6.9. Le microprocesseur

6.9.1. Introduction

6.9.1.1. Généralités

Le premier microprocesseur (Intel 4004) a été inventé en 1971. Il s'agissait alors d'une unité de calcul de 4 bits, cadencé à 108 kHz. Depuis, la puissance des microprocesseurs augmente exponentiellement.
Le microprocesseur est devenu aujourd'hui un composant électronique de base des automatismes industriels (variateurs de vitesse, automates programmables, PC industriels, etc.). Il permet de manipuler des informations numériques, c'est-à-dire des informations codées sous forme binaire, et d'exécuter les instructions stockées en mémoire.

Fig. 6.9.1

En fait, en toute rigueur, le microprocesseur est un processeur ou *Central Processing Unit* (CPU) qui a été réduit à une taille suffisamment petite pour tenir sur un seul circuit intégré (puce) illustré par la photo ci-contre. Le microprocesseur est donc un composant électronique complexe, fabriqué le plus souvent en silicium, qui regroupe un très grand nombre de transistors élémentaires interconnectés.

Néanmoins, la distinction entre *Central Processing Unit*, *CPU*, *processeur* et *microprocesseur* est souvent abandonnée au profit d'une banalisation de ces termes. La distinction se fait désormais dans sa fonction entre celle centrale et celle prenant en charge des fonctions comme le graphisme ou la compression/décompression audio-vidéo
Le microprocesseur a progressivement remplacé les fonctions analogiques de commande et de régulation des automatismes de Levage et de Manutention.

6.9.2. Définitions et vocabulaire

6.9.2.1. Instruction

Une **instruction** est l'opération élémentaire que le microprocesseur peut accomplir. Les instructions sont stockées dans la mémoire principale, en vue d'être traitée par le microprocesseur.

Une instruction est composée de deux champs :

- le **code opération**, représentant l'action que le processeur doit accomplir,
- le **code opérande**, définissant les paramètres de l'action. Le code opérande dépend de l'opération. Il peut s'agir d'une donnée ou bien d'une adresse mémoire.

Code opération	Champ opérande

Le nombre d'octets d'une instruction est variable selon le type de donnée (l'ordre de grandeur est de 1 à 4 octets).
Les instructions peuvent être classées en catégories dont les principales sont :

- **Accès à la mémoire** : des accès à la mémoire ou transferts de données entre registres.
- **Opérations arithmétiques** : opérations telles que les additions, soustractions, divisions ou multiplication.
- **Opérations logiques** : opérations ET, OU, NON, NON exclusif, etc.
- **Contrôle** : contrôles de séquence, branchements conditionnels, etc.

6.9.2.2. Jeu d'instructions

On appelle **jeu d'instructions** l'ensemble des opérations élémentaires qu'un microprocesseur peut accomplir. Le jeu d'instruction d'un processeur détermine ainsi son architecture, sachant qu'une même architecture peut aboutir à des implémentations différentes selon les constructeurs.

Le microprocesseur travaille effectivement grâce à un nombre limité de fonctions, directement câblées sur les circuits électroniques. La plupart des opérations peuvent être réalisé à l'aide de fonctions basiques. Certaines architectures incluent néanmoins des fonctions évoluées courante dans le microprocesseur.

6.9.2.3. Registres

Lorsque le microprocesseur exécute des instructions, les données sont temporairement stockées dans de petites mémoires rapides de 8, 16, 32 ou 64 bits que l'on appelle **registres** (voir chapitre 4). Suivant le type de processeur le nombre global de registres peut varier d'une dizaine à plusieurs centaines.

Les registres principaux sont :

- **le registre accumulateur** (*ACC*), stockant les résultats des opérations arithmétiques et logiques,
- **le registre d'état** (*PSW*, *Processor Status Word*), permettant de stocker des indicateurs sur l'état du système (retenue, dépassement, etc.),
- **le registre instruction** (*RI*), contenant l'instruction en cours de traitement,
- **le compteur ordinal** (*CO* ou *PC* pour *Program Counter*), contenant l'adresse de la prochaine instruction à traiter,
- **le registre tampon**, stockant temporairement une donnée provenant de la mémoire.

6.9.2.4. Mémoire cache

La **mémoire cache** (également appelée *antémémoire* ou *mémoire tampon*) est une mémoire RAM *Random Access Memory* (mémoire à accès aléatoire) très rapide (de l'ordre du nanoseconde) permettant de réduire les délais d'attente des informations stockées en mémoire vive. En effet, la mémoire centrale de l'ordinateur possède une vitesse bien moins importante que le processeur. Ainsi, si le microprocesseur cherche une donnée et qu'elle se trouve dans la mémoire cache, le gain de temps s'en trouve réduit de l'ordre de 90 % (avec 64 Ko de cache) par rapport à un accès direct à la mémoire vive.

Il existe néanmoins des mémoires beaucoup plus rapides, mais dont le coût est très élevé. La solution consiste donc à inclure ce type de mémoire rapide à proximité du processeur et d'y stocker temporairement les principales données devant être traitées par le microprocesseur. Les ordinateurs récents possèdent plusieurs niveaux de mémoire cache :

- La **mémoire cache de premier niveau** (appelée **L1 Cache**, pour **Level 1 Cache**) est directement intégrée dans le processeur. Elle se subdivise en 2 parties :
  - La première est le cache d'instructions, qui contient les instructions issues de la mémoire vive décodées lors de passage dans les pipelines.
  - La seconde est le cache de données, qui contient des données issues de la mémoire vive et les données récemment utilisées lors des opérations du processeur.

  Les caches du premier niveau sont très rapides d'accès. Leur délai d'accès tend à s'approcher de celui des registres internes aux processeurs.
- La **mémoire cache de second niveau** (appelée **L2 Cache**, pour **Level 2 Cache**) est située au niveau du boîtier contenant le processeur (dans la puce). Le cache de second niveau vient s'intercaler entre le processeur avec son cache interne et la mémoire vive. Il est plus rapide d'accès que cette dernière mais moins rapide que le cache de premier niveau.
- La **mémoire cache de troisième niveau** (appelée **L3 Cache**, pour **Level 3 Cache**) est située ainsi au niveau de la carte mère d'un PC.

Tous ces niveaux de cache permettent de réduire les temps de latence des différentes mémoires lors du traitement et du transfert des informations. Pendant que le processeur travaille, le contrôleur de cache de premier niveau peut s'interfacer avec celui de second niveau pour faire des transferts d'informations sans bloquer le processeur. De même, le cache de second niveau est interfacé avec celui de la mémoire vive (cache de troisième niveau), pour permettre des transferts sans bloquer le fonctionnement normal du processeur.

6.9.2.5. Mémoire ROM

La mémoire ROM est une mémoire permanente, non volatile et en lecture seule. Le contenu d'une mémoire ROM ne s'efface pas quand son alimentation est coupée, elle permet ainsi de conserver les données qui y été inscrites préalablement avant la mise hors tension de la machine.

C'est une mémoire destinée aux informations qui ne doivent pas être perdus et qui n'étaient à l'origine que lues, c'est à dire qu'il n'est pas possible d'écrire sur ce type de mémoire (sauf pour les mémoires EPROM plus récentes qui permettent de mettre à jour les programmes BIOS qui stocke par exemple les informations d'amorçage, le programme d'autotest du système ou de ceux à caractère applicatif tels que les options des variateurs de vitesse).

Les différents types de ROM

Il existe différents types de ROM :

- **La ROM** (Read Only Memory) pour laquelle, il n'est pas possible d'écrire.
- **La PROM** (Programmable Read Only Memory) pour laquelle, il est possible d'écrire une seule fois lors d'un processus appelé «burning». Les informations ne sont pas écrites lors du processus de fabrication du composant électronique mais plus tard, pour personnaliser un produit par exemple.
- **L'EPROM** (Erasable Programmable Read Only Memory) qui nécessite un équipement spécial pour en effacer le contenu (rayons ultraviolets de haute densité) avant de réécrire les nouvelles données.

