PREMIERS PAS VERS LA SELECTION: L'IDENTIFICATION

Depuis nos Fiches n° 6 et 7, nous savons pour ainsi dire tout ce qu'il faut pour dessiner le cœur d'un petit ordinateur. Tout, sauf les procédés qui vont permettre, dans l'espace adressable d'un processeur, de désigner précisément les circuits de mémoire ou d'entrées/sorties.
En ce sens, la première étape, c'est de leur donner un nom et de les identifier.

Les boîtes de sélection

En dehors de leur bus de données, les lignes qui sortent d'un microprocesseur sont, pour l'essentiel, de deux sortes :

- lignes d'**adresses,** qui véhiculent en binaire le numéro d'un « mot » de mémoire ou d'entrées/sorties,
- lignes auxiliaires, qui donnent le **sens** du transfert voulu (lecture ou écriture) et l'**instant** de ce transfert de donnée.

Dès qu'un système est un tant soit peu complexe **(fig. 1),** les signaux d'adresses et, en règle générale, quelques-uns des signaux auxiliaires de commande, entrent dans une série de boîtes en cascade qui finissent par émettre pour chaque composant « élémentaire » **un** signal de sélection.

Ce signal va, par exemple, rendre transparente telle barrière trois-états, ou encore activer tel bloc de mémoire morte...

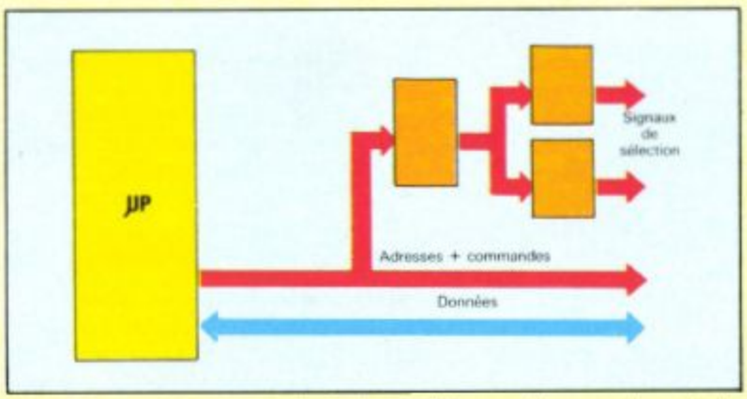


Fig. 1. – Dans un ordinateur, une partie des signaux d'adresses et de commandes sont traités par des circuits de « décodage » en cascade, élaborant les signaux de sélection détaillés pour les sous-ensembles de la machine.

ADRESSES		MODULE/FONCTION
00000	OFFFF	Mémoire vive sur carte mère
10000	2FFFF	Carte extension 192 K
40000	A3FFF	Carte extension 384 K
A4000	F3FFF	Extensions et cartes d'E/S
F4000	FFFFF	Mémoire morte sur carte mère

Fig. 2. – Un parti pris très important pour la « personnalité » d'un micro : sa cartographie-mémoire. Ici, une version simplifiée de celle du PC d'IBM... et des vrais compatibles.

Une carte d'identités

Le pluriel de ce sous-titre n'est pas fortuit, mais voulu...

Quand un ingénieur dessine un micro, il règle bien souvent en une poignée de minutes, sur un coin de table et sur une feuille de papier volante, une question d'une importance « stratégique » : **où** seront localisés, dans les adresses connues du programmeur, les différents composants présents : mémoires et entrées/sorties du modèle de base, et à venir : les fameuses « extensions ».

Choix incroyablement critique, dont dépendra jusqu'à la survie de son produit.

Cette question à elle seule mériterait un long, trop long exposé.

