

Architecture et Système

Stefan Schwoon

Cours L3, 2020/2021, ENS Cachan

Multiplexeur

Supposons qu'on possède des mots (= vecteurs) x_0, \dots, x_{k-1} en entrée, pour $k = 2^n$, où chaque mot est composé de m bits.

On souhaite sélectionner l'une de ces mots par son indice, p.ex. étant donné s (codé par n bits), on souhaite sélectionner 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

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

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

Supposons $m = 1, n = 2$, du coup on possède x_0, \dots, 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 généralise pour $n > 2$ avec un circuit de profondeur $\mathcal{O}(n)$.

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

Décodeur

On possède une valeur s codée par n bits et 2^n sorties y_0, \dots, y_{2^n-1} .

Sortie : $y_s = 1$ 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 grande variété de verrous et bascules, selon des besoins spécifiques.

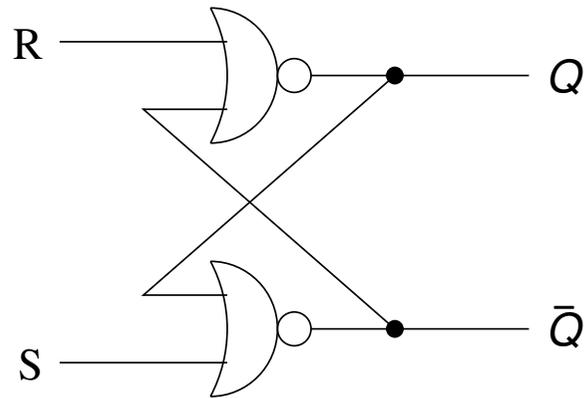
Quelques éléments typiques :

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):

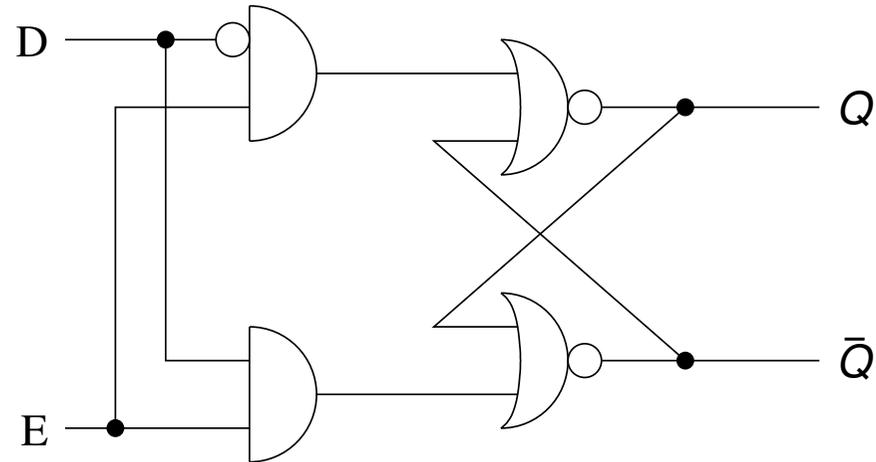
$R = 0, S = 1$: après peu de temps, $Q = 1$ et $\bar{Q} = 0$

$R = 1, S = 0$: 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



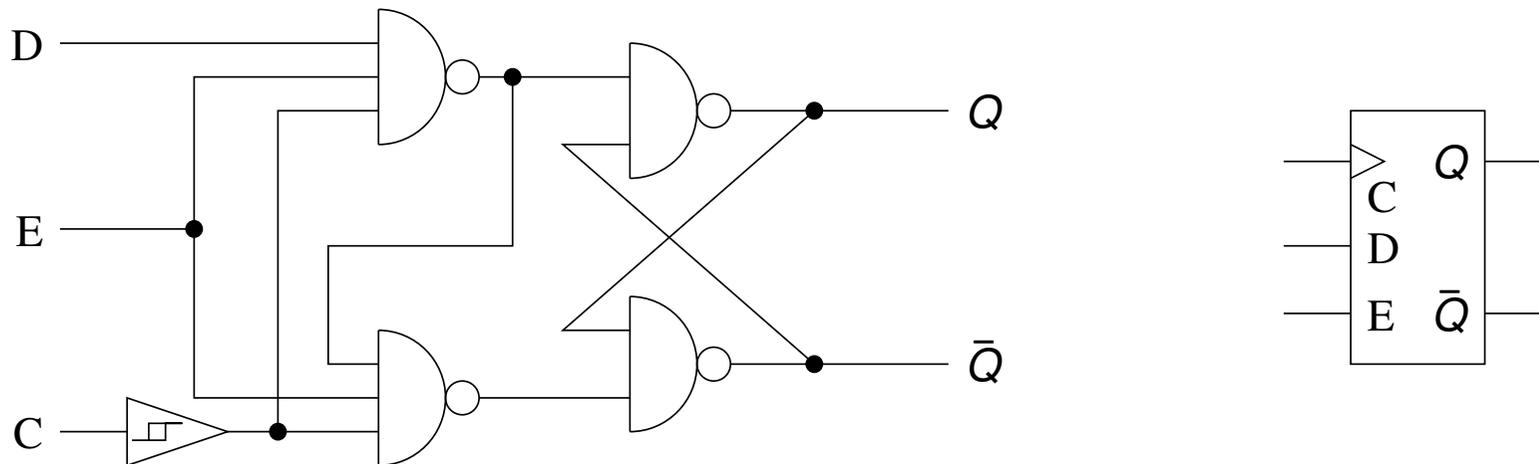
contrôles D (data) et E (enable):

$E = 0$: 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.

Quand le signal C est partagé parmi tous les bascules du système, ceci permet de synchroniser les calculs dans le système entier.

Le délai entre les “1” de C doit être suffisamment long pour que tous les calculs en parallèle puissent réussir.

En plus, 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 propriétés 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)

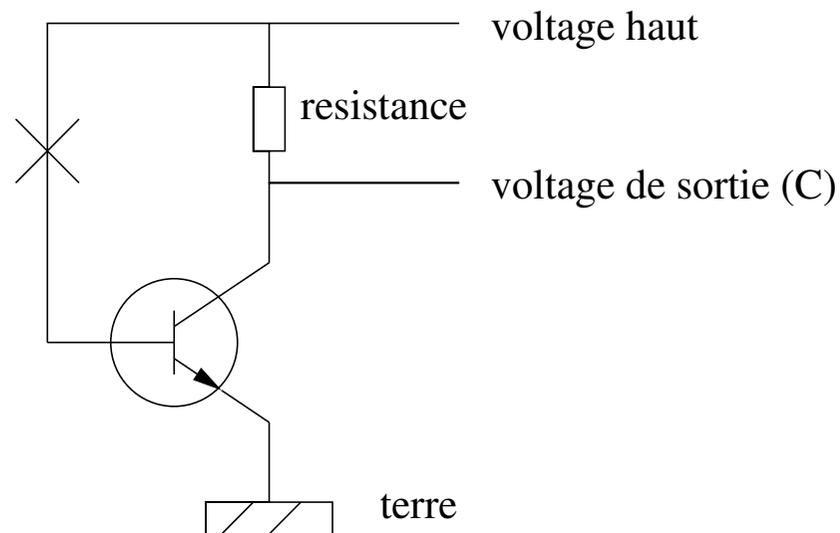
en anglais: ROM (read-only memory)

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

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 transfère des données entre les composants

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 : mémoire 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 généraux (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 succès 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 données 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**.