6.9.2.6. Signaux de commande

Les signaux de commande sont des signaux électriques permettant d'orchestrer les différentes unités du processeur participant à l'exécution d'une instruction. Les signaux de commandes sont distribués grâce à un élément appelé *séquenceur*. Le signal *Read / Write*, en français *lecture / écriture*, permet par exemple de signaler à la mémoire que le processeur désire lire ou écrire une information.

6.9.2.7. Langage

Les instructions (parfois décomposées en micro instructions) données au processeur sont exprimées en binaire (code machine). Elles sont généralement stockées dans la mémoire. Elles sont lues et l'UAL les interprète. L'ensemble de ces instructions constitue un programme.
Le langage le plus proche du code machine tout en restant lisible par des humains est le langage d'assemblage, aussi appelé langage assembleur (forme francisée du mot anglais «*assembler*»). Toutefois, l'informatique a développé toute une série de langages, dits de *haut niveau* (comme le Basic, Pascal, C, C++, Fortran, etc.), destinés à simplifier l'écriture des programmes.

6.9.3. Familles de microprocesseurs

Il existe plusieurs familles de microprocesseurs possédant chacun leur propre jeu d'instructions :

- La plus connue par le grand public est celle de la famille x86, développée principalement par Intel (Pentium), AMD (Athlon), VIA, Transmeta... Les deux premiers constructeurs sont qui fabriquent la plus grande partie des processeurs pour les PC (2005).
- Les PowerPC d'IBM et Motorola équipent actuellement les Macintosh (Apple Computer) ainsi que divers systèmes embarqués. Une puce différente équipée de 10 processeurs périphérique intégrés a été choisie pour les consoles de jeu : Playstation 3, la Xbox 360 et probablement la future Nintendo Revolution...

- Le 6502 a servi à fabriquer le célèbre ordinateur Apple II.
- La famille 68000 de Motorola animait les anciens Macintosh, les Megadrive, les Atari ST et les Commodore Amiga. Leurs dérivés (Dragonball, ColdFire) sont toujours utilisés dans des systèmes embarqués.

Parmi les familles moins connues du grand public :

- La famille Sparc anime la plus grande partie des stations de travail de Sun Microsystems.
- La famille MIPS anime les stations de travail de Silicon Graphics, des consoles de jeux, comme les PSOne et des systèmes embarqués, ou des routeurs Cisco.
- La famille StrongARM est de nos jours utilisée uniquement dans les systèmes embarqués.

Remarque :

Chaque type de microprocesseur possède son propre jeu d'instruction. On distingue ainsi les familles de processeurs suivants, possédant chacun un jeu d'instruction qui leur est propre :

Cela explique qu'un programme réalisé pour un type de processeur ne puisse fonctionner directement sur un système possédant un autre type de processeur, à moins d'une traduction des instructions, appelée **émulation**. Le terme « **émulateur** » est utilisé pour désigner le programme réalisant cette traduction.

6.9.4. Structure de base du microprocesseur

6.9.4.1. Unités fonctionnelles

Le processeur est constitué d'un ensemble d'unités fonctionnelles reliées entre elles. L'architecture d'un microprocesseur est très variable d'une architecture à une autre, cependant les principaux éléments d'un microprocesseur sont les suivants :

- Une **unité d'instruction** (ou *unité de commande*, en anglais *control unit*) qui lit les données arrivant, les décode puis les envoie à l'unité d'exécution. L'unité d'instruction est notamment constituée des éléments suivants :
  - Le **séquenceur** (ou *bloc logique de commande*) chargé de synchroniser l'exécution des instructions au rythme d'une horloge. Il est ainsi chargé de l'envoi des signaux de commande,
  - Le **compteur ordinal** contenant l'adresse de l'instruction en cours,
  - Le **registre d'instruction** contenant l'instruction suivante.
- Une **unité d'exécution** (ou *unité de traitement*), qui accomplit les tâches que lui a données l'unité d'instruction. L'unité d'exécution est notamment composée des éléments suivants :
  - L'**unité arithmétique et logique** (notée **UAL** ou en anglais *ALU* pour *Arithmetical and Logical Unit*). L'UAL assure les fonctions basiques de calcul arithmétique et les opérations logiques (ET, OU, Ou exclusif, etc.),
  - L'**unité de virgule flottante** (notée **FPU**, pour *Floating Point Unit*), qui accomplit les calculs complexes non entiers que ne peut réaliser l'unité arithmétique et logique,
  - Le **registre d'état**,
  - Le **registre accumulateur**.
- Une **unité de gestion des bus** (ou *unité d'entrées-sorties*), qui gère les flux d'informations entrant et sortant, en interface avec la mémoire vive du système.

6.9.4.2. Schéma simplifié

Le schéma ci-dessous donne une représentation simplifiée des éléments constituant le processeur :

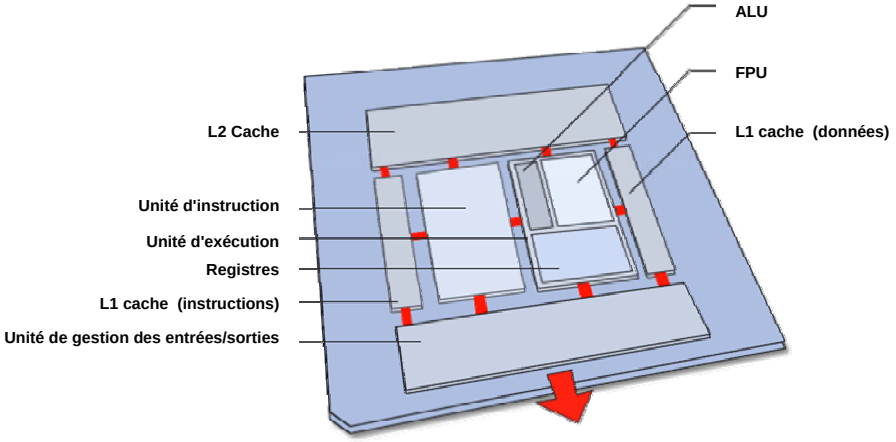


Fig. 6.9.2

Remarque :

On peut également trouver des circuits avec une horloge à quartz qui est intégrée au microprocesseur.

6.9.5. Fonctionnement

6.9.5.1. Introduction

Le microprocesseur est un circuit électronique cadencé au rythme d'une horloge interne, grâce à un cristal de quartz qui, soumis à un courant électrique, envoie des impulsions appelées « **top** » (signal régulier rapide, imposant un rythme au circuit et, assurant éventuellement une synchronisation avec les autres composants, tel que la mémoire).

Au milieu des années 1980, les microprocesseurs fonctionnaient de 4 à 8 MHz. Courant 2004, cette vitesse d'horloge atteint 4 GHz sur des modèles commerciaux (5 GHz en laboratoire). Plus la vitesse de l'horloge est élevée, plus le microprocesseur sera capable d'exécuter à un rythme élevé les instructions de base des programmes. Mais l'augmentation de la vitesse d'horloge présente des inconvénients : plus le microprocesseur tourne vite, plus il consomme, et plus il chauffe.

A chaque top d'horloge le processeur exécute une action, correspondant à une instruction ou une partie d'instruction. L'indicateur appelé **CPI** (*Cycles Par Instruction*) permet de représenter le nombre moyen de cycles d'horloge nécessaire à l'exécution d'une instruction sur un microprocesseur. La puissance du processeur peut ainsi être caractérisée par le nombre d'instructions qu'il est capable de traiter par seconde. L'unité utilisée est le **MIPS** (Millions d'Instructions Par Seconde) correspondant à la fréquence du processeur que divise le *CPI*.

