Architecture et Système

Stefan Schwoon

Cours L3, 2023/2024, ENS Cachan

Multiplexeur

Données :

vecteurs $x_0, \ldots, x_{k-1} \in \{0,1\}^m$, pour $k = 2^n$

indice de sélection s entre 0 et $k-1$, codé par n bits

Sortie: x_s

Applications:

Lire un mot dans la mémoire en spécifiant son adresse.

Choisir entre plusieurs sources de données (p.ex. périphériques).

Obtenir une valeur dans un tableau pré-calculé.

Multiplexeur : Exemple

Soit $m = n = 1$, du coup on possède deux bits x_0, x_1 et un bit de sélection s_0.

Solution: on réalise la fonction $(x_0 \wedge \neg s_0) \vee (x_1 \wedge s_0)$.

Soit $m = 1, n = 2$, du coup on posséde $x_0, \cdots, x_3$ et $s_1 s_0$:

Dans un premier temps, on évalue s_0 pour sélectionner (en parallèle) entre x_0, x_1 et x_2, x_3.

Dans un deuxième temps, on évalue s_1 pour sélectionner parmi les deux bits choisis avant.

Le problème se géneralise pour $n > 2$ avec un circuit de profondeur $\mathcal{O}(n)$.

Pour $m > 1$: On utilise m circuits en parallèle, un par bit.

Décodeur

Données:

valeur s codé par n bits en entrée

2^n lignes de sortie $y_0, \ldots, y_{2^n-1}$

Spécification : $y_s = 1$ et $y_{s'} = 0$ pour tout $s' \neq s$.

Solution (facile) : Pour tout $0 \leq s' < 2^n$, on construit la conjonction qui évalue si $s' = s$.

Applications :

Sélectionner une case mémoire pour y écrire.

Sélectionner un périphérique parmi plusieurs pour échanger des données.

Verrous et bascules

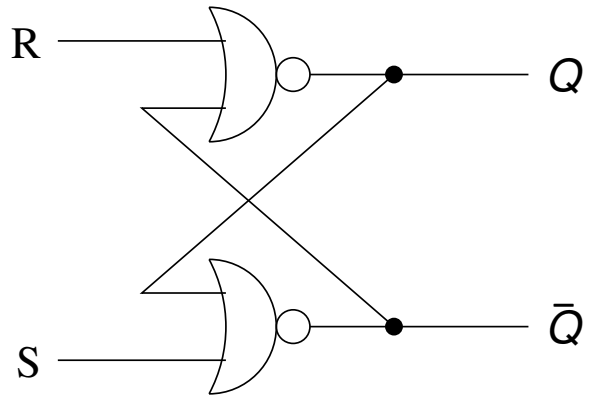
Ce sont des circuits utilisés pour stocker des bits de mémoire. La littérature distingue parfois deux types de circuits (mais sans y être strict) :

- les circuits non-temporisés sont appelés verrous (*latch* en anglais)
- les circuits temporisés sont appelés bascules (*flip-flop* en anglais)

Il y a une grand variété de verrous et bascules, selon des besoins spécifiques. Quelques éléments typiques :

- signaux de contrôle : déterminent si la valeur stockée doit changer
- signal d'horloge (bascules seulement)
- deux sorties Q et $\bar{Q}$, la valeur du bit et son complément

Verrou RS



controls R (reset) and S (set):

