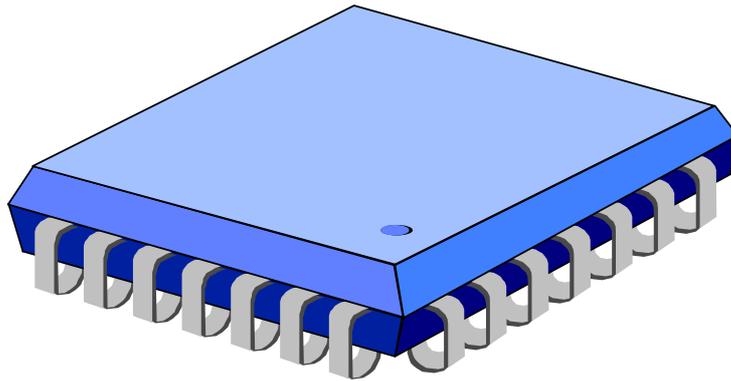
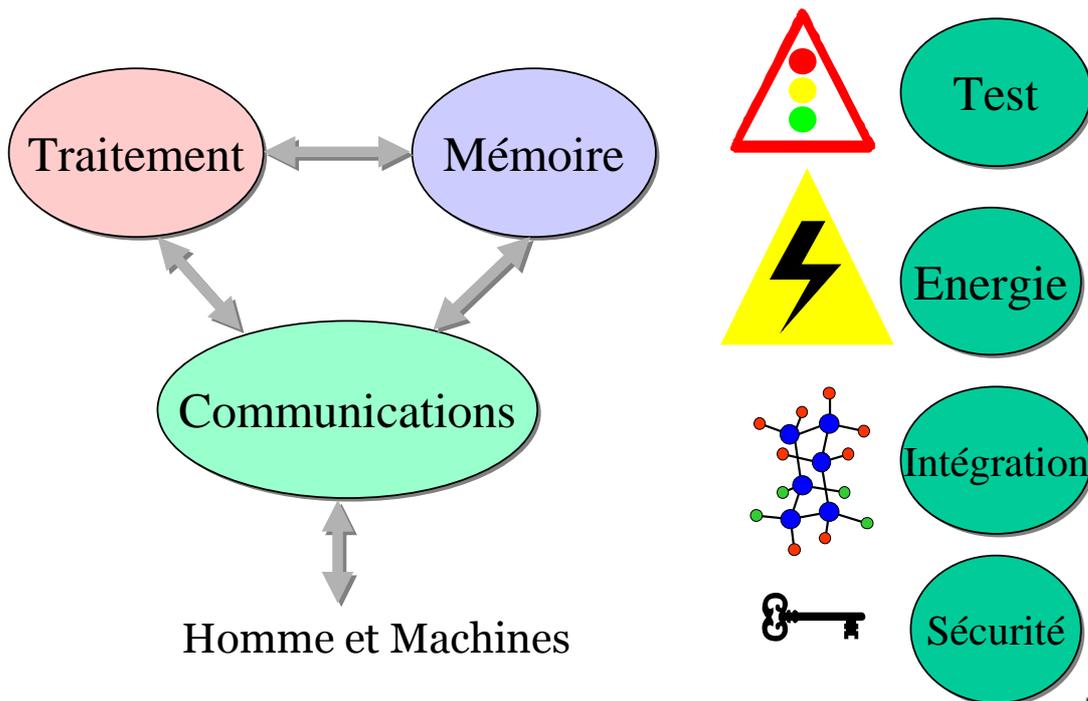


Bases d'architectures des systèmes électroniques



Architecture générique



A côté des 3 grands blocs fonctionnels (**traitement, mémoire et communication**) que composent une architecture de système électronique, se trouvent des blocs très importants pour la bonne marche du système :

- La partie **test** : qui garantit la fiabilité du système. Par exemple pour les ASICs il faut utiliser des structures de "Scan Path" pour générer des vecteurs vérifiant l'intégrité des équipotentielles après la fabrication du circuit.
- La partie **énergie**. Qui permet d'alimenter le système d'en façon optimale, c'est-à-dire en ayant la consommation de puissance minimale. Cette partie peut consister en une optimisation architecturale ou bien un bloc spécifique contrôlant dynamiquement l'alimentation de blocs fonctionnels comme c'est le cas dans les appareils portables avec les mises en veille des blocs à forte consommation.
- Le bloc **intégration** qui doit faciliter et/ou respecter la compatibilité de l'environnement dans lequel le système va s'insérer. Cette partie concerne principalement le bloc communication ou il est question de respecter des caractéristiques électriques et des protocoles de communication.
- Le bloc **sécurité** qui permet doit protéger à la fois au sens du piratage et au sens cryptographique quand le système traite des informations confidentielles (cas des cartes à puces)

Le types de signaux

■ Signaux de contrôle

- Horloge
 - Indispensable pour l'évolution des calculs
- Reset
 - Indispensable pour le démarrage
- Commandes
 - Générées pas les machines à états

■ Données

- Données effectives à traités
- Pointeurs de données (adresses)
- Indicateurs

• Tout système électronique dispose d'au moins 2 signaux de commandes que sont l'horloge et le Reset. Le **reset** fixe l'état initial des machines à états des contrôleurs, alors que l'**horloge** fixe le cadencement. Le fonctionnement synchrone permet de fiabiliser le traitement par la simple connaissance du chemin critique entre 2 registres (voir la leçon sur la logique synchrone)

• Les signaux de **commandes** sont générés et utilisés dans les contrôleurs qui sont des machines à états spécifiables par des graphes d'états.

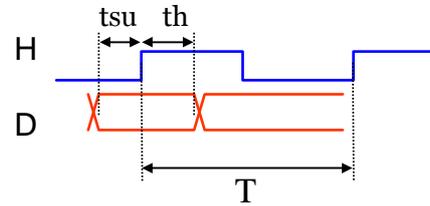
• Les **données** sont générées et utilisées dans les chemins de données « data path » composés d'opérateurs pilotés par des signaux de commandes



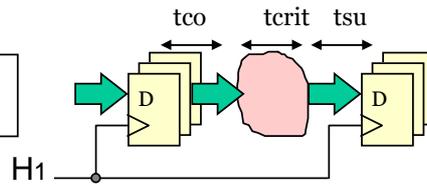
Rappel de la logique synchrone

1 seule horloge de période T
Calcul fiable si :

- $T > t_{co} + t_{crit} + t_{su}$
- $t_{co} > t_h$ (toujours garanti)

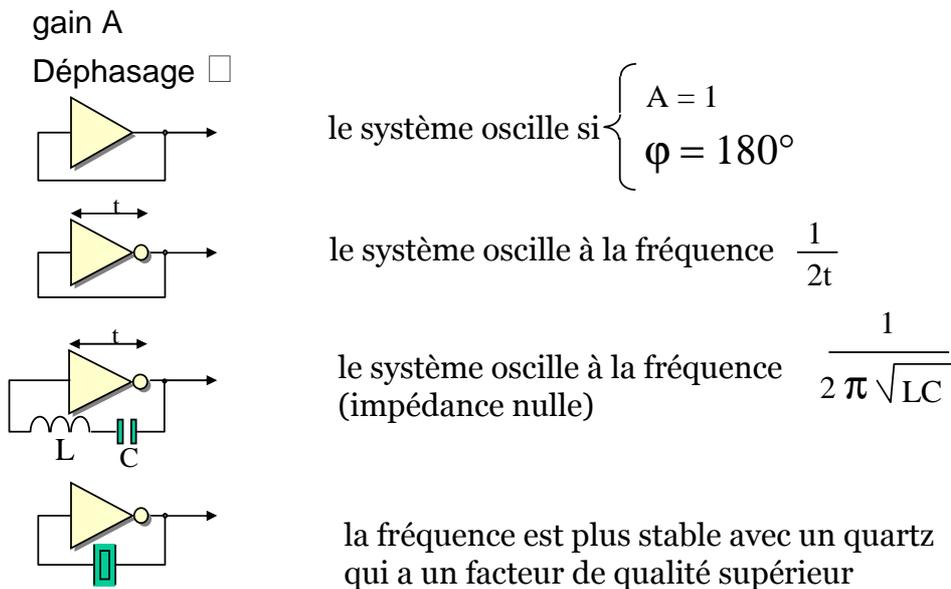


Il suffit de connaître T_{crit} !



Le fonctionnement synchrone permet de fiabiliser le traitement par la simple connaissance du chemin critique entre 2 registres (voir la leçon sur la logique synchrone)

Génération d'horloge



La théorie des systèmes asservis nous dit qu'un système en "boucle fermée" est instable si son gain en "boucle ouverte" est supérieur ou égal à 1 quand sa phase est de 180°. (critère de stabilité de Nyquist). Lorsque le gain est précisément égal à 1 à une certaine fréquence où la phase vaut 180°, le système oscille à cette fréquence. C'est ce principe qui est généralement utilisé pour les oscillateurs.