Les microprocesseurs actuels sont optimisés pour exécuter plus d'une instruction par cycle d'horloge, ce sont des microprocesseurs avec des unités d'exécution parallélisées. De plus ils sont dotés de procédures qui «anticipent» les instructions suivantes avec l'aide de la statistique.
Dans la course à la puissance des microprocesseurs, deux méthodes d'optimisation sont en concurrence :

- La technologie du jeu d'instructions simplifié (RISC, Reduced Instruction Set Computer), rapide avec des instructions simples, facile à fabriquer et dont on peut monter la vitesse de l'horloge sans trop de difficultés techniques.
- La technologie appelée CISC (Complex Instruction Set Computer soit « *ordinateur à jeu d'instruction complexe* »), dont chaque instruction nécessite plus de cycles d'horloge, mais qui a en son cœur beaucoup d'instructions pré câblées.

Néanmoins, avec la considérable augmentation de la taille des puces électroniques et la gigantesque accélération des fréquences d'horloge, la distinction entre *RISC* et *CISC* a quasi complètement disparu. Là où des familles tranchées existaient, on observe aujourd'hui des microprocesseurs où une structure interne *RISC* apporte de la puissance tout en restant compatible avec une utilisation de type *CISC*.

6.9.5.2. Circulation de l'information dans un microprocesseur

Les phases d'exécution d'une instruction pour un processeur contenant un pipeline « classique » à 5 étages sont les suivantes :

1. **LI** : (*Lecture de l'Instruction* (en anglais *FETCH instruction*) depuis le cache,
2. **DI** : *Décodage de l'Instruction* (*DECODe instruction*) et recherche des opérandes (Registre ou valeurs immédiate),
3. **EX** : *Exécution de l'Instruction* (*EXECute instruction*) (si ADD, on fait la somme, si SUB, on fait la soustraction, etc.),
4. **MEM** : *Accès mémoire* (*MEMory access*), écriture dans la mémoire si nécessaire ou chargement depuis la mémoire
5. **ER** : *Ecriture* (*Write instruction*) de la valeur calculée dans les registres.La supervision de l'exécution de l'instruction

6.9.6. Caractéristiques principales

Un microprocesseur est définit par :

- La largeur de ses registres internes de manipulation de données (8, 16, 32, 64, 128) bits.
- Par la cadence de son horloge exprimée en MHz (mega hertz) ou GHz (giga hertz).
- Son jeu d'instructions *ISA* (*Instructions Set Architecture*) dépendant de la famille (CISC, RISC, etc)
- La finesse de sa gravure exprimée en nm (nanomètres) et sa microarchitecture interne.

Mais ce qui caractérise principalement un microprocesseur est la famille à laquelle, il appartient :

- L'architecture **CISC** consiste à câbler dans le microprocesseur des instructions complexes, difficiles à créer à partir des instructions de base. Cette technologie est basée sur un jeu de plus de 400 instructions.

  Cette architecture est utilisée en particulier par les microprocesseurs de type 80x86. Elle possède un coût de réalisation élevé dû aux fonctions évoluées imprimées sur le silicium.

  D'autre part, les instructions sont de longueurs variables et peuvent parfois nécessiter plus d'un cycle d'horloge. Or, un microprocesseur basé sur l'architecture CISC ne peut traiter qu'une instruction à la fois, d'où un temps d'exécution conséquent.

- L'architecture d'un microprocesseur utilisant la technologie **RISC** n'a pas de fonctions évoluées câblées. Le RISC est constitué d'instructions simples qui permettent de gagner en rapidité d'exécution.

  Les programmes doivent ainsi être traduits en instructions simples, ce qui entraîne un développement plus difficile et/ou un compilateur plus puissant. Une telle architecture possède un coût de fabrication réduit par rapport aux processeurs CISC. De plus, les instructions, simples par nature, sont exécutées en un seul cycle d'horloge, ce qui rend l'exécution des programmes plus rapide qu'avec des processeurs basés sur une architecture CISC.
  Un microprocesseur RISC peut ainsi atteindre une vitesse d'exécution jusqu'à 70% plus rapide qu'un CISC de même fréquence.
  Depuis le P5 (pentium), les microprocesseurs INTEL utilisent des technologies empruntées de la famille RISC.
  Enfin, il est également intéressant de noter que de tels microprocesseurs sont capables de traiter plusieurs instructions simultanément en les traitant en parallèle.

- **VLIW** (Very Long Instruction Word) VLIW, est une famille de microprocesseurs à mot d'instruction très long (couramment supérieur à 128 bits). Sont codés à l'intérieur de ces instructions les opérations à effectuer par les différentes unités de calcul disponible dans l'unité centrale.
  Il est de la responsabilité du compilateur de générer un code qui prend en compte la disponibilité des ressources et ne provoque pas de problèmes matériel lors de son exécution.
  Un mot VLIW est parfois appelé un *bundle*, selon l'architecture. Chaque *bundle* peut contenir plusieurs instructions. Ces instructions sont positionnées à un certain emplacement du *bundle* : un *slot*. En général chaque *slot* correspond à un type d'instruction, c'est-à-dire que le compilateur ne peut pas affecter à un *slot* dédié aux instructions arithmétiques entières une multiplication flottante, par exemple.

- **DSP** (Digital Signal Processor). Cette dernière famille de microprocesseur est relativement spécifique. En effet un microprocesseur est un composant programmable et est donc à priori capable de réaliser tout type de programme. Toutefois, dans un souci d'optimisation, des microprocesseurs spécialisés sont conçus et adaptés à certains types de calculs (3D, son, ...). Les DSP sont ainsi des processeurs orientés pour les calculs liés au traitement de signaux. Par exemple, il n'est pas rare de voir implémenter des Transformées de Fourier dans un DSP.

Enfin, il faut remarquer qu'un microprocesseur possède trois types de bus :

- Un **bus de données**, définit la taille des données manipulable (indépendamment de la taille des registres internes)
- Un **bus d'adresse** définit le nombre de cases mémoire accessibles
- Un **bus de commande** définit la gestion du microprocesseur IRQ, RESET etc..

6.9.7. Améliorations des performances

Au cours des années, les constructeurs de microprocesseurs (appelés *fondeurs*), ont mis au point un certain nombre d'améliorations technologiques permettant d'optimiser le fonctionnement du processeur.

6.9.7.1. Le parallélisme

Le parallélisme consiste à exécuter simultanément, sur des processeurs différents, des instructions relatives à un même programme. Cela se traduit par le découpage d'un programme en plusieurs processus traités en parallèle afin de gagner en temps d'exécution.

Ce type de technologie nécessite toutefois une synchronisation et une communication entre les différents processus, à la manière du découpage des tâches dans une entreprise : le travail est divisé en petits processus distincts, traités par des services différents. Le fonctionnement d'une telle entreprise peut être très perturbé lorsque la communication entre les services ne fonctionne pas correctement.

6.9.7.2. Le pipeline

Le pipeline (ou *pipelining*) est une technologie visant à permettre une plus grande vitesse d'exécution des instructions en parallélisant des étapes.

Les instructions sont alors organisées en file d'attente dans la mémoire, et sont chargées les unes après les autres.
Grâce au pipeline, le traitement des instructions nécessite au maximum les cinq étapes décrites précédentes. Dans la mesure où l'ordre de ces étapes est invariable (LI, DI, EX, MEM et ER), il est possible de créer dans le processeur un certain nombre de circuits spécialisés pour chacune de ces phases.

L'objectif du pipeline est d'être capable de réaliser chaque étape en parallèle avec les étape amont et aval, c'est-à-dire de pouvoir lire une instruction (LI) lorsque la précédente est en cours de décodage (DI), que celle d'avant est en cours d'exécution (EX), que celle située encore précédemment accède à la mémoire (MEM) et enfin que la première de la série est déjà en cours d'écriture dans les registres (ER).

Il faut compter en général 1 à 2 cycles d'horloge (rarement plus) pour chaque phase du pipeline, soit 10 cycles d'horloge maximum par instruction. Pour deux instructions, 12 cycles d'horloge maximum seront nécessaires (10+2=12 au lieu de 10*2=20), car la précédente instruction était déjà dans le pipeline. Les deux instructions sont donc en traitement dans le processeur, avec un décalage d'un ou deux cycles d'horloge). Pour 3 instructions, 14 cycles d'horloge seront ainsi nécessaires, etc.