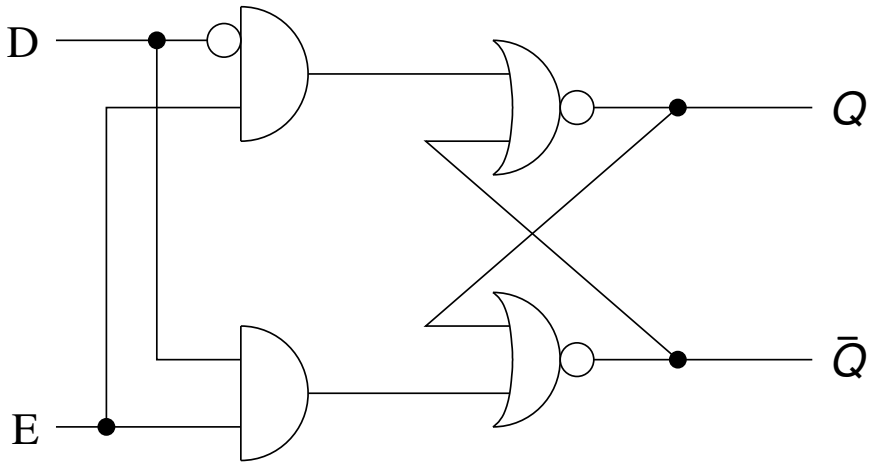
$R = 0$, $S = 1$: après peu de temps, $Q = 1$ et $\bar{Q} = 0$

$R = 1$, $S = 0$: après peu de temps, $Q = 0$ et $\bar{Q} = 1$

$R = S = 0$: les deux sorties restent inchangées

$R = S = 1$: on obtient $Q = \bar{Q} = 0$, devient instable lorsqu'on on revient à $R = S = 0$

Verrou D



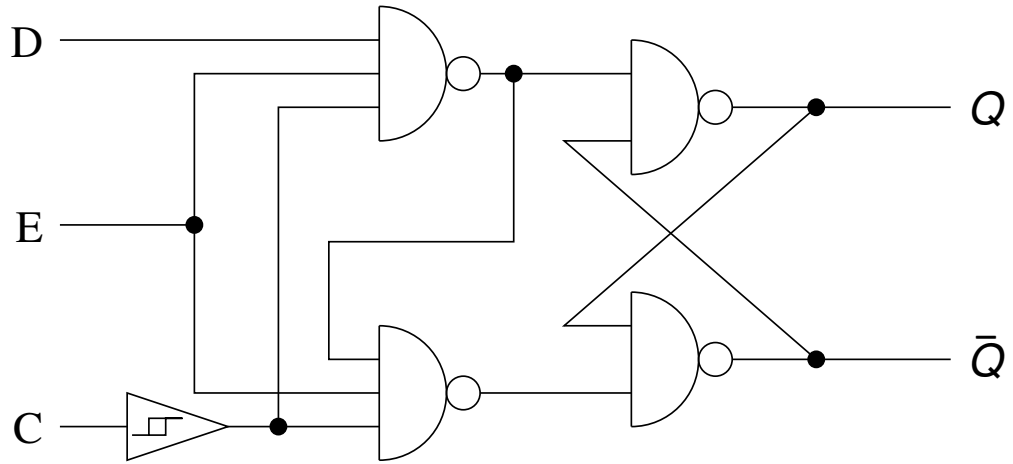
contrôles D (data) et E (enable):

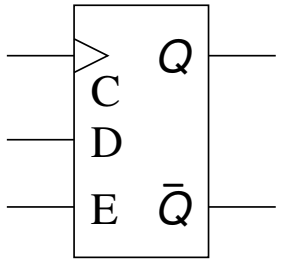
$E = 0$: stable

$E = 1$: Q devient D

Bascule D

On reprend le verrou D, on ajoutant un signal oscillant C (*clock* = horloge). Dans ce cas, D n'est pris en compte que si C est dans sa phase haute et que $E = 1$.





Dans les diagrammes, le signal d'horloge est typiquement indiqué par un triangle.

