte par fabricant.
4. Un programme de configuration EISA est lancé lors du démarrage du système. Ce programme scrute les slots EISA et cherche les cartes dans les emplacements libres. Si une carte est trouvée, il utilise le registre ID pour composer le nom du script et interroge la disquette si elle contient ce fichier. Si le fichier est présent le programme assigne les ressources mémoire au slot et copie les valeurs dans la CMOS (non-volatile) associée au slot afin de ne pas redemander ces informations à chaque redémarrage. Sous Windows 2000 on peut accéder directement au bus EISA par le HAL (HalGetBusData et HalSetBusData).

5.4 Le bus local VLB (Vesa Local Bus)

Du point de vue matériel, le bus VLB se présentait sous forme d'un slot supplémentaire souvent de couleur brun dans le prolongement du connecteur ISA. Les autres connecteurs ISA 16 bits étaient conservés. Ce bus était surtout réservé à la carte graphique, mais comme il était possible d'avoir 3 connecteurs VESA, on pouvait aussi mettre une carte d'interface pour disque dur et sorties séries ou parallèle. La limitation du nombre de connecteurs était lié au fait que ce bus d'extension venait se greffer directement sur le bus système en fonctionnant à la cadence de l'horloge du bus système de la carte mère. Cette technologie disparaîtra en même temps que le processeur auquel elle est liée majoritairement: le 486. La raison en est simple: avec la sortie du processeur Pentium, Intel lance en 1991 le projet d'un nouveau bus d'extension: le bus PCI.

5.5 Le bus PCI (Peripheral Component Interconnect)

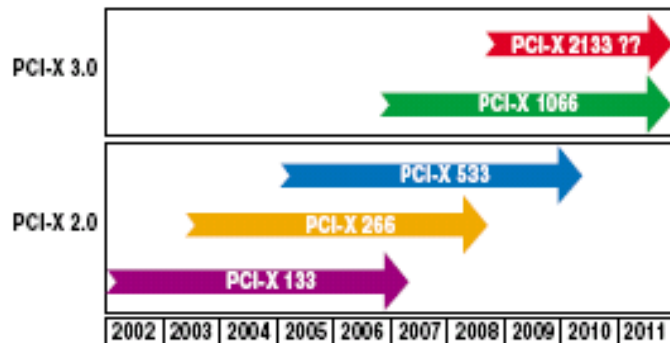
Pour développer ce projet d'un nouveau bus d'extension, Intel prévoit les aspects suivants:

- le bus doit permettre à toute carte d'extension de pouvoir assurer des transferts vers d'autres cartes sans devoir par cela dépendre du processeur. Ce bus intègre le Bus Mastering.
- Le bus doit comprendre un bus de données de 32 bits et fonctionner à la cadence du bus système qui pour le processeur Pentium est de 33Mhz.
- Alors que jusqu'à présent la configuration des cartes d'extension au niveau des ressources telles que choix de l'interruption, du canal de DMA, des ports E/S demande des connaissances techniques, le bus PCI doit permettre la configuration automatique des cartes. On parle de Plug and Play.
- Le bus PCI doit permettre le partage des interruptions.

A l'origine, le bus PCI gère 32 bits d'adresses et de données à une vitesse de 33 MHz soit une vitesse de transfert d'informations à $33 \times 4 = 132$ Mo/s en théorie. Sa deuxième version (PCI