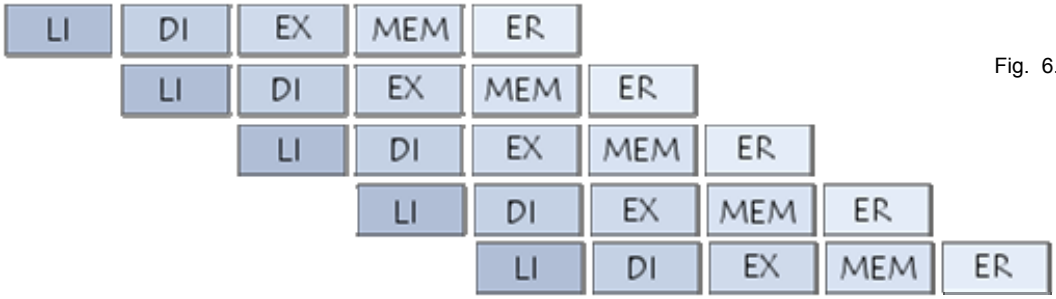


Fig. 6.9.3

Le principe du pipeline est ainsi comparable avec une chaîne de production de voitures. La voiture passe d'un poste de travail à un autre en suivant la chaîne de montage et sort complètement assemblée à la sortie du bâtiment. Pour bien comprendre le principe, il est nécessaire de regarder la chaîne dans son ensemble, et non pas véhicule par véhicule. Il faut ainsi 3 heures pour faire une voiture, mais pourtant une voiture est produite toute les minutes !

Il faut noter toutefois qu'il existe différents types de pipelines, de 2 à 40 étages, mais le principe reste le même.

6.9.7.3. L'exécution dynamique

L'exécution dynamique peut être décomposée en trois composants :

- La prédiction de branchement :

  Ce procédé consiste à deviner l'emplacement de la prochaine instruction devant être traitée, puis à la diriger vers le bon pipeline. Cela permet d'éviter les sauts et les boucles risquant de faire perdre les gains apportés par les pipelines. Au dire d'Intel, un processeur tel que le Pentium II aurait une capacité de prédiction de l'ordre de 90%

- L'analyse de flux :

  Ce procédé est chargé de réagencer l'ordre de traitement des données afin de l'optimiser. Il devra aussi choisir entre les deux pipelines, l'entier et celui à virgule flottante. De plus, il lui est nécessaire de tenir compte du temps de traitement de chaque instruction.
  Ainsi, il permet d'obtenir de bien meilleures performances qu'en traitant le programme original tel quel. En fait, il se charge de réparer les dégâts provoqués par un mauvais compilateur.

- L'exécution spéculative *:*

  Ce dernier procédé permet de traiter les instructions des différentes portions de code envisageable à l'avance. Ainsi, il peut anticiper le résultat qui devra être obtenu après un saut.

6.9.7.4. Technologie superscalaire

La technologie superscalaire (en anglais *superscaling*) consiste à disposer plusieurs unités de traitement en parallèle afin de pouvoir traiter plusieurs instructions par cycle.

6.9.7.5. Hyper Treading

La technologie HyperThreading (ou *Hyper-Threading*, noté *HT*, traduisez *HyperFlots* ou *HyperFlux*) consiste à définir deux processeurs logiques au sein d'un processeur physique. Ainsi, le système reconnaît deux processeurs physiques et se comporte en système multitâche en envoyant deux thréads simultanés, on parle alors de **SMT** (*Simultaneous Multi Threading*). Cette « supercherie » permet d'utiliser au mieux les ressources du processeur en garantissant que des données lui sont envoyées en masse.

6.9.8. Les multiprocesseurs

Les architectures multiprocesseurs permettent à une machine d'utiliser de façon concurrente, plusieurs processeurs qui fonctionnent en parallèle. On peut ainsi partager les tâches et obtenir une puissance de calcul plus importante qu'avec un seul processeur. Il existe deux types d'architecture multiprocesseurs :

- l'architecture symétrique SMP (*Symmetric multiprocessing*), qui utilise plusieurs processeurs identiques afin d'augmenter la puissance de calcul brute de la machine,
- l'architecture asymétrique AMP (*Asymmetric multiprocessing*), qui adjoint au processeur central des processeurs souvent spécialisés, tels qu'on en trouve dans tous les ordinateurs modernes, par exemple pour contrôler les périphériques ou traiter des images ou des sons.

6.9.9. Les microcontrôleurs

Un microcontrôleur est un microprocesseur possédant dans le même circuit intégré de la mémoire et d'autres périphériques (pour gérer par exemple un bus I2C, un bus série ...). Il existe beaucoup de familles de microcontrôleurs, se différenciant par la vitesse de leur processeur et par le nombre de périphériques qui le composent.
L'utilisation d'un microcontrôleur est beaucoup plus simple que celle d'un processeur.

6.9.10. Le problème de l'échauffement

6.9.10.1. Définition

Notons qu'un microprocesseur contenant beaucoup de transistors verra sa puissance dépendre fortement de la température. Ceci est dû aux mouvements de plus en plus aléatoires des électrons en fonction de l'élévation de la température

Malgré l'usage de techniques de gravures de plus en plus fines, l'échauffement des microprocesseurs reste sensiblement proportionnel au carré de leur tension à architecture donnée. Si V est la tension d'alimentation, f la fréquence d'horloge, et k un coefficient d'ajustement, on peut calculer la puissance dissipée P par la relation :

$$P = k \bullet V^2 \bullet f$$

Remarque :

Le problème d'évacuation de la chaleur générée par les microprocesseurs nécessite l'utilisation fréquente de ventilateurs qui sont sources de nuisances sonores parfois difficilement compatibles avec un environnement de bureau.

6.9.10.2. Exemples

- Un Intel 686 à 1 GHz (1,7V), dissipe typiquement 34 W.
- A 2 GHz un Opteron dissipe 107 W et un PowerPC G5 55 W.

6.9.11. Technologies microélectroniques

6.9.11.1. Introduction

Il existe deux classes de circuits intégrés :

- Les circuits bipolaires (TTL, ECL, I^2L),
- Et les circuits unipolaires (NMOS, PMOS, CMOS).

Chacune de ces familles ont leurs avantages et leurs inconvénients. Parmi ces différentes technologies, nous en étudierons plus particulièrement deux très utilisées pour la réalisation des circuits intégrés logiques dans le monde industriel (automates programmables, variation de vitesse, …). Il s'agit de :

- La famille TTL (Transistor Transistor Logic), réalisée avec des transistors bipolaires, qui est rapide, dont le courant de sortie peut atteindre 16 mA mais qui nécessite une tension très précise de 5 volts et a une consommation relativement élevée.
- La famille CMOS (Complementary MOS), réalisée avec des transistors MOS de type canal N et canal P. Elle fonctionne avec une tension comprise entre 3V et 18V. Sa consommation, extrêmement faible facilite son intégration à grande échelle.

6.9.11.2. Présentation

Les circuits intégrés logiques comportent fréquemment plusieurs opérateurs du même type (ET, OU NAND etc.) qui se trouvent encapsulés dans un seul et même boîtier.

Le boîtier contenant ces opérateurs dispose des broches d'entrée et de sortie de chaque opérateur ainsi que de broches d'alimentation (V_{CC} ou V_{DD} pour le potentiel le plus élevé et GND ou V_{SS} pour le potentiel le plus bas). Une alimentation du boîtier est indispensable au fonctionnement des opérateurs logiques.

La figure 6.9.4 ci-dessous illustre le schéma de brochage d'un circuit intégré TTL 7400 comportant 4 portes NAND. L'alimentation de ce boîtier s'effectue entre les broches 14 (Vcc) et 7 (0V).