DRAM (dynamic RAM)

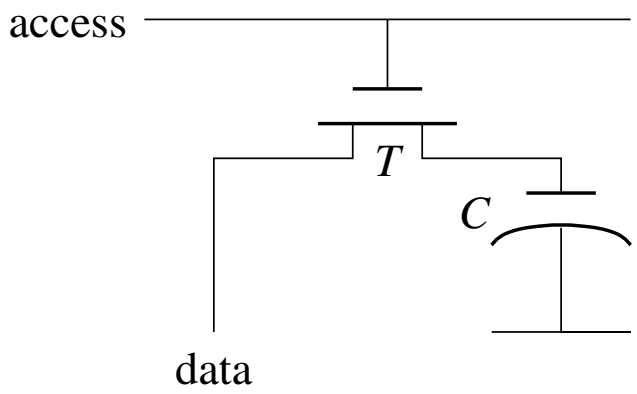
Une autre façon de stocker un bit :

utilise un seul transistor et un condensateur; 1 $\widehat{=}$ condensateur chargé

activer le signal d'accès permet de charger le condensateur ou de transmettre sa charge au signal D

opération de lecture destructrice ; il faut rétablir la charge après

plus compact d'un verrou (SRAM) mais moins rapide



Calcul cyclique d'un ordinateur

Les bascules définissent l'*état actuel* de l'ordinateur :

mémoire, registres, signaux de périphériques, . . .

Le signal fourni par l'horloge *C* est composé d'une longue phase de 0 alternant avec une courte phase de 1.

Pendant la phase 0, le processeur calcule l'état suivant :

Quel registre doit changer ? Quelle case de mémoire ?

Quelles sont les nouvelles valeurs ?

Placer les 'enable' des bascules concernées.

Lorsque l'horloge passe en phase 1, les nouvelles valeurs sont acceptés dans les bascules. L'ordinateur passe ainsi dans un nouvel état, et le cycle recommence.

Signal de horloge

L'horloge C est donc partagé parmi toutes les bascules.

La pause entre les "1" de C doit être suffisamment longue que tous les calculs d'un cycle puissent réussir.

Les "1" doivent être suffisamment courts pour bien séparer les phases d'un calcul.

Mémoire vive

Les bascules/verrous sont un exemple d'une *mémoire vive*, avec les propritétés suivantes :

mode lecture ou écriture

volatile (contenu perdu sans alimentation électrique)

utilisée pour faire des calculs, stocker des données

en anglais: RAM (random-access memory, on accède à toute adresse en temps constant)

Mémoire morte

Par contre, une *mémoire morte* est dite d'avoir les propriétés suivantes :

mode lecture seulement (pas de moyen pour l'ordinateur ou le programmeur d'en modifier)

durable (même sans alimentation électrique)

utilisée pour assurer le fonctionnement de l'ordinateur (microprogrammation, tables mathématiques pré-calculées)

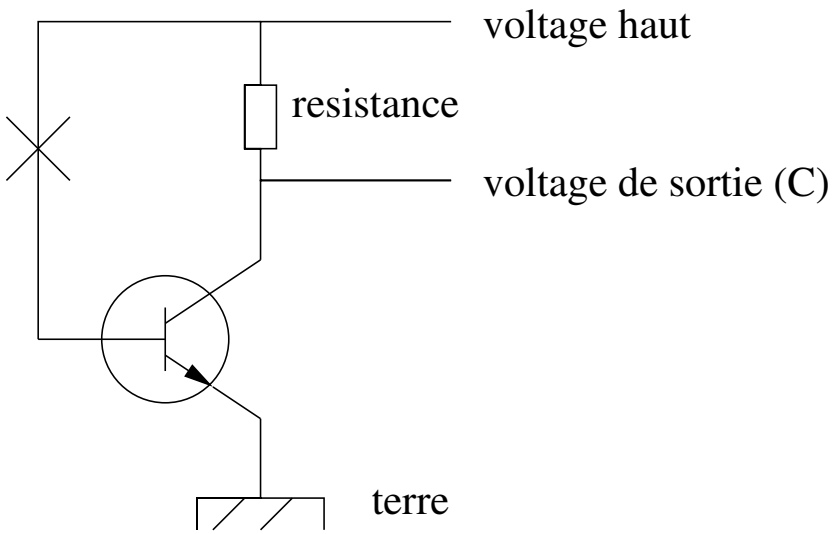
en anglais: ROM (read-only memory)

Exemple d'une mémoire morte

La façon la plus simple de produire un ROM :

prendre une série de transistors, chacun représentera un bit

physiquement interrompre le fil à la position X pour donner 1, sinon 0.