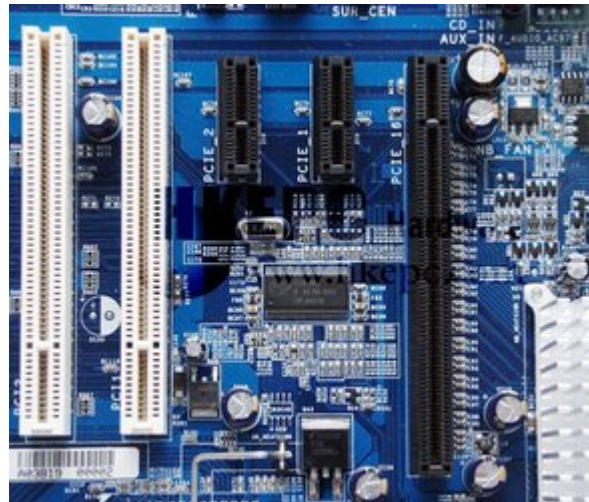
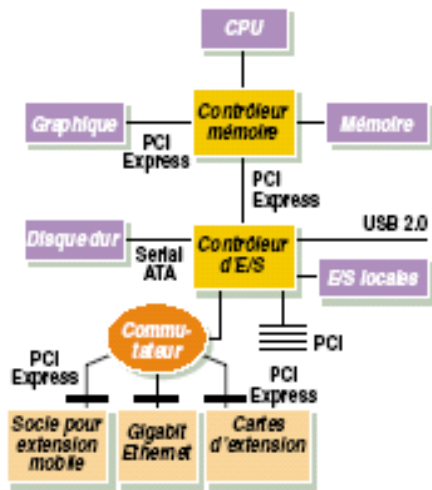
2.0) pour processeur Pentium voit sa largeur étendue à 64 bits pour une fréquence maintenue à 33 MHz soit une vitesse de transfert de 230 Mo/s. La version PCI 2.1 adopte une fréquence de 66 MHz (460 Mo/s). Ce bus ne se substitue cependant pas au bus ISA qui reste l'architecture de base, c'est un complément qui cohabite par l'intermédiaire d'un pont (Host Bridge) et propose jusqu'à 5 connecteurs blancs.

De nouvelles spécifications PCI font leur apparition, PCI-X, cadencé à 133 MHz sur 64 bits (débit de 1 Go/s, MiniPCI pour portable, PCI Low Profile pour faire des cartes moins encombrantes. Nous pouvons établir l'évolution prédictive pour le PCI dans les années suivantes:



La norme PCI-X 266 envisage l'utilisation des deux flancs d'horloge pour le transfert des données tandis que la norme PCI-X 533 envisage le transfert de quatre données par cycle d'horloge ce qui nous permet d'envisager avec un bus de données de 64bits un débit de $8 \times 133 \times 4 = 4.2 \text{ Go/sec}$.

Le PCI Express sur lequel travaillent Intel et ATI, annoncé pour 2004. Ses spécifications ont été entérinées en juillet 2002 par le groupe de réflexion PCI-SIG. Il doit offrir dans un premier temps un taux de transfert théorique de 4 Go/s (2.5Gb/sec par canal, PCI Express 16x) mais le débit est adaptable au besoin des matériels connectés. Ce débit peut être dupliqué sur 2, 4, 8, 12, 16, 32 voies, atteignant dans ce dernier cas un débit record de 10Go/s. Il faudra donc sur les cartes mères des connecteurs de différentes tailles. Une carte réseau à 1 Gbits/s s'installera sur un emplacement PCI express à une voie. Si la technologie PCI Express proposée au départ par Intel sous la dénomination 3GIO devait percer, cela signifierait la disparition du bus AGP des cartes mères. Il est important de signaler que le bus PCI Express n'est pas compatible avec la norme actuelle PCI d'un point de vue matériel mais bien d'un point de vue logiciel, c'est la raison pour laquelle Intel a déjà prévu des passerelles vers la technologie PCI-X et PCI-X et PCI parallèle. En juin 2002, un groupe de travail s'est en effet constitué pour élaborer des spécifications d'interopérabilité entre PCI-X et PCI Express. Fin 2003, Intel a déjà présenté ses deux nouveaux chipsets sous le nom de « Lindenhurst » pour les serveurs et « Tumwater » pour les stations de travail ainsi qu'une passerelle (Bridge) PCI Express vers PCI/PCI-X 1.0 qui permet aux adaptateurs et aux cartes d'extension existants de fonctionner sur les plates-formes basées sur la technologie PCI Express, il s'agit de l'Intel 41210. Nous pouvons dès lors envisager une architecture qui pourrait être la suivante:



Sur l'image de droite, nous retrouvons sur une carte mère les connecteurs PCI traditionnels ainsi que deux connecteurs PCI Express 1x et 1 connecteur PCI express 16x.