De la réflexion initiale du concepteur, il résulte un document connu sous l'appellation de **cartographie-mémoire** *(memory mapping)* ; si le processeur a ses entrées/sorties hors du champ des adresses-mémoire, il y a aussi une **carte des E/S** *(I/O mapping).*

Pour le montage électronique, cette carte dit, au moins en « première approximation », **où** doit être situé **tel composant.**

Dans le PC

A la **figure 2,** nous donnons une version simplifiée de la carte des mémoires du fameux IBM PC. Regardons, à la loupe, le cas du premier module d'extension-mémoire de 192 K-octets.

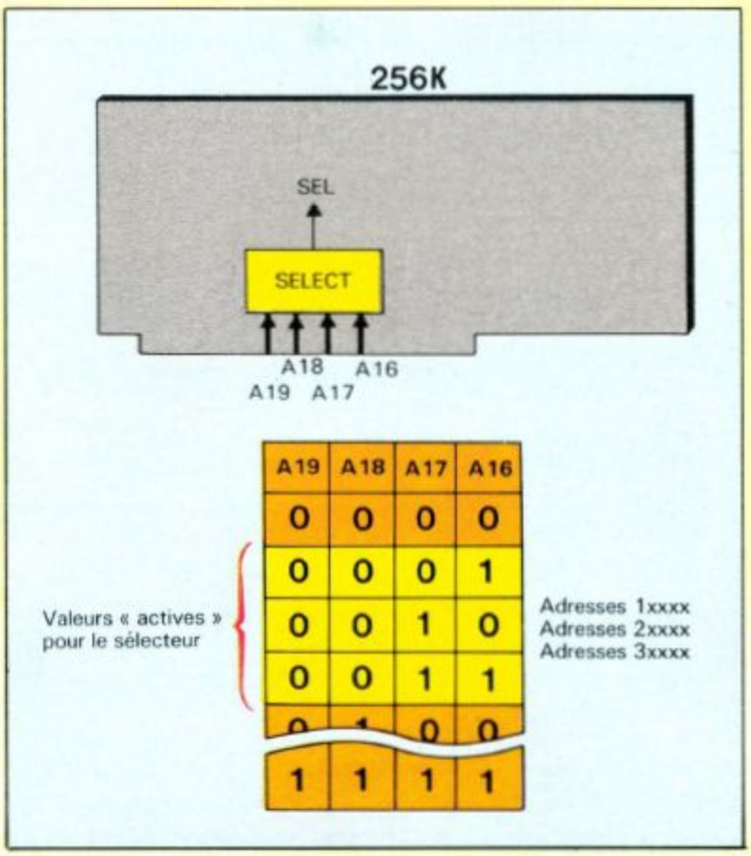


Fig. 3. – Une carte de mémoire occupant les 192 Ko considérés devra s'identifier à partir des trois combinaisons « autorisées » des bits 19 à 16 sur le bus d'adresses (valeurs hexadécimales de 1 à 3).

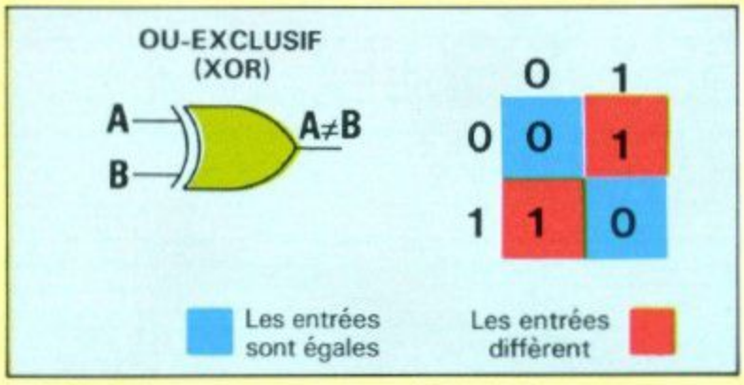


Fig. 4. – Le circuit comparateur par excellence, le OU-EXCLUSIF. C'est la fonction « différent de... » utilisable en inverse pour « égal à... ».