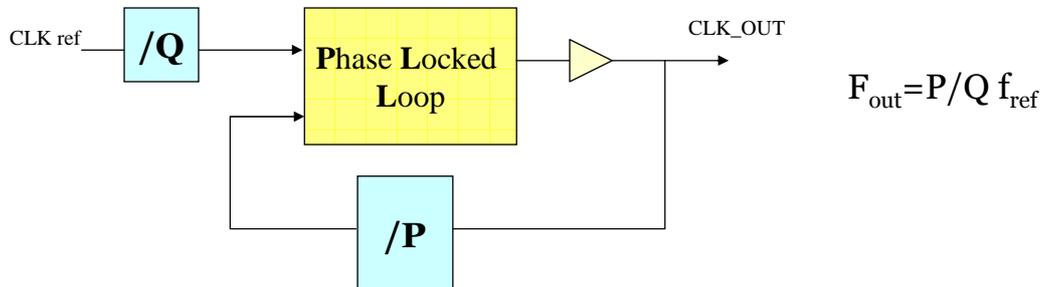
Lorsqu'on reboucle sur lui-même un inverseur de temps de propagation t , il est facile de montrer sur un chronogramme que le système oscille à la fréquence $1/2t$.

Les temps de propagation étant très dispersifs, il est difficile d'obtenir une fréquence précise. Un circuit résonant LC a une impédance nulle à sa fréquence de résonance, ce qui permet d'osciller à cette fréquence.

Les bobines utilisées pour l'inductance L ont une résistance R non négligeables, ce qui engendre une fréquence de résonance peu stable.

Le cristal de quartz a un modèle électrique correspondant à un circuit électrique LC avec un très bon facteur de qualité L/R . Ce dispositif est très utilisé comme générateur d'horloge dans les systèmes électroniques.

Principe : système asservi en phase



- ⇒ compensation le "skew" d'horloge
- ⇒ multiplication de la fréquence si diviseur dans la boucle de réaction

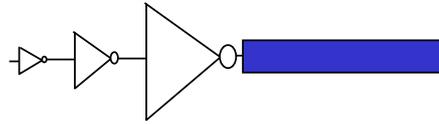
Les boucles à verrouillage de phase ou Phase Locked Loop **PLL** permettent de synthétiser un signal d'horloge à la fréquence voulue. L'horloge de référence peut être générée à l'aide d'un oscillateur à quartz. Elle est comparée à la sortie de la PLL. Ceci constitue un système asservi ayant pour but de minimiser l'erreur de phase entre l'horloge de référence et l'horloge générée.

Cette erreur est ensuite moyennée et pilote une VCO « Voltage Control Oscillator » générant l'horloge de sortie.

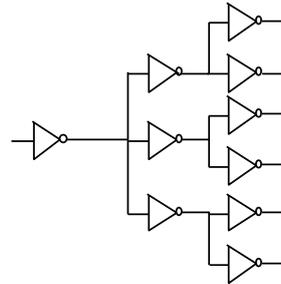
Quand la PLL est verrouillée, l'horloge de sortie à la même phase que l'horloge de référence. Ceci permet de compenser le « skew », ou déphasage de l'horloge utile, qui est dû à sa forte charge pour piloter les bascules D du circuit.

En utilisant des diviseurs de fréquence sur l'horloge de référence (facteur Q) et dans la boucle de retour de l'horloge de sortie (facteur P), la fréquence de l'horloge de sortie est égale à P/Q la fréquence de l'horloge de référence.

Rail d'horloge



Arbre d'horloge

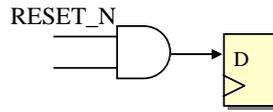


Une horloge a une charge très élevée dans les circuits. Elle peut atteindre des dizaines de milliers d'entrées de bascule. Les méthodes pour diminuer l'influence de cette charge sont soit :

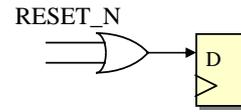
- Utiliser un amplificateur (buffer) basé sur des paires CMOS surdimensionnées et une équipotentielle d'horloge large donc moins résistive.
- Utiliser un arbre d'horloge générant des équipotentielles différentes mais équilibrées en charge de façon à avoir des « équitemporelles ».

Démarrage du système

RESET synchrone :



mise à 0

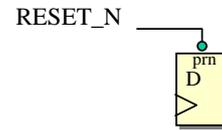


mise à 1

RESET asynchrone :

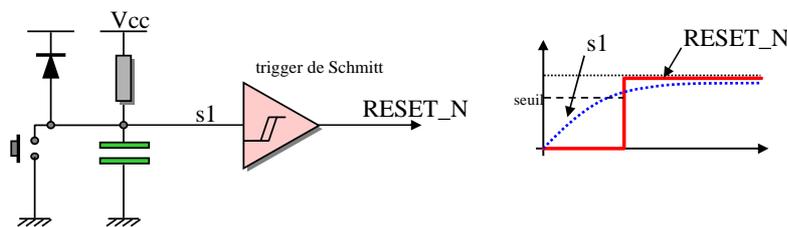


mise à 0



mise à 1

Exemple de circuit de RESET :



Le système électronique nécessite un signal spécifique d'initialisation : le RESET.

Il correspond à une équipotentielle particulière. Celle-ci peut agir d'une façon synchrone sur l'entrée D de la bascule mais il est plus judicieux d'utiliser une entrée asynchrone (mise à 1 : preset ou mise à 0 : clear) des bascules D. Les bibliothèques de cellules ASIC et FPGA disposent généralement de ce type de bascule.

L'utilisation des entrées asynchrones est réservée au RESET et il est vivement recommandé de ne pas utiliser de la logique sur ces entrées qui, au moindre parasite, risquent de réinitialiser la bascule.

Le RESET est généré automatiquement à la mise sous tension. Pour régler son temps d'activation nécessaire à l'établissement de l'alimentation, un circuit RC est utilisé.

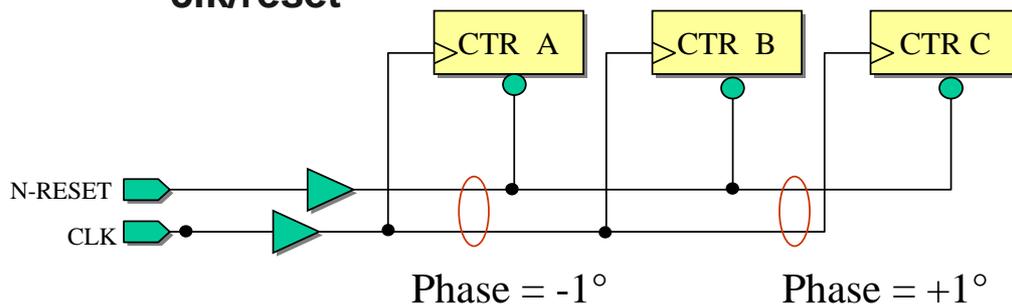
Un comparateur à hystérésis (Trigger de Schmitt) permet d'avoir un bel échelon sur la sortie RESET. L'hystérésis évite l'influence du bruit thermique qui peut donner lieu à des oscillations lorsque le RESET atteint le seuil de comparaison.

Certains systèmes disposent d'un bouton poussoir pour réarmer manuellement le système. C'est le cas pour certains ordinateurs PC.



Problèmes de démarrage

- **Oubli du Reset sur les machines à état**
 - la variable d'état est initialisée sans condition sur le Reset. La simulation ne détecte pas le problème et le circuit démarre dans un état quelconque.
- **Les contrôleurs peuvent ne pas démarrer en même temps à cause des différences de phase clk/reset**



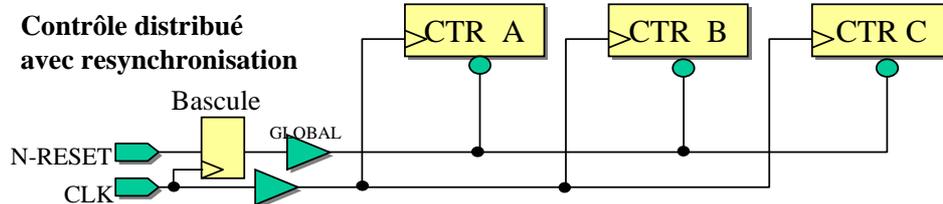
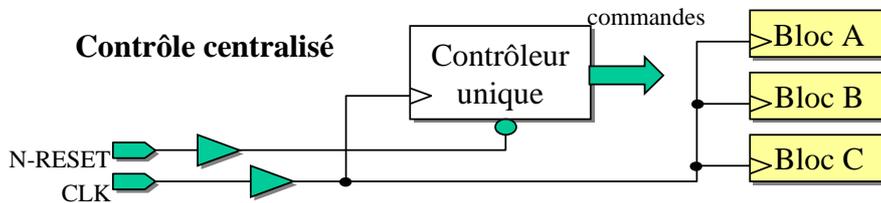
Certaines précautions doivent être prises pour démarrer correctement un système.

Un contrôleur (ou machine à états) doit impérativement être initialisé avec le RESET. Si ce n'est pas le cas, le système risque de démarrer dans un état qui n'est pas fonctionnel ou qui n'est pas celui de l'état initial de la machine à états.

Comme le RESET a une charge très élevée, il existe un risque sur la cohérence du démarrage des différents contrôleurs du système. Le RESET peut arriver juste avant le front d'horloge pour certains et juste après pour d'autres. Il existera alors un cycle d'horloge d'écart entre eux.