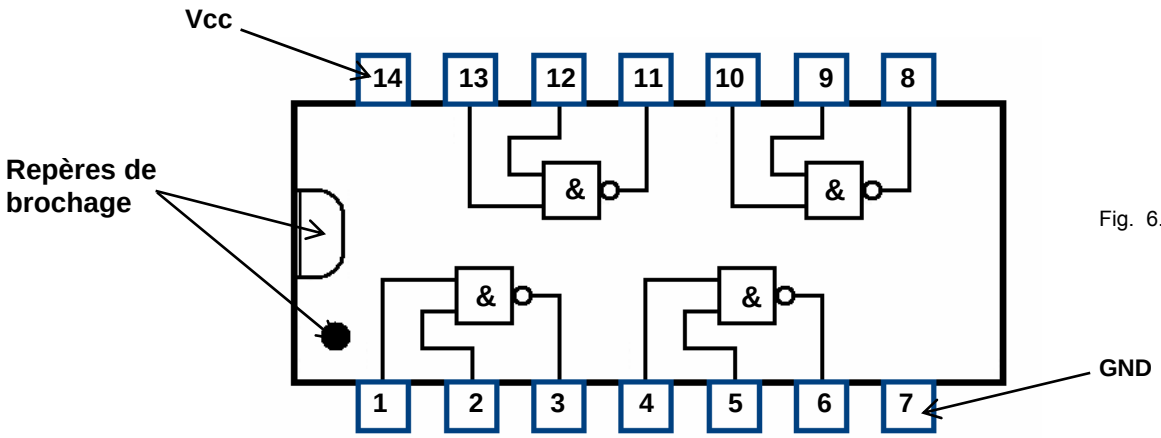


Fig. 6.9.4

Nous avons vu au cours de l'étude des fonctions logiques au chapitre 5 que les opérateurs logiques interprétaient les états logiques 0 et 1 en en entrée, et fournissaient en sortie des états logiques 0 et 1 décrits par la table de vérité de l'opérateur.

En pratique, ces états logiques correspondent à des niveaux de tension dépendant de la technologie des circuits. On retrouve ces informations dans la documentation technique des constructeurs des composants électroniques.

Pour la comprendre il faut absolument connaître les quelques définitions suivantes rencontrées dans les DATA BOOKs au sein des DATA SHEETs de la documentation technique des constructeurs.

6.9.11.3. Structure TTL

La figure 6.9.5 ci-après illustre la structure d'une porte NAND TTL :

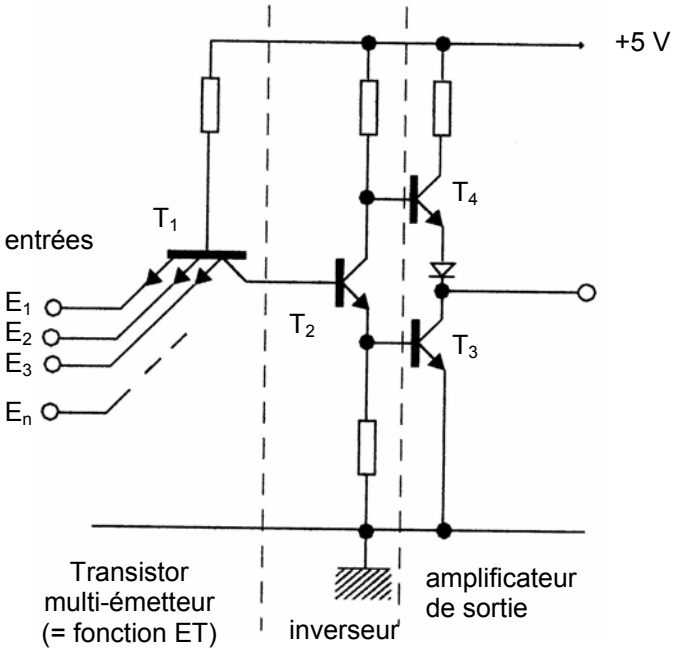


Fig. 6.9.5

Fonctionnement :

Le transistor T_1 possède autant d'émetteurs que la porte possède d'entrées (transistor multi-émetteur). Si une seule des entrées est à l'état bas, le transistor T_1 conduit. Il s'en suit que T_2 est bloqué et que T_4 est conducteur. La sortie est à l'état haut (+ 5 V) : c'est une fonction NAND.

Lorsqu'une entrée est au niveau bas, le courant d'émetteur de T_1 se referme par le circuit d'attaque : la logique TTL est à extraction de courant.

6.9.11.4. Structure CMOS

Ce sont des circuits intégrés à base de transistors MOS.
Dans la structure CMOS, tous les composants sont actifs (pas de résistances), la consommation est très faible et le taux d'intégration de la technologie est très élevé

Les figures 6.9.6a et 6.9.6b ci-après illustrent la structure d'un circuit inverseur CMOS réalisé à l'aide deux transistors MOS complémentaires.

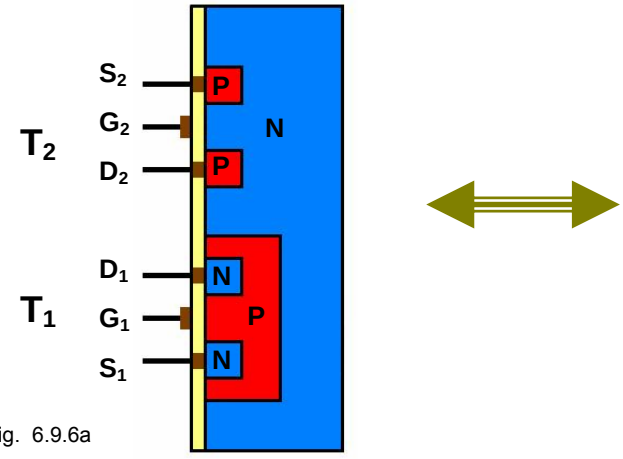


Fig. 6.9.6a

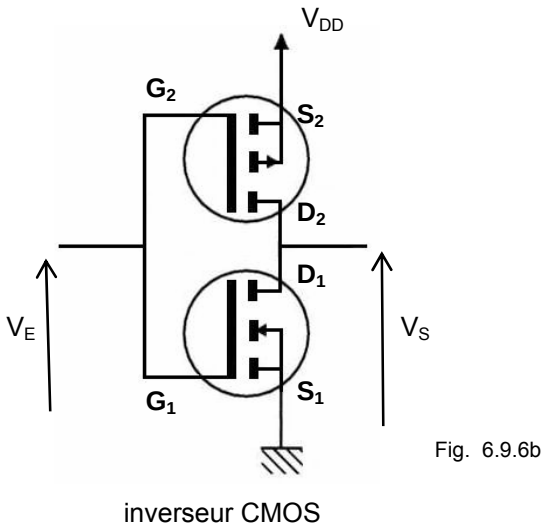


Fig. 6.9.6b

inverseur CMOS

Fonctionnement

Etudions le fonctionnement de l'inverseur CMOS décrit précédemment en fonction du niveau de la tension de commande V_E. Soit :

Si $V_E = 0$ (état 0) :

$V_{G1S1} = 0 \rightarrow T_1$ bloqué
$V_{G2S2} = -V_{DD} \rightarrow T_2$ conducteur
$\Rightarrow V_S = V_{DD}$ (état 1)

Si $V_E = V_{DD}$ (état 1) :

$V_{G1S1} = 0 \rightarrow T_2$ bloqué
$V_{G2S2} = V_{DD} \rightarrow T_1$ conducteur
$\Rightarrow V_S = 0$ (état 0)

6.9.11.5. Alimentation (Supply Voltage):

Le boîtier contenant les opérateurs logiques doit être alimenté dans la plage des tensions d'alimentation préconisée par le constructeur.

Exemple :
Le circuit intégré TTL 7400 contenant 4 opérateurs NAND, pour fonctionner correctement, doit disposer entre ses bornes Vcc et GND (bornes 14 et 7) d'une tension d'alimentation de 5 Volts ± 5%.