D'autres types de mémoire morte : PROM, EPROM, EEPROM

EPROM

EPROM = Erasable programmable ROM
et EEPROM = electrically erasable PROM

Transistors avec une porte supplémentaire (“floating gate”) qui accède à une couche entre la base et les autres élements du transistor.

Le “floating gate” est programmable avec des charges électriques supérieur à ceux utilisé pendant l’opération normale de l’ordinateur.

$\rightarrow$ reprogrammable par l’extérieur, mode lecteur seulement pour l’ordinateur

Architecture d'un ordinateur simple

Les éléments:

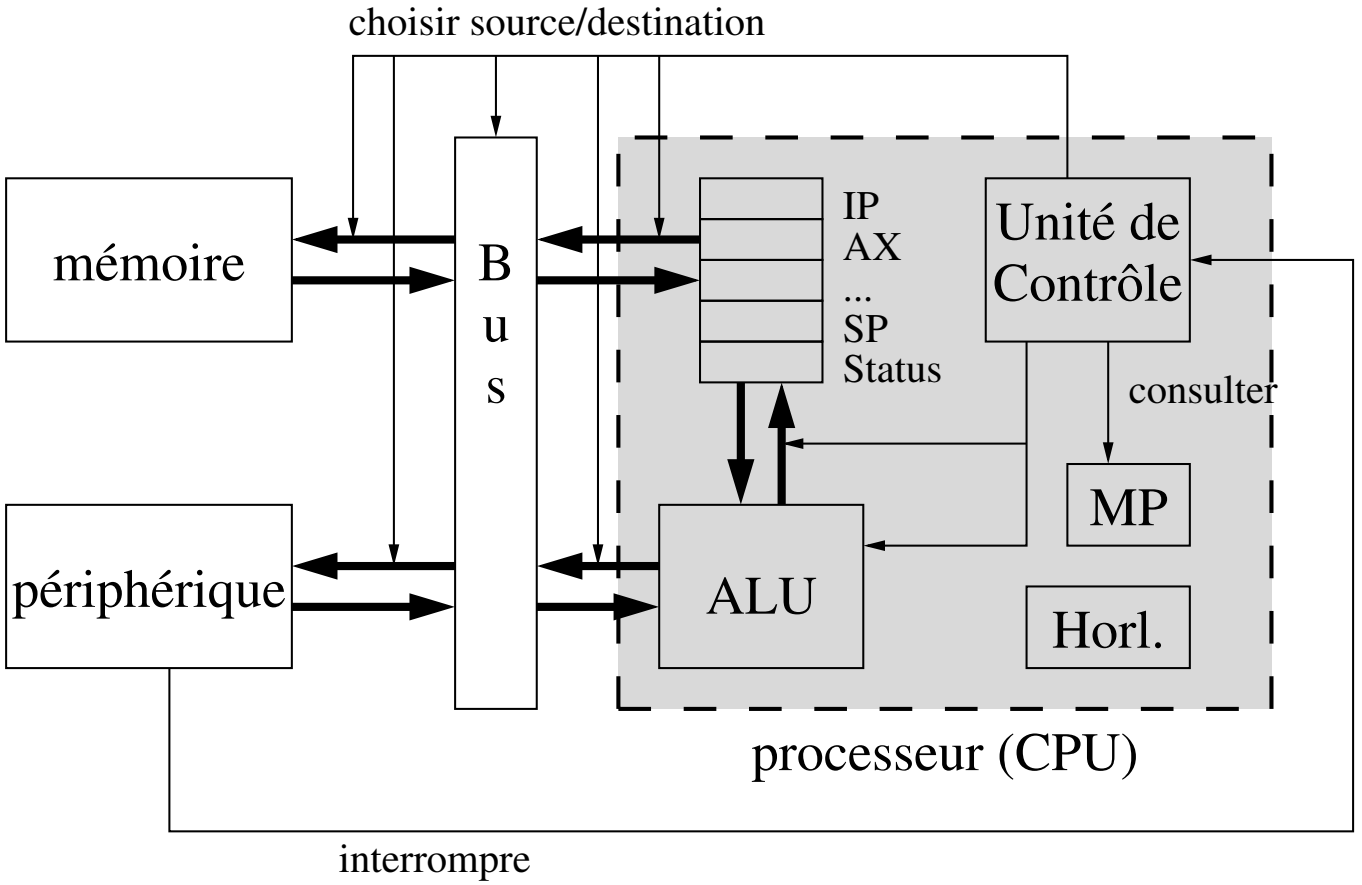
processeur (CPU), consiste d'une unité de contrôle (CU), unité arithmétique-logique (ALU), unité de microprogrammation et horloge