Stratégies pour bien démarrer

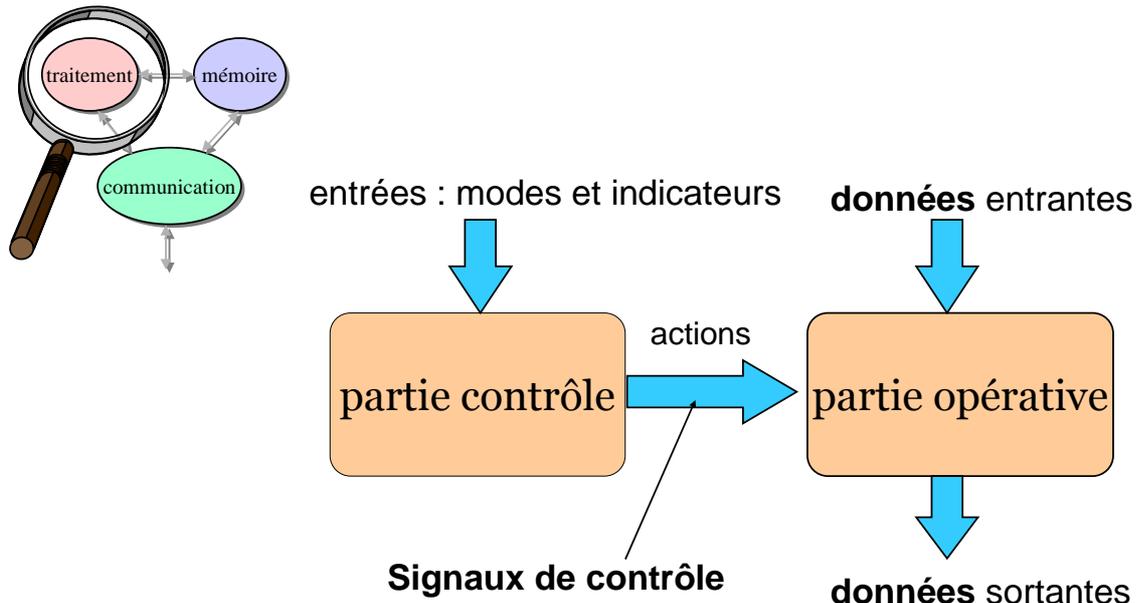
Utilisation systématique du Reset sur tous les processus de contrôle



Pour éviter le problème précédent; les solutions consistent à :

- Centraliser le contrôle, ce qui n'est possible que pour les petits système
- Auto synchroniser les contrôleurs entre eux.
- Synchroniser le RESET avant de l'utiliser. Cette solution simple est souvent utilisée. Il faut toutefois vérifier que le RESET a le temps de se propager durant une période d'horloge car il est très chargé.

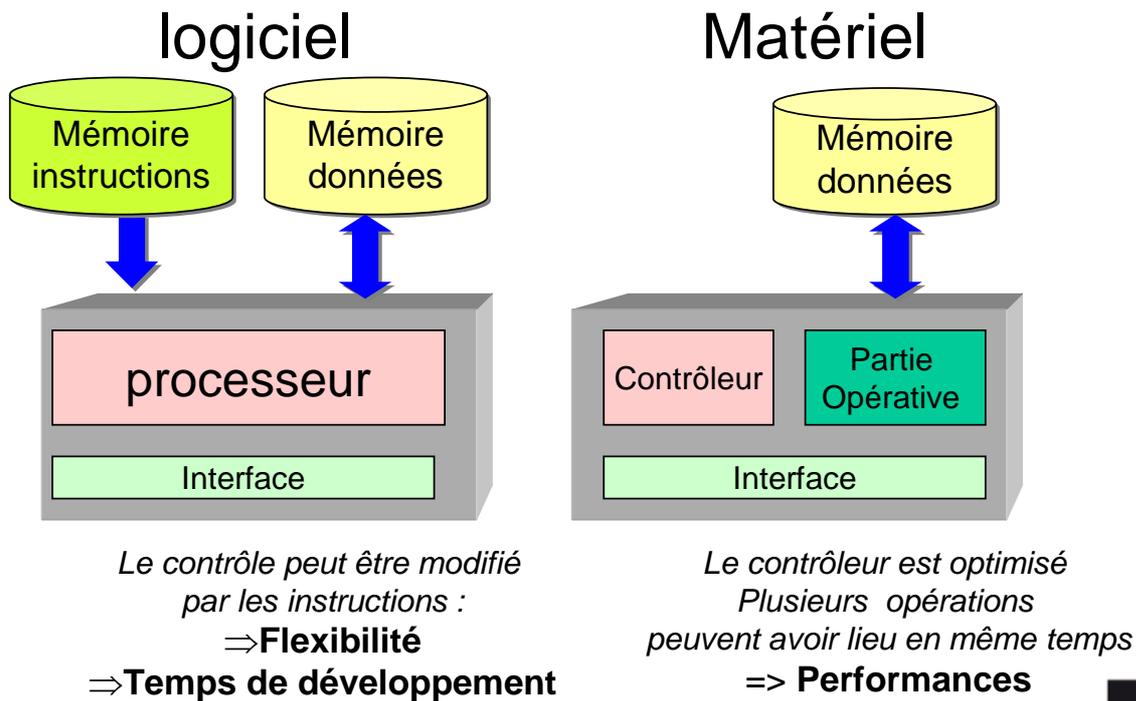
Unité de traitement



Le traitement est composé de la partie **contrôle** pour générer les signaux de commandes et de la partie **opérative** pour traiter les données.

Les systèmes électroniques de contrôle pur (robots, automates) n'ont pas ou peu de données à traiter. Ils génèrent des commandes pour les actionneurs électromécaniques.

Les types de traitement

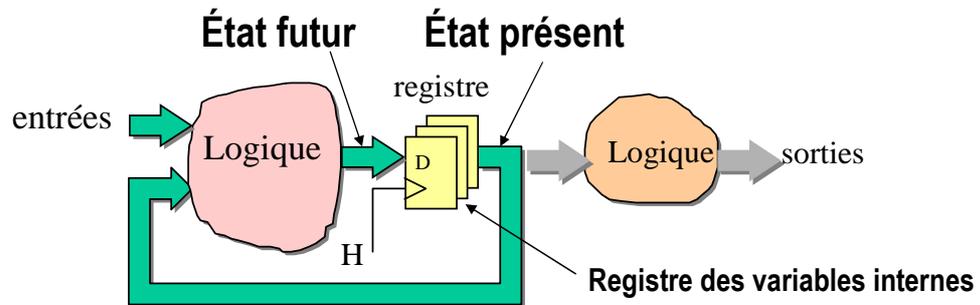


Les 2 types de traitement reposent avant tout sur des technologies **électroniques**. Un traitement **logiciel** utilise un processeur alors qu'un traitement **matériel** utilise un contrôleur spécialisé.

Le traitement **logiciel** est par essence très flexible car il est facile de changer les instructions et leur séquence. Le génie logiciel permet des temps de développement très courts.

Le traitement **matériel** permet d'obtenir des performances très supérieures du fait que le contrôle est spécifique et les opérateurs peuvent être utilisés en parallèle.

Génère les signaux de contrôle :
Machine séquentielle synchrone à états finis



Machine à états de Moore : $sorties = f(\text{état})$

Machine à états de Mealy : $sorties = f(\text{entrées}, \text{état})$

Rappel sur les machines à états qui constituent la partie contrôle des systèmes.

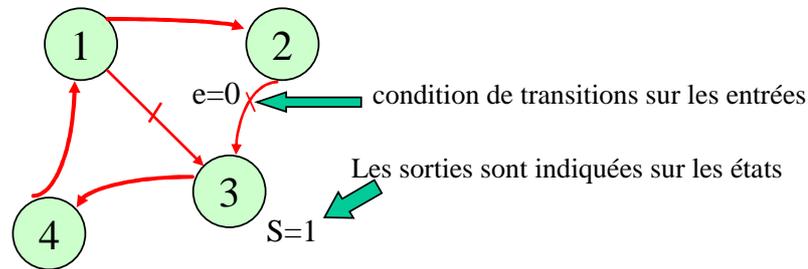
L' « intelligence » du contrôleur vient de son calcul séquentiel qui dépend des entrées et des états passés du circuit.

Une machine à états est un circuit séquentiel possédant des variables internes dont les valeurs illustrent l' **état** du circuit. Pour un système synchrone, ces variables internes sont stockées dans un registre d'états et elles servent d'entrées à la fonction de calcul de l'état futur qui va devenir l'état des variables internes au prochain front d'horloge.

Les sorties de la machine à états dépendent des variables internes (Machine de Moore) ou des variables internes et des entrées (Machine de Mealy)

Graphe d'états

spécification d'une machine à états par graphe :



Le graphe doit être :

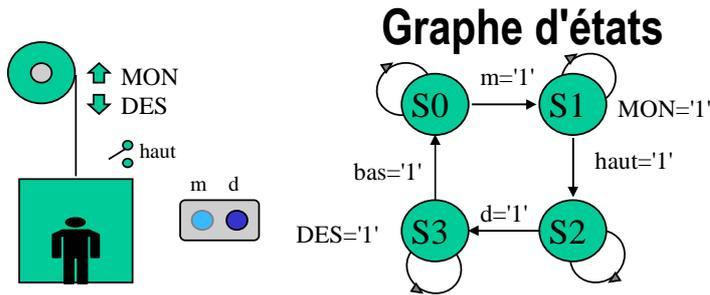
- **complet** : pour chaque état $\sum \text{conditions} = 1$
- **Non contradictoire** : $\sum_i \sum_{j \neq i} C_i C_j = 0$

La spécification par graphe nécessite des vérifications fonctionnelles garantissant la spécification du système. D'autre part le graphe doit respecter ces 2 règles :

- La **complétude** : Tout état actif doit donner lieu à un passage à un autre état (ou passage d'un « jeton »). Il faut donc que toutes les combinaisons d'entrée définissent le futur état sinon le jeton risque d'être perdu et la machine peut se bloquer.
- La **non-contradiction**. La passage d'un état doit se faire vers un unique état. Le fait d'avoir plusieurs jetons simultanément rend le système non fiable.