PCI Express modifie également les habitudes d'installation. Des cartes d'extension PCI Express pourront être ajoutées sous tension, sans redémarrage du système. Des cartes au format Mini PCI Express (3 cm de large) viendront remplacer les modules d'extension de type PC Card (8,5 cm de large), exploités actuellement par les ordinateurs portables. En terme de coût de fabrication, l'architecture PCI Express serait, selon les dires de ses concepteurs, moins coûteuse à implémenter que l'interface PCI.

A titre de comparaison, la bande passante maximale des bus PCI, PCI-X 2.0 et AGP 8x est respectivement de 1 Go/s (64 bits à 133 MHz), 4,3 Go/s et 2 Go/s. En revanche, le bus HyperTransport - qui est la technologie d'interconnexion promue notamment par AMD, Nvidia et Transmeta - atteint un débit maximal de 12,8 Go/s alors que la version 2.0 du bus devrait atteindre 40 Go/s !

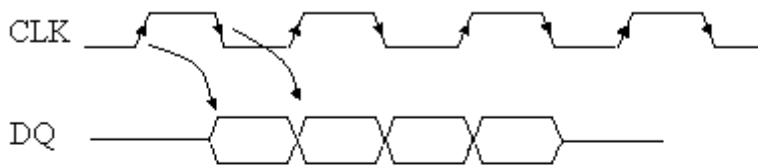
5.6 Le bus AGP (Accelerated Graphic Port)

En 1996, Intel de concert avec Microsoft et les principaux fabricants de cartes graphiques définit le bus AGP. En effet, malgré l'évolution rapide des cadences d'horloge sur le bus système des microprocesseurs, le bus PCI reste en arrière avec ses 33Mhz et les cartes graphiques ont besoin d'un canal de liaison avec la mémoire vive présentant une bande passante de plus en plus élevée pour le transfert des textures. A cette époque, du fait de la capacité mémoire faible des cartes vidéo, les textures sont mémorisées dans la mémoire vive. Un tel intérêt est peut être à revoir du fait de l'évolution technologique des mémoires permettant une capacité élevée même sur ce type de carte. Le bus d'extension AGP travaille sur 32 bits avec une cadence de 66Mhz. On parle de la norme AGP 1x qui permet des vitesses de transfert de $66 \times 4 = 264 \text{ Mo/sec}$. Si les mémoires SDRAM subissent une évolution en adoptant le type de transfert DDR (double Data Rate), AGP connaît le même sort sous la norme AGP 2x. Les transferts se font à la fois sur le flanc montant et sur le flanc descendant de l'horloge permettant alors tout en conservant une horloge de 66Mhz, d'atteindre une vitesse de transfert de $66 \times 4 \times 2 = 528 \text{ Mo/sec}$.

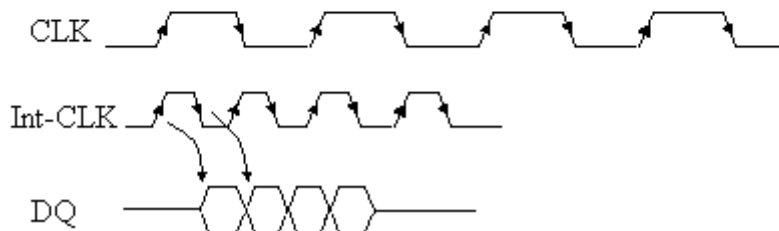
AGP 4x propose de quadrupler la vitesse de transfert en travaillant toujours avec la même horloge cadencée à 66Mhz. Nous retrouvons dans les schémas suivants, le principe du DDR et du QDR (quadruple data rate). Nous obtenons alors pour cette norme une vitesse de transfert de

$66 * 4 * 4 = 1,05 \text{Go/sec}$ (il faut naturellement que la mémoire vive puisse suivre cette vitesse pour tirer profit de cette technologie).