La valeur de la tension d'alimentation dépend de la technologie du circuit intégré :

- ***En TTL***

  L'alimentation doit être impérativement continue et stable. Vcc = 5V +/- 5%

- ***En CMOS***

  Le choix de la tension d'alimentation est plus large de 3V à 18V. Mais les performances dynamiques se dégradent aux faibles niveaux d'alimentation.

6.9.11.6. Niveaux de tension d'entrée et de sortie (Input Voltage/Output Voltage)

L'état logique des entrée et sorties des fonctions logiques dépend du niveau des tensions reçues et générées par les différents opérateurs du boîtier.
La figure 6.9.7.ci-dessous permet d'en donner les différentes définitions et quelques ordres de grandeur :

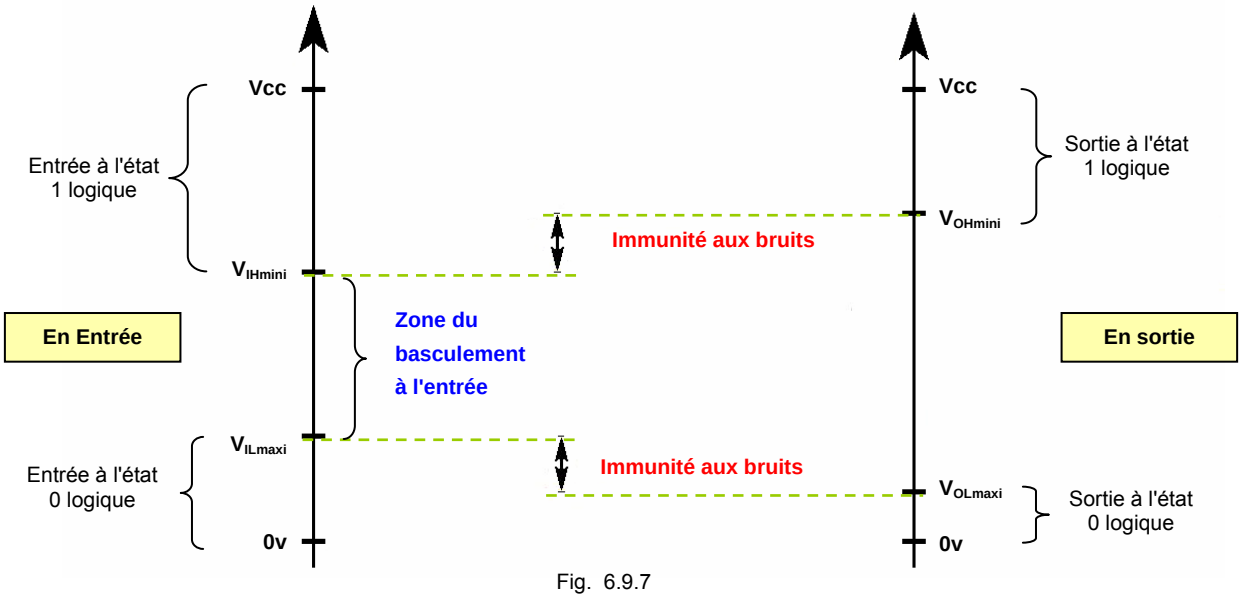


Fig. 6.9.7

- V_{IHmin} correspond à la différence de potentiel minimale qu'il faut appliquer sur l'entrée d'un opérateur logique pour que cette entrée soit interprétée comme un état haut logique (1 logique).
  Ex : V_{IHmin} = 2 V pour le circuit 74LS00

- V_{ILmax} correspond à la différence de potentiel maximale qu'il est possible d'appliquer sur l'entrée d'un opérateur logique pour que cette entrée soit interprétée comme un état bas logique. (0 logique).
  Ex : V_{ILmax} = 0,8 V pour le circuit 74LS00

- V_{OHmin} correspond à la valeur minimale de tension, garantie par le constructeur en sortie de l'opérateur logique, lorsque cette sortie est à l'état haut (1 logique).
  Ex : V_{OHmin} = 2,7 V pour le circuit 74LS00

- V_{OLmax} correspond à la valeur maximale de tension, garantie par le constructeur en sortie de l'opérateur logique, lorsque cette sortie est à l'état bas (0 logique).
  Ex : V_{OLmax} = 0,5 V pour le circuit 74LS00

L'immunité aux bruits correspond à l'amplitude de la tension parasite qui peut exister entre la sortie d'une fonction logique et l'entrée d'une autre fonction logique qui lui est reliée.

- ***En TTL (74LS00)***

  $V_{OH\ mini}$ = 2,7 V
  $V_{IH\ mini}$ = 2 V
  } **Immunité aux bruits à l'état Haut ⇨ 0,7 V**

  $V_{OL\ maxi}$ = 0,5 V
  $V_{IL\ maxi}$ = 0,8 V
  } **Immunité aux bruits à l'état Bas ⇨ 0,3 V**

- ***En CMOS***

  En technologie CMOS, les seuils de tension dépendent de la valeur de la tension d'alimentation Vcc du circuit intégré. Soit :

  $V_{OH\ mini}$ = 0,95•Vcc
  $V_{IH\ mini}$ = 0,55•Vcc
  } **Immunité aux bruits à l'état Haut ⇨ 0,4 Vcc**

  $V_{OL\ maxi}$ = 0,05•Vcc
  $V_{IL\ maxi}$ = 0,45•Vcc
  } **Immunité aux bruits à l'état Bas ⇨ 0,4 Vcc**

6.9.11.7. Courants d'entrée et de sortie (Input Current)/ (Output current)

(Dans ce qui suit les courants seront notés positifs lorsqu'ils rentrent dans le circuit, et négatifs dans le cas contraire)

- I_{ILmax} correspond à la valeur maximale du courant pouvant être fourni par l'entrée d'un opérateur logique lorsque cette entrée est à l'état bas (0 logique).

- I_{IHmax} correspond à la valeur maximale du courant pouvant être absorbé par l'entrée d'un opérateur logique lorsque cette entrée est à l'état haut (1 logique).

- I_{OLmax} correspond à la valeur maximale du courant absorbé par la sortie d'un opérateur logique, lorsque cette sortie est à l'état bas (0 logique).

- I_{OHmax} correspond à la valeur maximale du courant fourni par la sortie d'un opérateur logique, lorsque cette sortie est à l'état haut (1 logique).

Les valeurs de ces différentes intensités dépendent de la technologie des circuits intégrés :

- ***En TTL***

  A l'état bas une entrée de fonction TTL a besoin d'un courant sortant I_{ILmax} = 1,6mA

  A l'état haut le courant d'entrée est I_{IHmax} = 40µA

  La sortie peut délivrer I_{OHmax} = 400µA au 1L et absorber I_{OLmax} = 16mA au 0L

- ***En CMOS***

  Les courants d'entrée sont inférieurs à 1µA et les sorties peuvent véhiculer plus de 1 mA.

6.9.11.8. Sortance (Fan out)

On appelle *sortance* le nombre maximal d'entrées que l'on peut connecter à une sortie sans que son niveau ne sorte de la zone autorisée. Elle s'exprime en unité de charge (U.L).

Si l'on reprend les valeurs de courant précédentes, la sortance en TTL est typiquement de 10 à l'état haut. En CMOS, la sortance est limitée par les capacités parasites d'entrée (≈ 5pF) qui réduisent les temps de commutation et non par les valeurs statiques des courants d'entrée-sortie.