Dans les systèmes complexes, les machines à états peuvent être architecturées de façon hiérarchique.

Exemple de machine à états



Codage des états

"one hot":
 1 état sur 4 bits Q3 Q2 Q1 Q0
 S0=0001
 S1=0010
 S2=0100
 S3=1000

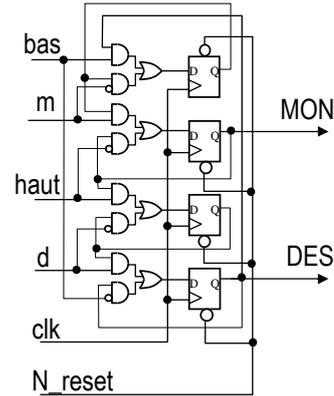
Équations des sorties :

MON = Q1
 DES = Q3

Équations des états futurs :

D1 = Q0.m + Q1. /haut
 D2 = Q1.haut + Q2. /d
 D3 = Q2.D + Q3. /bas
 D0 = Q3.Bas + Q0. /m

Schéma logique

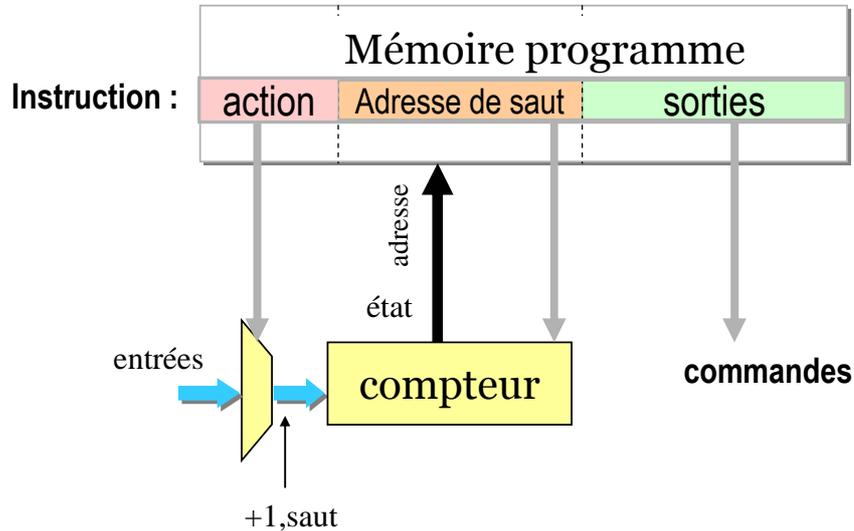


Dans cet exemple, le codage « One hot » est choisi. Les équations illustrent la facilité d'obtention des résultats avec ce type de codage, au détriment du nombre de bascules.

L'utilisation des langage VHDL ou Verilog avec des outils de synthèse évite au concepteur les étapes de choix des bits et de codage.

Machine à état programmable : Séquenceur

Le séquenceur ou "Automate" est une machine à états de MOORE où un compteur indique l'état courant



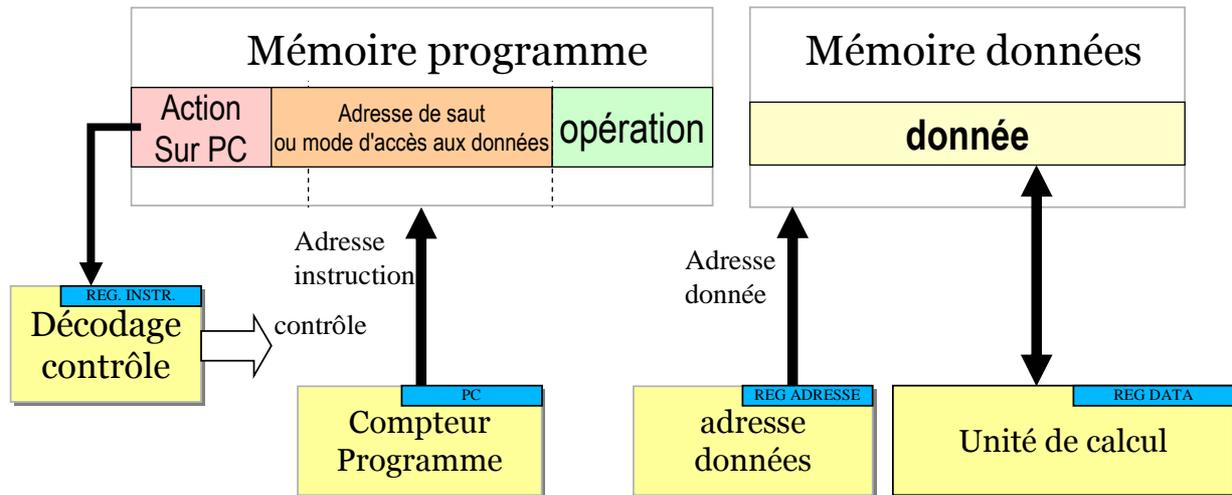
Le séquenceur est une machine à états de Moore générique dont le registre d'état est un compteur qui peut être incrémenté ou chargé par une nouvelle valeur. Ce compteur pointe dans une mémoire sur une « instruction » qui contient l'action à effectuer (+1 ou saut) en fonction des entrées et l'adresse de saut dans le cas d'un saut. L'instruction a également un champ pour les sorties servant à piloter des actionneurs extérieurs.

L'ensemble des instructions s'appelle un programme et le compteur s'appelle le compteur de programme ou PC. Cette architecture s'inspire de la fameuse machine de Turing où le ruban est remplacé par une mémoire électronique.

Cette architecture de séquenceur s'appelle aussi automate programmable. Elle est très utilisée en électromécanique pour piloter des robots industriels.

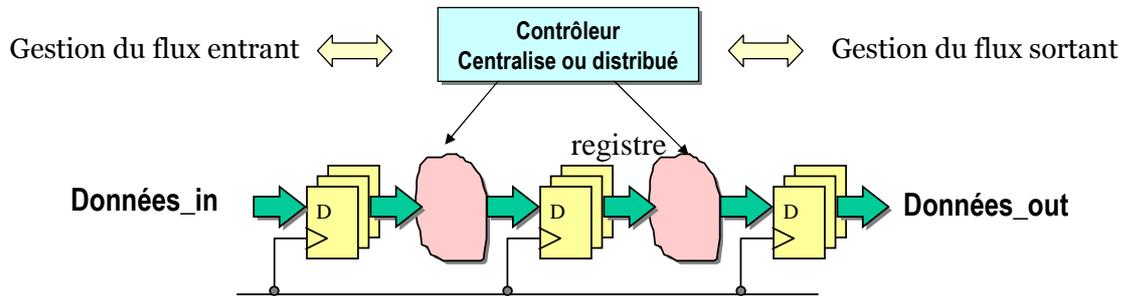
Processeur

Architecture à mémoire unique (Von Neumann) ou séparée (Harvard) :



L'architecture Von Neuman oblige à aller chercher successivement en mémoire l'instruction puis la donnée. En disposant de 2 mémoires respectives pour les instructions et les données, il devient possible de paralléliser les accès et donc d'augmenter les débits de calcul. C'est le but de l'architecture Harvard qui est utilisée dans tous les processeurs performants.

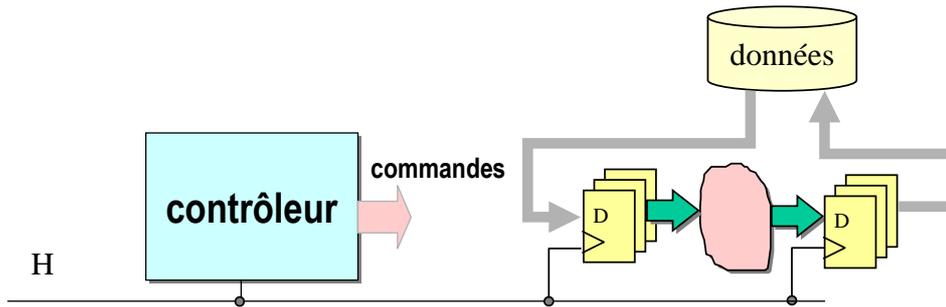
Partie opérative « data flow »



- Traitement en pipeline
- Contrôle simple

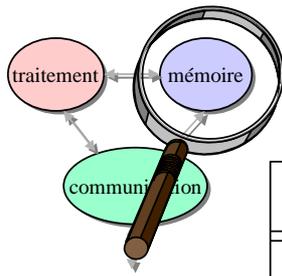
Dans le cas d'un traitement d'un flux de données, les structures en pipeline sont appropriées. Le contrôle est en général assez simple. Il peut lui-même être distribué en pipeline et être associé à chaque opérateur.

Partie opérative « Control Driven »



- Contrôleur centralisé complexe
- séquentialisation des opérations
- Utilisé dans les processeurs

L'approche "control driven" consiste à piloter le cœur opératif par un contrôleur unique qui décide du type d'opération et de l'aiguillage des données. Cette approche est celle qu'on retrouve dans les calculateurs génériques et les microprocesseurs avec l'unité arithmétique et logique.