DDR (Double Data Rate: utilisation du flanc montant et descendant)

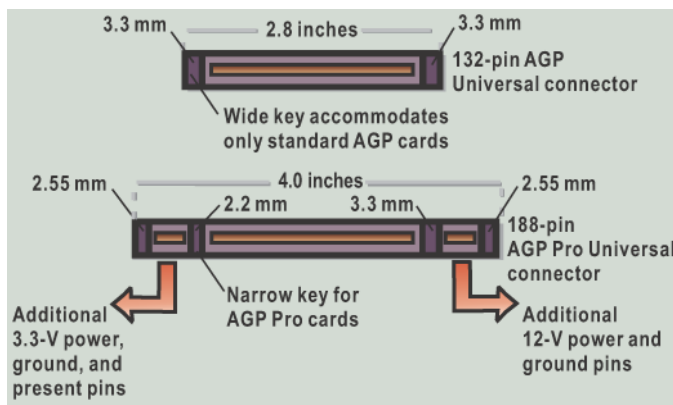


QDR (Quadruple Data Rate)

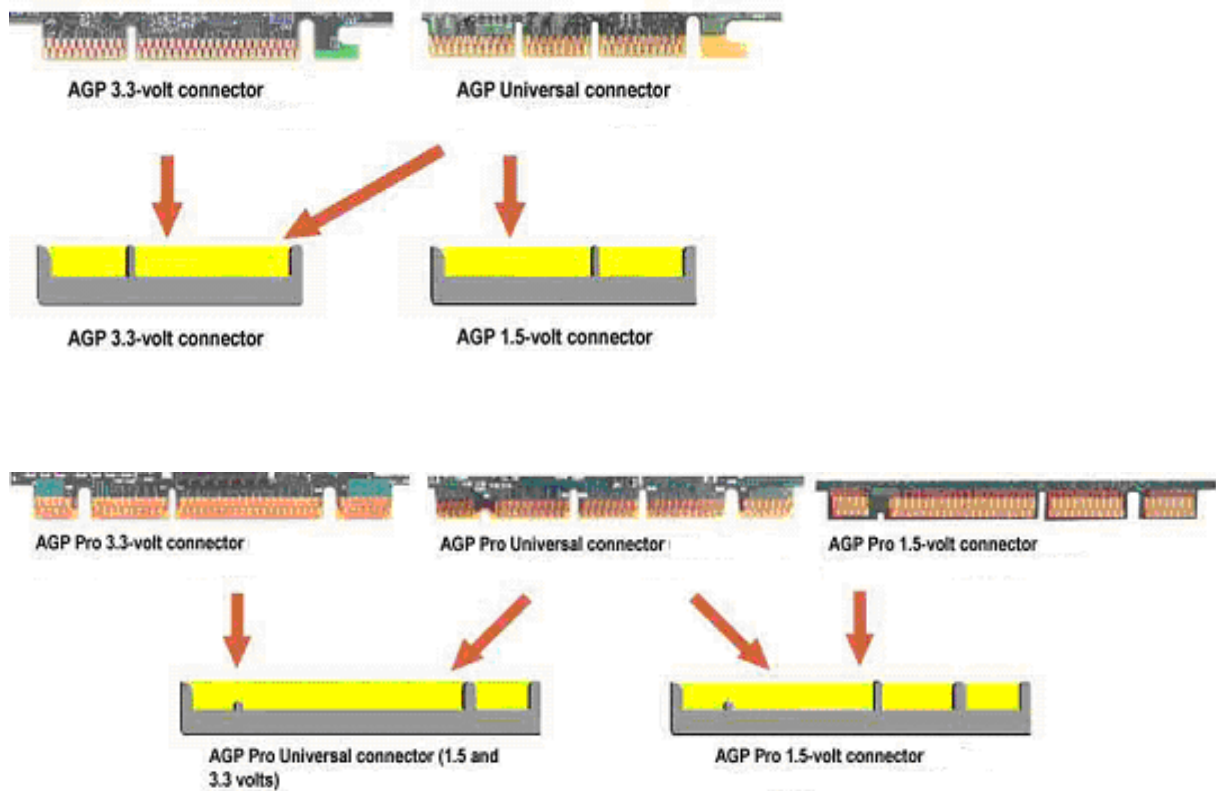


En plus des caractéristiques de vitesses, on retrouve une évolution dans les différentes tensions d'alimentations présentes sur le connecteur AGP ou ses extensions. Il y a en fait cinq connecteurs définis: AGP 3.3v, AGP 1.5v, AGP universel, AGP pro universel, AGP pro 3.3v et AGP pro 1.5v. L'aspect mécanique de ces connecteurs se retrouve dans les différentes spécifications de l'AGP:

La spécification AGP 1.0 définit les vitesses 1x et 2x avec le connecteur signalé à 3.3v
 La spécification AGP 2.0 définit les vitesses 1x, 2x et 4x avec le connecteur signalé 3.3v ou signalé 1.5v ou un connecteur universel qui supporte les deux types de cartes.
 La spécification AGP Pro qui définit les vitesses 1x, 2x, 4x avec le connecteur signalé 3.3v ou signalé 1.5v pour les signaux ou un connecteur universel qui supporte les deux cartes.
 La spécification AGP 3.0 qui définit les vitesses 1x, 2x, 4x et 8x avec le connecteur signalé 1.5v ou le connecteur signalé 1.5v AGP pro.