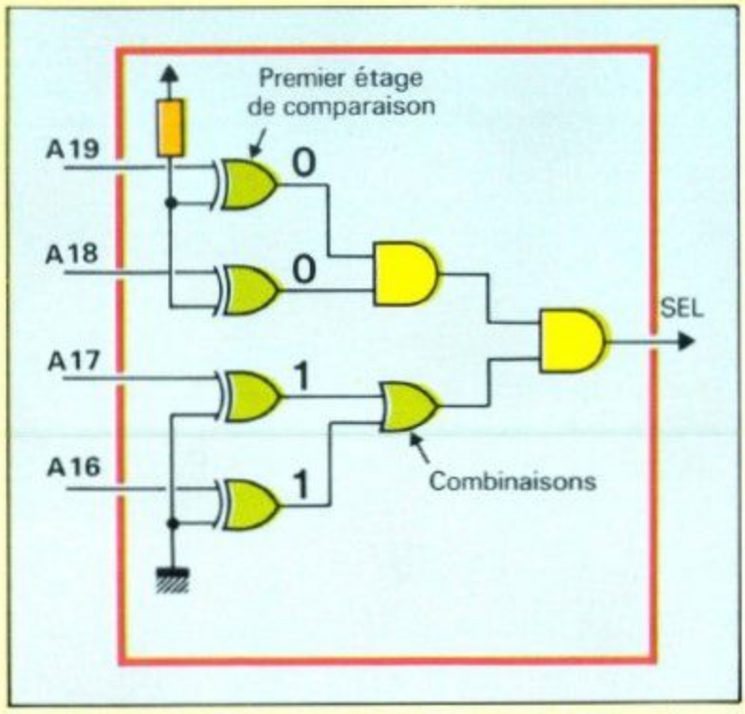


Fig. 5. – Exemple (non optimisé) d'un montage réalisant la fonction de la figure 3. Ce dernier correspond rigoureusement à l'énoncé du texte.

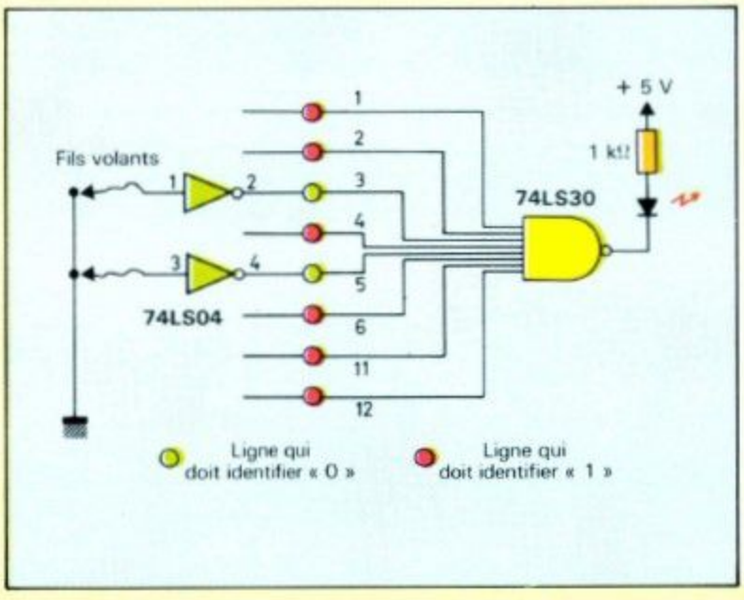


Fig. 6. – Montage identifiant le motif binaire 11010111. Pour simplifier le câblage, les entrées « à 1 » sont laissées « en l'air » ; ce qui en soi n'est pas recommandé : il faudrait des résistances de rappel dans un système réel.

Il n'existe pas (encore) de puce 192 K-octets, il faudra donc construire ce module avec des rangées de composants plus petits, qui occuperont la surface d'une carte d'extension dont la **figure 3** donne une vision très grossière.