Principaux types

Type	Accès	Vitesse	Densité	volatilité	techno
DFF	direct	*****	-	Y	
SRAM	adresse	***	++	Y	
DRAM	adresse	**	+++	Y	
FLASH	adresse	*	++	N	

Les principales mémoires sont listées dans le tableau. Chaque type a son propre intérêt :

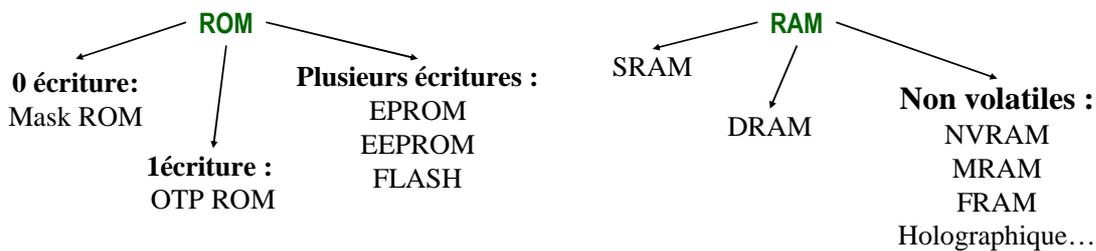
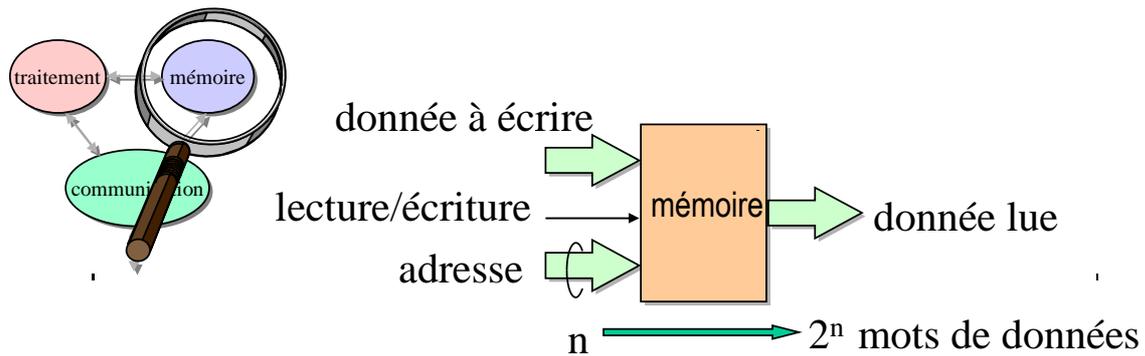
DFF : accès individuel et rapidité

SRAM : rapidité et densité

DRAM : densité

FLASH : non volatilité

Blocs mémoire



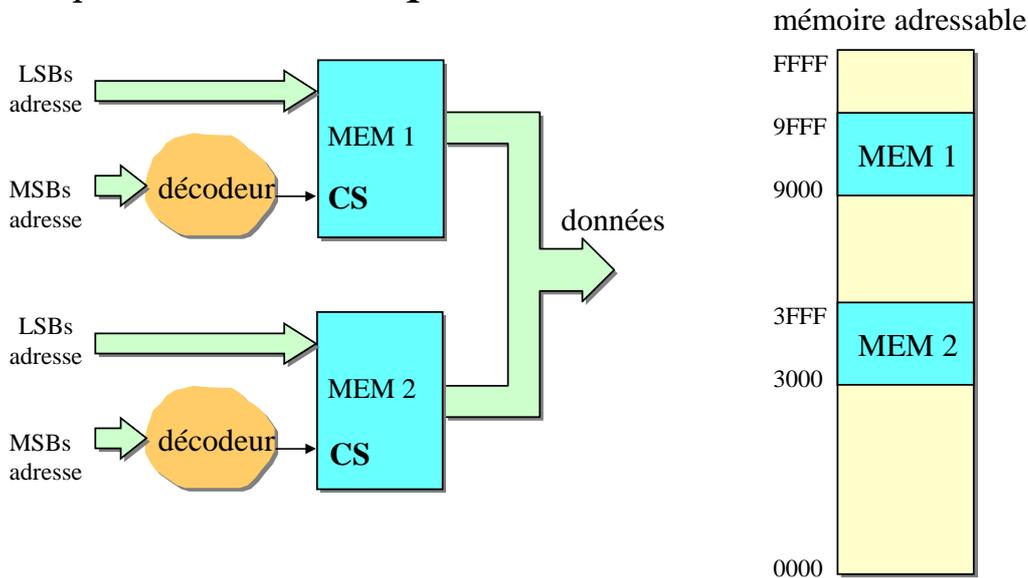
La mémoire électronique stocke 2^n mots de m bits. Un seul mot est accédé à la fois. Chaque mot est donc associé à une adresse. La mémoire agit comme une fonction combinatoire sur les bits d'adresse en lecture. En écriture il faut de plus activer le signal de commande d'écriture et lui fournir la donnée à écrire.

Les mémoires récentes sont synchrones et ont une horloge en entrée. Cela simplifie beaucoup le contrôle qui doit juste satisfaire la contrainte du chemin critique par rapport à la période d'horloge : $T_{crit} < T_h$

Mappe mémoire

indique les champs d'adresse mémoire dans un système

Chaque boîtier a son "**Chip Select**"

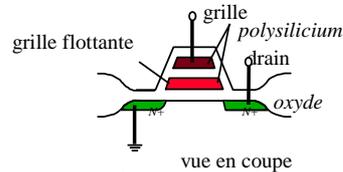


Un système électronique utilisant un champ mémoire, comme un microprocesseur, doit avoir une mappe mémoire correspondant aux zones d'adresse des différents éléments.

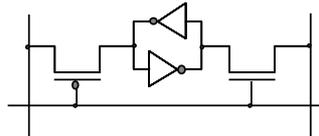
Une façon de dissocier 2 éléments, comme par exemple des boîtiers mémoire, est de piloter différemment un signal d'entrée souvent appelé "chip select" CS. Ce CS fait l'objet d'un décodage sur les bits de fort poids MSBs de l'adresse alors que le boîtier reçoit les bits de faible poids.

Si on considère un système disposant de 16 bits d'adresse, la mappe mémoire se situe entre 0x0000 et 0xFFFF. Si on veut qu'une mémoire de 4K octets (donc 12 bits d'adresse) soit située dans la zone 0x9000 à 0x9FFF, Le CS doit correspondre au décodage des 4 MSBs à la valeur 9.

EEPROM, FLASH : basée sur les transistors à grille flottante



SRAM : CMOS



DRAM : capacité grille d'un transistor MOS

=> nécessité de rafraîchissement mais capacité x4

Les principales technologies sont la mémoire **ROM** ou plus récemment **FLASH**, la RAM statique **SRAM** et la ram dynamique **DRAM**.

La mémoire FLASH utilise des transistors double grille avec grille flottante. Les charges injectées dans la grille flottante sont capturées et peuvent rester quelques années. La mémoire est dite non volatile.

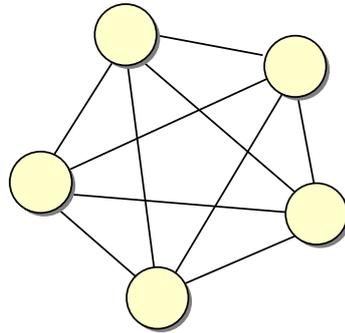
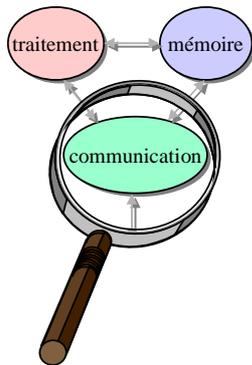
La mémoire SRAM repose sur un système stable assuré avec 2 inverseurs rebouclés sur eux-mêmes. 2 transistors de commande de part et d'autre permettent l'écriture et la lecture.

La mémoire DRAM est la capacité grille d'un transistor MOS. Cette petite capacité se décharge assez vite et un système de rafraîchissement est nécessaire.

Les mémoires SRAM et DRAM sont volatiles. La mémoire DRAM est optimale en terme de taille (1 point=1 transistor) mais est plus lente.

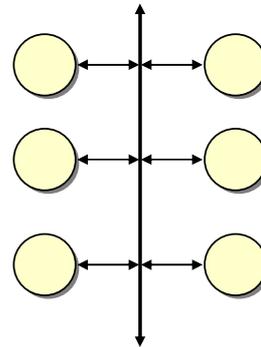
De gros progrès ont été fait sur les interfaces des mémoires de façon à accélérer les débits. Par exemple la mémoire FLASH est écrite par secteurs de données. De même la mémoire DRAM, assez lente en accès aléatoire devient rapide si toute une rangée est préchargée dans une mémoire rapide et accédée en rafale.

Canaux de transmission locaux



point à point

$$C_2^n = \frac{n(n-1)}{2} \text{ connexions}$$



Bus

1 seule connexion

- nécessite un arbitrage
- transfert simultané entre 2 éléments seulement

Il existe de nombreuses façons de faire communiquer les éléments d'un système électronique entre eux. Les liaisons "**point à point**" sont simples à mettre en œuvre mais il faut de nombreuses liaisons pour chaque élément. La contrainte de modularité et d'intégration favorise les liaisons de type "**BUS**" ou un ensemble d'équipotentiels est commun à tous les éléments. Le prix à payer pour ce gros avantage est le fait qu'une seule communication est possible à la fois. D'autre part si plusieurs éléments veulent prendre le bus, il faut un arbitre qui décide des priorités à donner et indique le "**Maître**" du bus.