6.9.11.9. Temps de propagation (Propagation Delays)

Quand la sortie d'une fonction logique change d'état d'une sortie, celui-ci n'est pas instantané. Le traitement de l'information n'est pas immédiat, il existe un temps de propagation tp

Il faut considérer un temps de montée tm (passage du 0L au 1L) ou un temps de descente td (passage du 1L au 0L). Le passage de l'état logique "0L" à l'état logique "1L", et le passage inverse de l'état logique "1L" à l'état logique "0L" de la sortie d'un opérateur logique prennent respectivement les temps :

tp_{LH} et tp_{HL} (Pour temps de propagation LOW to HIGH et HIGH to LOW)

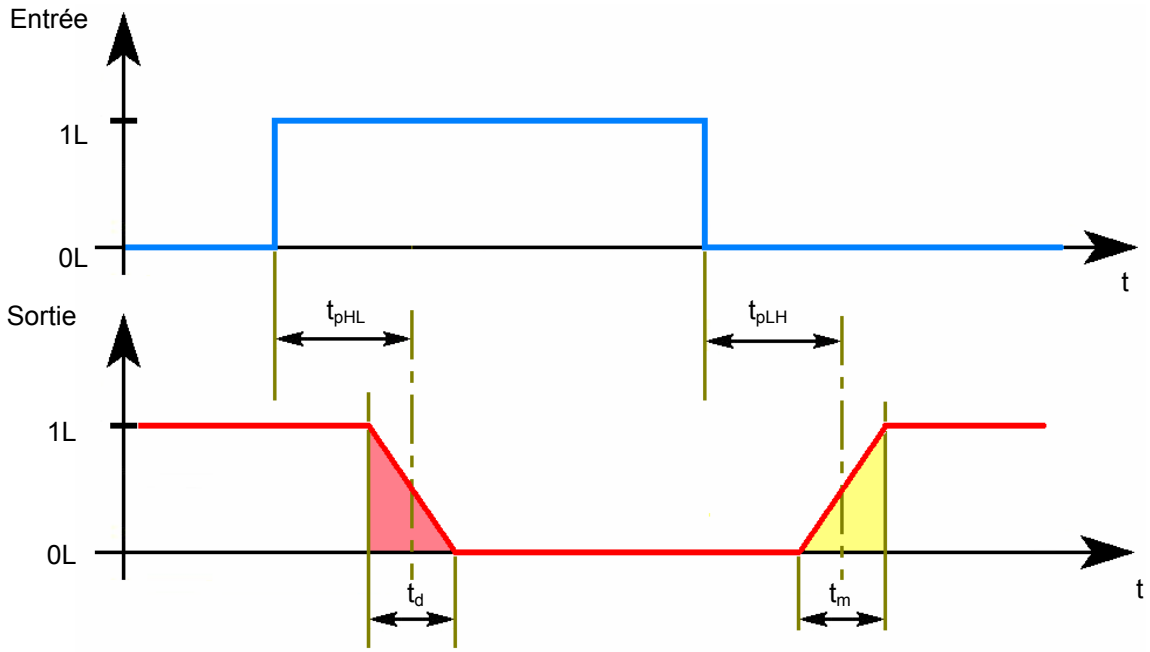


Fig. 6.9.8

La figure 6.9.8 ci-dessus illustre les évolutions dynamiques des signaux entrée/sortie d'une porte inverseuse

.

- ***En TTL***

  Tp varie selon la sous-famille de 10ns (TTL "N") à 1,5ns (TTL "AS")

- ***En CMOS***

  Le temps de propagation Tp dépend du niveau de la tension d'alimentation Vcc car la vitesse de transition augmente quand on fait croître Vcc.
  Par ailleurs, chaque entrée CMOS présentant une capacité parasite voisine de 5pF, la capacité équivalente vue par la sortie influe fortement sur le temps de réponse de celle-ci comme le montre la figure 6.9.9 ci-dessous :

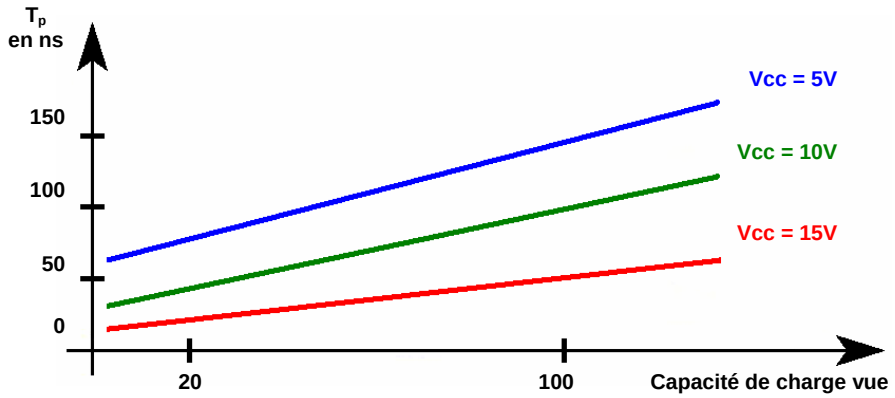


Fig. 6.9.9

6.9.11.10. Puissance consommée

La puissance consommée par un boîtier dépend de sa fréquence d'utilisation et du nombre d'opérateurs utilisés.

La figure suivante 6.9.10 propose une comparaison de la consommation des fonctions TTL et CMOS par porte :

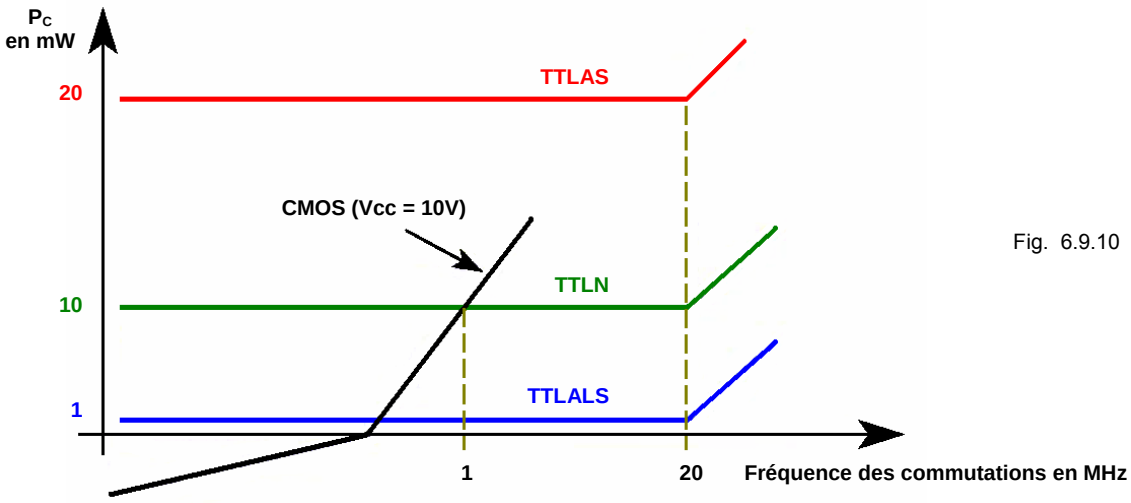


Fig. 6.9.10

On remarque que la consommation des boîtiers en technologie CMOS est quasi nulle aux basses fréquences, mais qu'elle atteint celle de la technologie TTLN (normale) au voisinage de la fréquence de commutation Fc = 1MHz.

Remarque :

A chaque commutation la fonction appelle une impulsion de courant (di/dt) sur les broches d'alimentation du boîtier. Pour que la tension Vcc ne varie pas du fait des inductances de câblage (L.di/dt), on place des condensateurs de découplage (0,1µF) sur l'alimentation de chaque circuit intégré.

6.9.11.11. Etage de sortie, interfaçage

Montage "*totem pôle*"

La plupart des fonctions logiques (ET, OU, NAND, etc) quelle que soit la technologie (TTL ou CMOS), utilisent le montage totem pole comme schéma électrique de l'étage de sortie des portes.