Ces composants devront *in fine* être sélectionnés individuellement. Mais il y a un point de départ, un préalable à toute sélection particulière : il faut élaborer un signal que l'on pourrait définir par cette périphrase : « l'adresse-mémoire courante se situe dans ce module ». Nommons SEL ce projet de signal.

Comparer, combiner

Avec des mots de tous les jours, le cahier des charges de la logique de sélection s'exprime ainsi : SEL sera actif :

– **si** les lignes d'adresses A_{19} **et** A_{18} **sont à « 0 »,**
– **et si** l'une **ou** l'autre des lignes A_{17} et A_{16} **est à « 1 ».**

Sur les étagères des électroniciens, il y a une porte qui se nomme OU-EXCLUSIF *(exclusive-or)*, et qui fait très bien l'affaire quand on doit réaliser des fonctions « différent de... » ; soit, à un inverseur près, « égal à... » **(fig. 4).**

La **figure 5** est un exemple de réalisation du sélecteur voulu ; pas forcément la meilleure, mais avec une « structure régulière » de portée générale.

Un premier étage de portes OUX élabore des signaux du genre « égal à 0 » (c'est-à-dire, « différent de 1 ») et vice-versa. Un second étage réalise littéralement les combinaisons ET et OU de notre énoncé.

Le résultat : un signal SEL qui est haut pour le **champ d'adresses** considéré.

Un grand classique

Parmi les multiples techniques de **reconnaissance** d'une combinaison de bits donnée *(pattern matching)*, l'une des plus usitées utilise un composant ET à entrées multiples, tel le 74LS30 qui en a huit ; c'est un NAND : la seule différence avec un AND est l'inversion de polarité de la sortie qui sera :

– à « 0 » si **toutes** les entrées sont à « 1 »,
– à « 1 » si l'**une au moins** est à « 0 ».

L'emploi d'un tel composant pour reconnaître telle combinaison de zéros et de uns est évident.

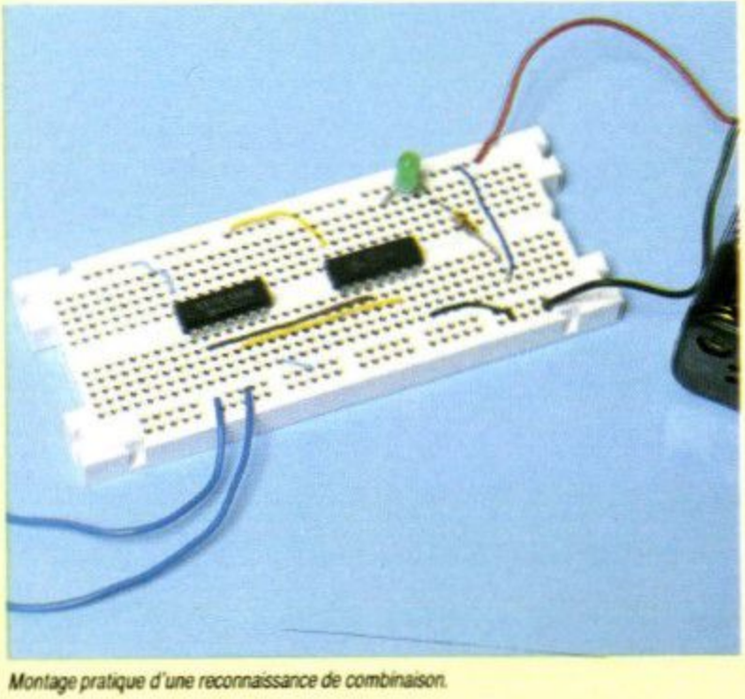
Montons, par exemple, un circuit qui « reconnaît » la combinaison 1 1 0 1 0 1 1 1 **(fig. 6).**

Pour chacune des lignes où « l'on veut 1 », rien de plus facile ; on relie cette ligne directement sur une entrée, où elle apportera sa contribution... positive au ET.

Pour les lignes où « l'on veut 0 », on intercale un inverseur ; et l'on est ramené au constat précédent.