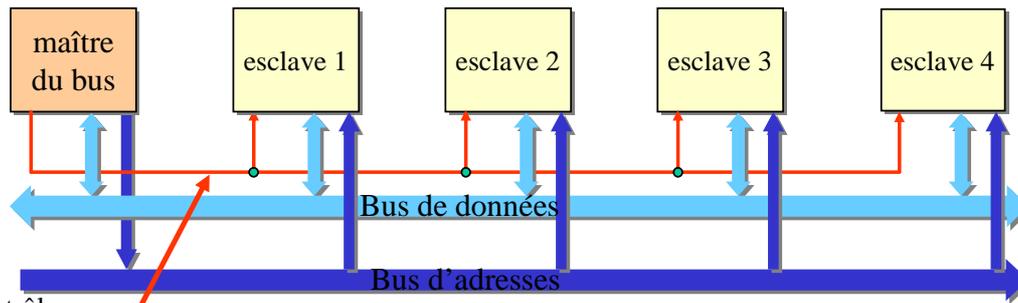
Transfert sur un bus

■ Nécessité d'associer un champ d'adresse à un élément.

- chaque élément a son propre décodeur d'adresse générant un "CS"

■ Espace adressable

- soit dans la mappe mémoire
- soit dans la mappe E/S (n'existe pas toujours)



bus de contrôle :

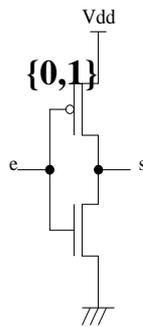
- horloge, activité du cycle de transfert, direction des transferts (lecture/écriture), gestion du protocole d'échange

Les éléments sur un bus ont leur propre champ d'adresse correspondant à l'activation de leur "Chip Sélect" . Quand un maître dispose du bus, il doit présenter l'adresse de l'"esclave" visé sur le bus d'adresse, le type de transaction sur le bus de commandes, et les données en écriture sur le bus de données (ou attendre les données s'il s'agit d'une lecture).

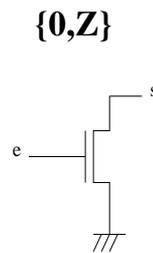
Sorties 3 états et "open drain"

Modèles fonctionnels des sorties de portes logiques :

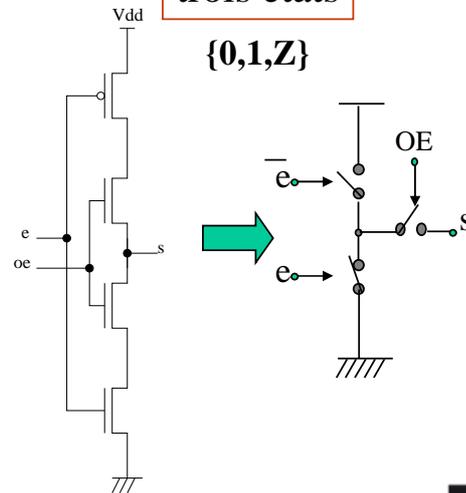
sortie classique



open drain



trois états



Les sorties d'un système électronique se font d'une façon traditionnelle en CMOS avec un réseau de transistors P tirant à 1 et un réseau dual de transistor N tirant à 0.

Pour relier les sorties entre elles, ce qui n'est pas possible en sortie CMOS, il peut être intéressant d'enlever le réseau de transistor P. La sortie prend 2 états 0 et Z; Le Z symbolise l'état "haute impédance" où la sortie n'est forcée à aucun potentiel. Ce type de sortie s'appelle sortie "**Open Drain**".

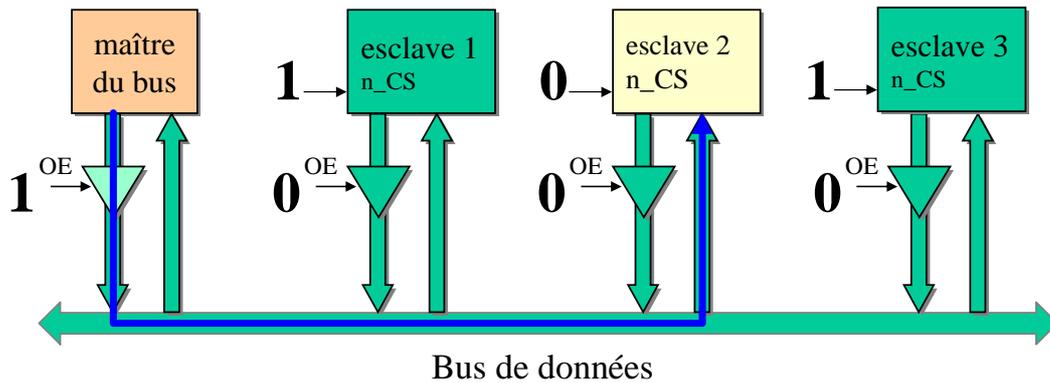
Une autre type de sortie est la sorties **3 états** qu'on peut symboliser comme un sortie CMOS avec un transistor supplémentaire en série pour permettre un 3ème état qui est Z. Cette sortie a donc un signal de commande "Output Enable" OE autorisant soit la sortie CMOS classique à 2 états, soit la mise en état haute impédance Z. Ce type de sortie se dit aussi "buffer 3 états"

Intérêt du 3 états pour les bus

Nécessité d'une logique 3 états pour assurer la bidirectionnalité du bus :

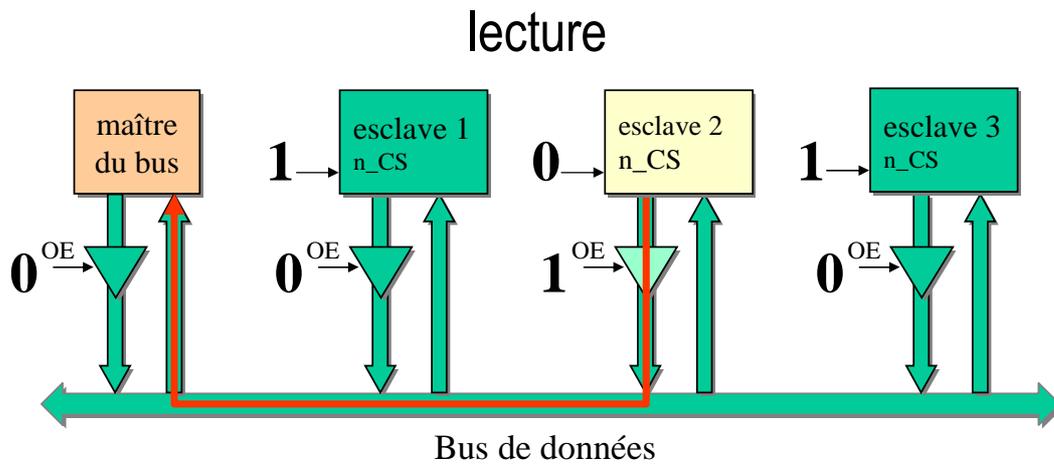
→ 1 seul buffer doit être actif sinon conflit.

écriture



Les sorties 3 états permettent les transferts bidirectionnels de données sur un bus.
Quand un maître écrit à l'esclave n°2, seul le signal OE du maître doit être actif.

Intérêt du 3 états pour les bus



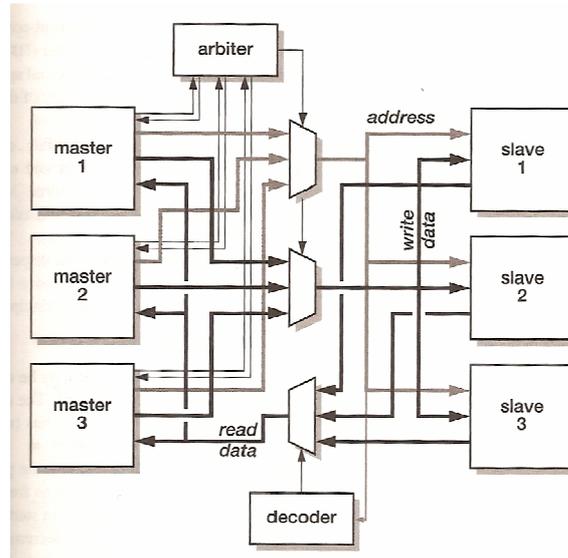
en résumé :

OE maître = écriture
OE esclave_i = lecture ET CS_i

Quand le maître lit l'esclave n°2, seul l'OE de la sortie de l'esclave 2 est actif.