 Voici les différentes cartes et slots associés à ces spécifications:



La spécification AGP Pro est une extension de la norme 2.0 et 3.0 qui permet d'apporter à la carte graphique une puissance plus importante (50 Watts et 100 Watts). L'AGP Pro utilise l'interface AGP standard mais ajoute des extensions de part et d'autre permettant d'apporter la puissance additionnelle à la carte.



Carte mère	Cartes graphiques					
	AGP 1.0	AGP 2.0		AGP 3.0		
	AGP 1x / 2x (3.3 volts)	AGP 4x (1.5 volts)	AGP 2x/4x universel (3.3ou 1.5)	AGP 8x (1.5 v **)	AGP 8x Universel (1.5v) ⁽²⁾	AGP 8x Universel ⁽³⁾
AGP 1.0 (3.3)	Ok	X	Ok	X	X	Ok
AGP 2.0 (1.5)	X	Ok	Ok	X*	Ok	Ok
Universel AGP 2.0	Ok	Ok	Ok	X*	Ok	Ok
AGP 3.0 (1.5)	X	X*	X*	Ok	Ok	Ok
Universel AGP 3.0 (signalé 1.5v **)	X	Ok	Ok	Ok	Ok	Ok
Universel AGP 3.0	Ok	Ok	Ok	Ok	Ok	Ok

* Ne présente pas de détrompeur mais la carte graphique présente une broche spéciale permettant de mettre la carte hors service.

** Le niveau des signaux électriques est de 0.8v

⁽²⁾ Les vitesses supportées seront 1x, 2x, 4x dans le mode AGP et 8x, 4x dans le mode 3.0

⁽³⁾ Les vitesses supportées seront 1x, 2x, 4x dans le mode AGP 2.0 (1x, 2x quand les signaux 3.3v sont utilisés) et 8x, 4x dans le mode AGP 3.0.