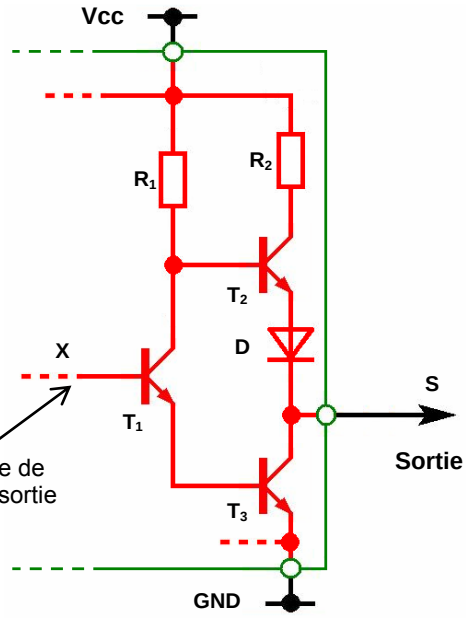


Fig. 6.9.11

Dans ce type de montage, l'étage de sortie est constitué par un ensemble de deux transistors qui, en régime statique, ne sont jamais passant en même temps. Selon l'état (passant ou bloqué) de ces transistors, ils permettent de relier la sortie soit au 0V soit à la tension d'alimentation Vcc définissant ainsi deux états logiques en sortie.

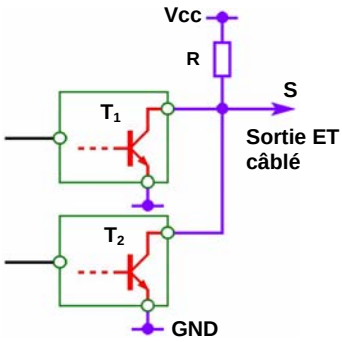
La figure 6.9.11 ci-contre illustre le principe d'un tel schéma en technologie TTL.

Lorsque l'entrée de commande X est à l'état haut (1L), les transistors T_1 et T_3, sont saturés. La tension collecteur du transistor T_1 est alors voisine de celle de son émetteur c'est-à-dire environ 0,7 V (V_{BE} du transistor T_3). La tension de sortie est en conséquence voisine de 0V (0L). Dans ce cas, le transistor T2 ne peut pas conduire car il faudrait sur sa base un potentiel supérieur à 1,4V (à cause de la diode D en série avec l'émetteur).

Lorsque l'entrée de commande X est à l'état bas (0L), le transistor T_1 est bloqué ce qui entraîne également le blocage du transistor T_3. Le transistor T_2 devient alors conducteur et la tension de sortie légèrement inférieure à la tension d'alimentation Vcc à cause des chutes de tension provoquées par la résistance R_2 et la diode D. c (état 1L).
Le transistor T_2 permet de fournir le courant de charge des capacités parasites des entrées qui sont connectées à la sortie et ainsi d'améliorer la transition bas-haut du signal de sortie.

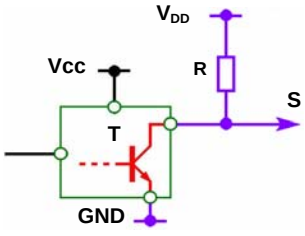
Sorties en *collecteur ouvert*

Il existe un autre type de montage de l'étage de sortie où seul le transistor câblé à la masse (transistor "du bas du montage totem pôle") est présent. Dans ce cas, le signal de sortie est pris sur le collecteur de ce transistor, d'où l'appellation de *collecteur ouvert* pour un tel montage. Ce transistor se comporte comme un interrupteur à la masse, ouvert ou passant, correspondant respectivement aux états de sortie H (1 logique) ou L (0 logique) du montage totem pôle. Pour retrouver à la sortie de ce montage un signal logique 1L ou 0L, il faut polariser le collecteur en le reliant à la source d'alimentation à travers une résistance dite "de tirage".
Le montage collecteur ouvert facilite l'élaboration d'interfaces (commande de relais de sortie, de transformateur d'impulsions, etc.) ou pour réaliser une, certaines fonctions spéciales.
Les figures 6.9.12 a), b) et c) illustrent quelques exemples d'applications de la sortie collecteur ouvert.



6.9.12a)

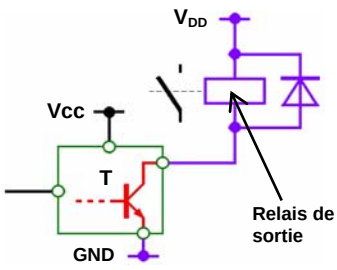
fonction "ET câblé"



La sortie S évolue de 0V à VDD (+15V par exemple) alors que la fonction est alimentée en 5V.

6.9.12b)

interface logique a



6.9.12c)

commande d'un relais de sortie

6.9.11.12. Précautions d'emploi

Entrées non utilisées

- ***En TTL***

  Une entrée non connectée (en l'air) est naturellement à l'état haut. Il est **toutefois conseillé** de ne pas laisser une entrée en l'air, car celle-ci devient sensible aux bruits (capacité parasite de couplage). Dans la mesure du possible, il est conseillé de la relier à l'alimentation (Vcc). On peut aussi, en fonction du schéma du circuit électronique considéré, la relier à une autre entrée utilisée ou à la masse, mais ceci augmente la consommation du circuit.

- ***En CMOS***

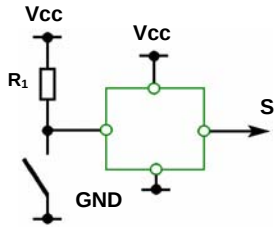
  Les entrées CMOS **ne doivent jamais** rester non branchées, mais raccordées au 0v, au V_{DD}, à une autre entrée utilisée ou mises à la masse ou encore reliées au V_{DD} à travers une résistance. En effet, comme pour la technologie TTL, une entrée non connectée capte les signaux parasites, mais du fait de la très forte impédance des circuits d'entrée, cela se traduit plus systématiquement par un dysfonctionnement entraînant une plus grande consommation et une surchauffe importante du composant.

Commande d'une entrée par un contact

Certaines applications imposent qu'une entrée puisse changer d'état en fonction de l'état d'un contact (bouton poussoir, interrupteur, contact de relais, capteur T.O.R., etc...).

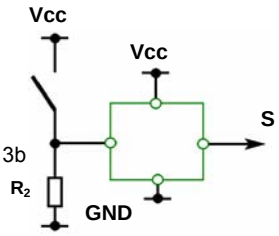
Deux schémas de commande représentés par les figures ci-dessous sont possibles :

Fig. 6.9.13a



cas a)

Fig. 6.9.13b



cas b)

- ***En TTL***

  Dans le cas a), c'est la résistance R_1 qui impose le niveau 1L sur l'entrée quand l'interrupteur est ouvert, et c'est la fermeture de l'interrupteur qui impose le niveau 0L.

  Dans ce cas, il n'y a pas de difficulté particulière, et l'on peut retenir par exemple $R_1 = 10\ k\Omega$.

  Dans le cas b), c'est la résistance R_2 qui doit imposer le niveau 0L sur l'entrée quand l'interrupteur est ouvert. La valeur de la résistance doit être définie en fonction du courant d'entrée maximum possible de l'entrée à l'état bas (I_{ILmax}) et le niveau de tension maximal correspondant l'état 0L, d'où :

  $$R_{2\max} \leq \frac{V_{IL\max}}{I_{IL\max}}$$

  Par exemple, pour un circuit du type LS une valeur de $1k\Omega$ est courante.

  C'est la fermeture de l'interrupteur qui impose sans difficulté particulière le niveau 1L.

- ***En CMOS***

  Du fait de l'impédance élevée des circuits d'entrée, on peut utiliser une valeur de résistance relativement élevée aussi bien dans le cas a) que dans le cas b).

  On peut retenir par exemple $R_1 = R_2 = 10K\Omega$

  Il faudra seulement veiller à assurer un débit minimum au contact du relais.

Découplage des alimentations

En pratique, lorsque les sorties totem pôle des circuits intégrés changent d'état, les deux transistors conduisent simultanément pendant un très bref instant pendant la transition (recombinaison), créant ainsi une pointe de courant sur l'alimentation pouvant occasionner, en raison des inductances parasites, des variations de tension importantes (Ldi/dt).

Pour atténuer ce phénomène, on utilise des condensateurs (10 nF à 1 µF), branchés en dérivation sur chaque circuit intégré, entre Vcc et la masse.