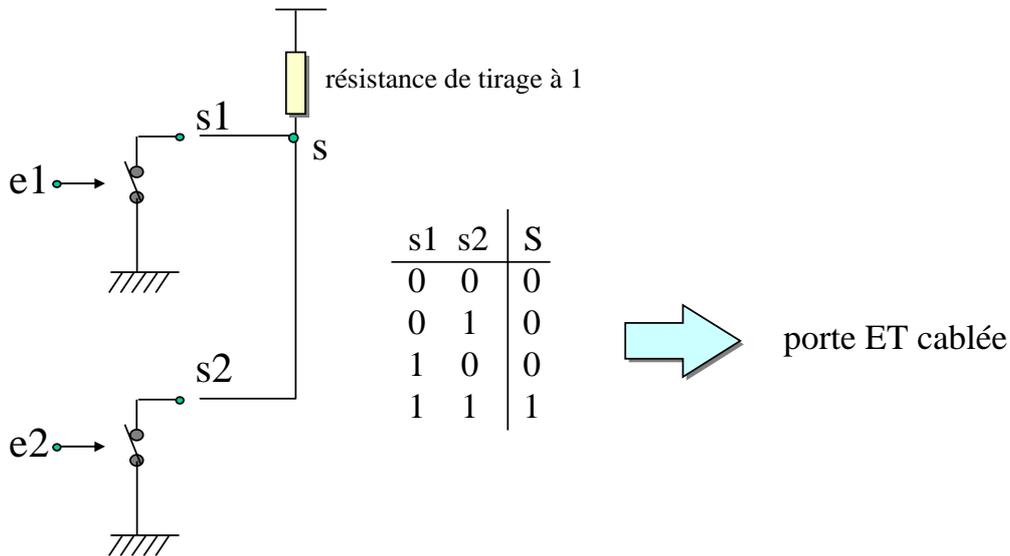
Bus multiplexé

- Permet d'éviter les conflits potentiels
- Pas besoin de 3-états
- Exemple : AMBA



Intérêt de l' open drain

possibilité de connecter les sorties entre elles :

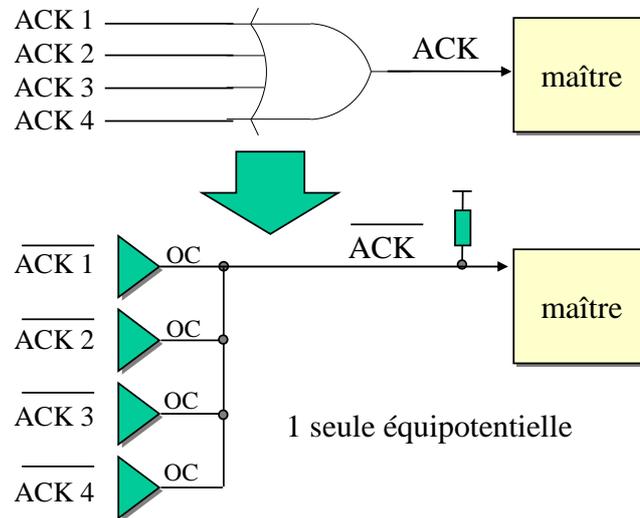


Les sorties "Open Drain" permettent des connexions de plusieurs sorties. Si on relie la sortie à une résistance connectée à la tension d'alimentation V_{dd} , l'état Z correspondra à un potentiel V_{dd} (donc niveau logique '1') par le biais de cette résistance de "tirage" ou "pull up".

Le signal S résultant de cette connexion est équivalent à une porte ET dont les entrées sont S1 et S2. Cette propriété est très intéressante car elle permet de construire une porte ET avec beaucoup d'entrées sans porte CMOS. On parle de "ET câblé" pour le différencier de la porte CMOS.

Exemple de sorties en open drain

signal d'acquittement d'un bus

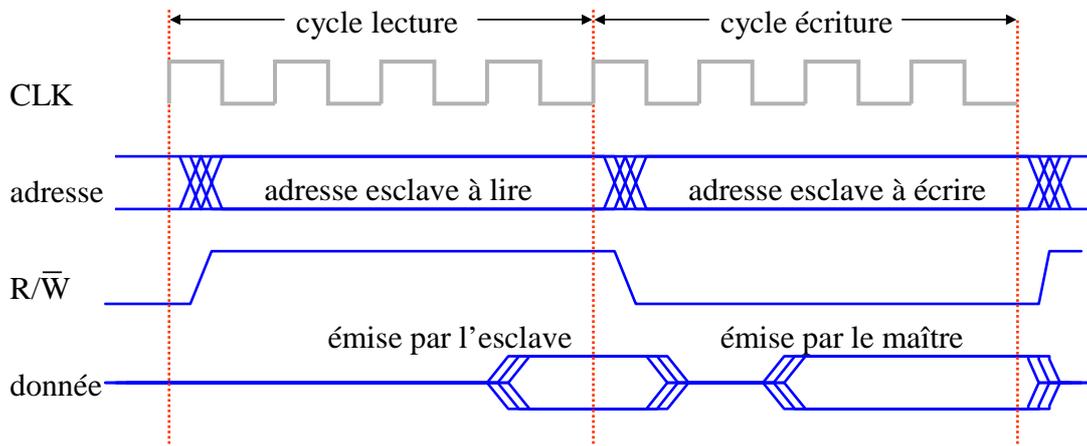


Dans un bus ayant un protocole asynchrone, le nombre de signaux "accusé de réception" peut être très grand. De même un processeur peut avoir une ligne d'interruption générée par un grand nombre d'éléments périphériques. La fonctionnalité demandée est un OU logique faisant la somme des signaux issus de chaque élément. Ce OU doit avoir un nombre d'entrées variable et important.

L'utilisation des sortie "Open Drain" permet de réaliser un ET câblé qui devient un OU câblé si tous les signaux sont en logique inverse (utilisation du théorème de De Morgan).

Protocole de communication synchrone

L'horloge rythme les échanges entre maître et esclave.
Le nombre de cycle est constant

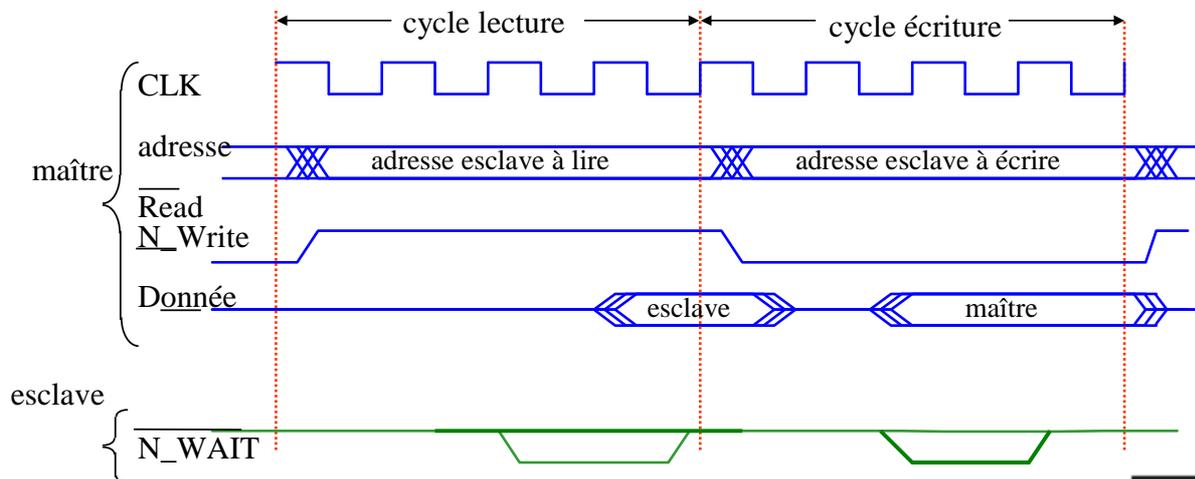


Le protocole utilisé sur un bus est celui qu'on retrouve dans tous les systèmes de communication. Il est soit direct, c'est à dire sans préoccupation d'acceptation de l'élément adressé, ou **asynchrone** c'est-à-dire avec un échange "poignée de main" ou "Handshake" pour s'assurer de la fiabilité de la transaction.

Dans le cas de l'échange direct, on parle plutôt d'échange **synchrone** en électronique car l'accès dure un nombre constant de cycles d'horloge. Les 2 éléments en communication se sont mis d'accord à l'avance sur le nombre de cycles d'horloge nécessaire à la transaction.

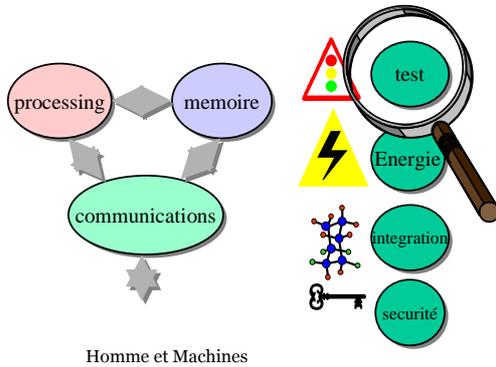
Protocole de communication asynchrone

L'échange est assuré par un processus de "Handshake", l'esclave renvoie au maître soit un signal pour acquiescer ou ralentir l'échange.



Le protocole asynchrone nécessite un signal "accusé de réception" ou "feu vert" de l'élément adressé vers le maître de façon à s'assurer que la transaction va bien se passer. Il permet à certains éléments lents de faire attendre le maître. Il existe 2 philosophies; Soit ce signal en retour dit : "Je suis prêt" , auquel cas il faut toujours que l'esclave génère ce signal à chaque transaction. Ou bien le signal en retour signifie "attendez !" auquel cas il n'est pas nécessaire que l'esclave génère le signal s'il est assez rapide.

Attention les protocole Asynchrone ne veut pas dire qu'il n'y a pas d'horloge. Dans l'exemple ci-dessus, il existe toujours une horloge car la logique est Synchronique mais la communication est Asynchrone.



■ Exhaustivité du test

- Electrique (testeur sous pointes)
- Optique
- Boundary scan test par JTAG pour les cartes
- Scan chain pour les ASICs

A côté des 3 blocs fonctionnels : Traitement, Mémoire et Interfaces, les blocs annexes ont d'un système électronique numérique visent à améliorer la fiabilité, la consommation faible, la compatibilité et la sécurité.