mémoire principale

les périphériques (fournissent des entrées/sorties pour les données)

le bus qui transfert des données entre les composants

Éléments d'un ordinateur simple



Mémoire principale

La mémoire est utilisée pour stocker les données et les instructions.

Organisée en 2^n mots de m bits (typiquement $m = 8$ – “octet”)

Échanges avec le processeur :

lecture : multiplexeur pour choisir un mot à une adresse précise, transfert au bus

écriture : obtenir un mot et une adresse du bus, on utilise un décodeur pour choisir la bonne case mémoire

Le bus

En général, un *bus* permet à plusieurs parties de communiquer par un moyen de transmission partagé. Généralement, le bus permet un seul transfert ciblé à la fois.

Dans notre cas, un tel bus connecte le processeur central, la mémoire et les périphériques: tous ces composants peuvent y lire et écrire (mais un seul à la fois).

Pour faire simple, on peut supposer que le processeur utilise des multiplexeurs et décodeurs pour sélectionner la source et la destination d'un transfert.

Le processeur central (CPU)

Le processeur contrôle toutes les opérations. Il a les composants suivants :

registres : mèmoire de court terme pour les opérations actuelles

ALU : pour manipuler des données

unité de contrôle : pour coordonner les transferts, avec l'aide de l'horloge et l'unité de microprogrammation

Registres

Un registre stocke un mot (souvent plus qu'un mot de mémoire).

Il existe plusieurs registres avec des finalités différentes :

registres géneraux (accumulateur, AX, BX, ...) : utilisés pour les calculs

compteur d'instructions : pointe vers la prochaine instruction dans la mémoire

registre de statut : pour stocker quelques bits indiquant le succés ou non d'une opération (comparaison, division par zéro, ...)

pointeur de pile (SP) : indique le sommet d'une pile qui sauvegarde des valeurs de certains registres, adresse de retour pour les sous-routines, etc

Unité de contrôle (CU)

La CU travaille dans des phases indiquées par l'horloge :

Chaque phase réalise une opération nécessaire pour exécuter une instruction:

Dans chaque phase, le CU doit organiser les transferts de données:

- donner les bons bits de sélection au bus
- transférer les donées entre ALU et registres
- réagir aux périphériques

Quelques phases typiques: obtenir une instruction, décoder une opération, exécuter une opération (en plusieurs phases)

Pour chaque phase d'une instruction, les signaux de contrôle sont stockés dans une ROM qu'on appelle unité de microprogrammation.

Interruptions

Des périphériques (clavier, réseau) n'ont pas toujours besoin de communiquer avec le processeur, mais si besoin, le processeur doit réagir rapidement.

Quand un périphérique souhaite échanger des données avec le processeur, il met son signal d'interruption à 1.

Quand le processeur est prêt pour la prochaine instruction, il vérifie s'il y a reçu un tel signal d'interruption et choisit celui avec la priorité la plus élevée.

Dans ce cas le processeur continue son travail à une certaine adresse en fonction du signal choisi. Cette adresse devrait contenir du code pour gérer la communication. Ensuite on continue l'exécution normale.