Le bloc de TEST est indispensable. Il permet de garantir la qualité du produit pour :

- Faire des économies au niveau de la maintenance
- Avoir une image de produit de qualité (impact marketing fort)

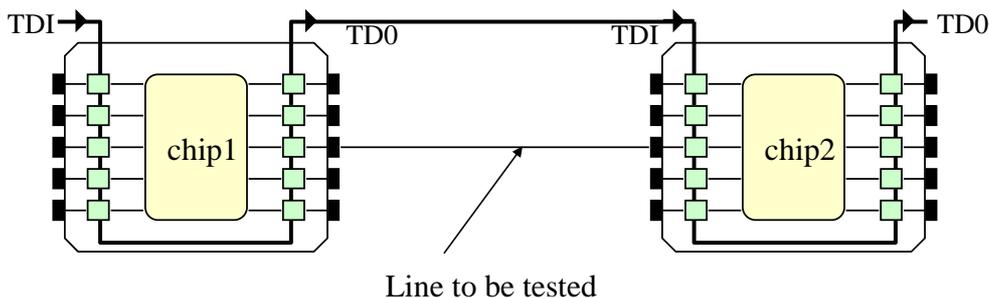
Pour un système électronique, l'architecture peut être modifiée de façon à rajouter des blocs de test spécifiques. Les stratégies de test reposent à la fois sur des tests fonctionnels (très longs) et sur des tests d'intégrité électrique vérifiant que chaque équipotentielle ne soit pas coupée en ou court-circuit. Une technique courante de vérification de l'intégrité électrique consiste à chaîner les bascules D de façon à obtenir un grand registre à décalage "scan chain". Par ce biais il est possible de rentrer des valeurs au circuit à tester puis de récupérer les résultats sur les bascules échantillonnant le résultat. Les outils de CAO sont capables de générer automatiquement les vecteurs de test pour qu'un maximum d'équipotentielles passe d'un niveau logique à l'autre et que le test soit le plus exhaustif possible.

Port JTAG

Le JTAG est un contrôleur interne ou "TAP controller" qui permet

- Test de la connectivité (par Boundary Scan testing)
- Accès aux ressources internes des processeur pour le débogage
- Chargement de netlists pour les FPGAs
- Fonctions personnalisées

exemple de Boundary Scan Test :

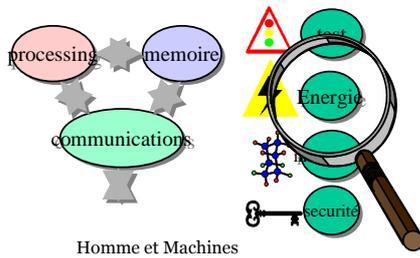


Le port JTAG "Joint Test Action Group" est un contrôleur gérant un flux de données série (TDI en entrée et TDO en sortie) permettant d'effectuer diverses opérations dont celle du test pour lequel il a été conçu à l'origine.

L'opération de test appelée "Boundary Scan test" permet de tester la connectivité des composants assemblés entre eux. Chaque composant a ses broches d'entrée/sortie reliées par un grand registre à décalage qui est lui-même relié par les broches TDI et TDO à un autre circuit. Le tout constituant un énorme registre à décalage. Les contrôleurs JTAG ou "TAP controller" effectuent des opérations d'écriture ou de lecture dans ces registres. Il est donc possible de forcer un niveau logique sur une broche de sortie et de lire le niveau sur la branche d'entrée de la même équipotentielle d'un autre composant. Si les niveaux coïncident, l'équipotentielle est considérée intègre.



Contrainte de consommation d'énergie



■ Consommation dynamique

En CMOS L'essentiel de la consommation provient des courants de charge et décharge des capacités réparties

$$P = \frac{1}{2} CV_{dd}^2 f$$



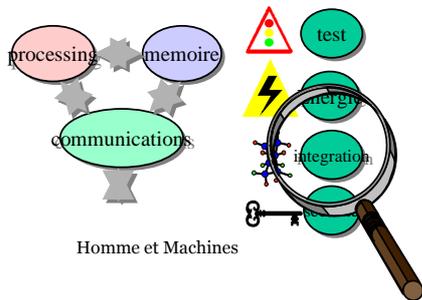
■ Consommation statique

- Due aux courants de fuite. Quasi nulle dans les technologies >130nm mais devient non négligeable

La consommation est majoritairement due à la décharge et charge des capacités lorsque les transistors d'une paire CMOS commutent. La puissance moyenne dissipée est donc proportionnelle à la fréquence moyenne d'activité des équipotentielles donc à la fréquence d'horloge du système synchrone.

La puissance statique est quasi-nulle. Dans les technologies inférieures à 100nm la proportion de la consommation statique due aux courants de fuite devient perceptible.

Contrainte d'intégration sur carte



■ Grande Densité d'intégration sur circuit imprimé

- 2*n couches de routage
- Composants CMS sur les 2 faces
- Boîtiers très petits avec de + en + de broches

■ Problèmes d'Intégrité du signal

- Interférence entre lignes (émission/réception)
- Adaptation de lignes (réflexion)

L'intégration dans un environnement se fait à tous les niveaux : mécanique, électrique, protocolaire, logiciel,... Ces contraintes sont à considérer depuis la spécification du système jusqu'à l'assemblage final.

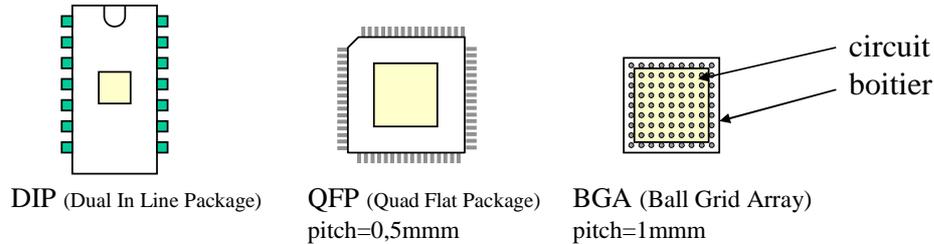
Concernant l'assemblage sur circuit imprimé qui est constitué de plusieurs couches de routage en cuivre, il existe de fortes contraintes de place, d'intégrité du signal et de rayonnement électromagnétique (CEM). Il est donc très important de vérifier l'assemblage. Il faut d'abord effectuer le placement et le routage des composants puis analyser statiquement les critères de transmission sur ligne, d'interférences et de rayonnement pour vérifier que le niveau de perturbations n'est pas critique.

Ce sont les mêmes problèmes rencontrés dans les circuits intégrés mais avec des dimensions moindres.

Encapsulation des circuits

■ Plus d'E/S* en moins d'espace

- Réduire l'espace entre broches "pitch". : 2,54mm => 0,5 mm
- Utiliser une matrice de broches plutôt que sur la périphérie du boîtier
- Utiliser les circuits "FlipChip" où les E/S sont internes à la puce



■ Dissipation de puissance

- Boîtier céramique
- Système de refroidissement

* Loi de Rent:

$$N = K.G^{\beta}$$

N=nombre d'E/S

K=nombre d'entrées/portes

G=nombre de portes

β =constante de Rent dépendant de l'application~0,6

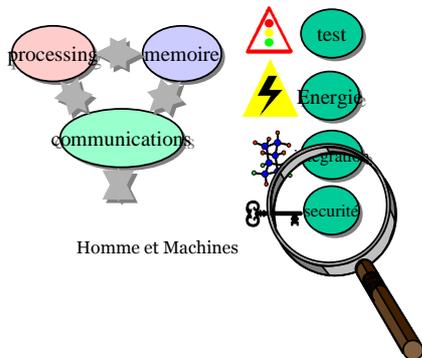
Les boîtiers ont beaucoup évolué pour tenir compte de l'augmentation de la densité de portes et de l'augmentation du nombre d'entrées/sorties E/S, comme l'indique la loi empirique de Rent.

Les premiers boîtiers étaient rectangulaires et disposaient de broches sur 2 côtés (boîtiers DIP Dual In Line Package). Le fait que les broches traversent l'isolant du circuit imprimé diminue la "routabilité" car toutes les couches du circuit imprimé sont pénalisées. Pour remédier à cela les boîtiers ont évolué pour avoir des broches soudables uniquement en surface (composants montés en surface CMS, Surface mounted devices SMD)

Avec l'augmentation des E/S, les broches se sont installées sur la périphérie de boîtiers carrés (Quad Flat Package, 240 broches maximum), donc avec un meilleur facteur de forme. La faible distance entre broches de l'ordre de 0,5mm diminue la fiabilité des connexions.

Les boîtiers récents disposent de broches disposées matriciellement sous le circuit permettant à la fois une augmentation du nombre de broches et une distance entre broches supérieure à 0,5mm (BGA : Ball Grid Array)

Contrainte de sécurité



■ Eviter le copiage

- Utiliser du matériel dédié

■ Protéger les communications

- **Cryptographie** : Algorithmes très efficaces. Matériel vulnérable car fuit de l'information par le biais de canaux physiques.
- Utiliser des structures résistantes aux "attaques"

La sécurité est à 2 niveaux :

- La protection contre les copiages
- La protection contre l'intrusion ou l'espionnage des messages véhiculés

Pour éviter le copiage, le simple fait de choisir la conception d'un circuit personnalisé constitue une protection par rapport à une solution logicielle. Pour les FPGAs, il existe la possibilité de chiffrer la netlist.

Pour la protection des communications, les algorithmes cryptographiques sont largement utilisés. Il existe cependant des vulnérabilités potentielles au niveau matériel du fait que les fonctions électroniques dissipent de l'énergie reflétant une partie du secret à protéger. Ces "attaques" sur le matériel sont difficiles à réaliser et peuvent être contrecarrées par des architectures résistantes en cours de mise au point dans des laboratoires de recherche.