Du particulier au général

Cette recette est appréciée du concepteur parce qu'elle a un caractère systématique : on dessine un grand ET, et l'on insère des inverseurs



Montage pratique d'une reconnaissance de combinaison.

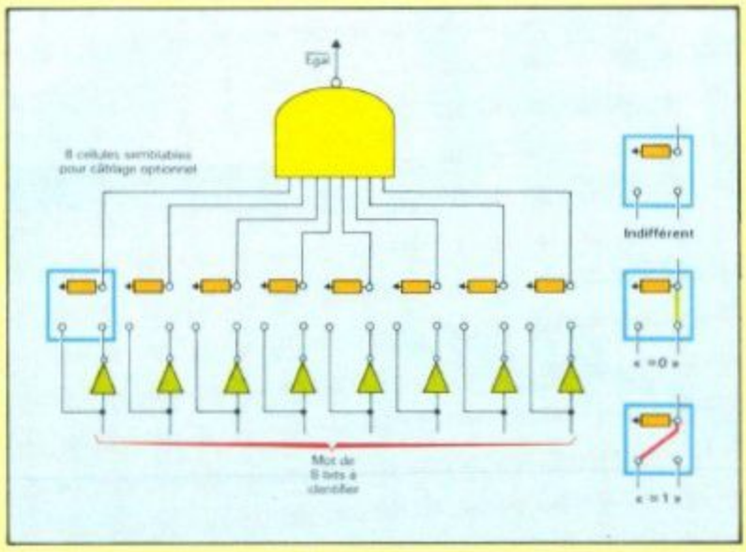


Fig. 7. – Montage universel d'identification sur 8 bits. On peut grâce à une résistance de rappel « ignorer » certains bits ; en anglais don't care.

sur les lignes que l'on veut identifier comme des zéros.

Il y a un cas particulier, très prisé parce qu'il n'y a pas d'inverseur du tout : la reconnaissance de 1 1 1 1 1 1 1 1 (le fameux FF hexadécimal). C'est en général l'adresse, ou un morceau d'adresse, que s'octroie le « premier arrivé » dans les montages d'extension. Les successeurs n'y couperont pas d'au moins un boîtier supplémentaire du genre 74LS04...

On peut aussi imaginer de traiter ainsi le cas le plus général, c'est-à-dire réaliser un montage qui permettra, grâce à des cavaliers d'options (*straps* ou commutateurs), de décider au dernier moment à quelle combinaison le comparateur sera sensible.

On aboutit à un schéma comme à la **figure 7**, qui n'est pas des plus performants : il faut trois « points » par option (trois contacts tels que des picots à enrouler), et trois boîtiers pour avoir jusqu'à huit inverseurs en service.

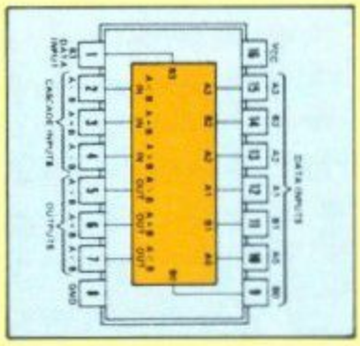


Fig. 8. – Brochage du classique comparateur 74LS85. Ce composant peut être monté en cascade pour comparer N fois 4 bits ; en outre, il considère leur valeur arithmétique, d'où les sorties « plus grand » et « plus petit », en plus de la sortie « égale ».

Encore le OU exclusif

En dehors des circuits comparateurs conçus comme tels **(fig. 8)**, on peut penser à des montages qui emploient astucieusement un composant un peu méconnu : le quadruple OUX à sortie en collecteur ouvert, le 74LS136.

Nous vous proposons de monter, d'expérimenter et d'analyser le montage de la **figure 9**, qui est un comparateur universel sur quatre bits ; extensible naturellement par la vertu du « ET câblé » que permettent les collecteurs ouverts.

Sans déranger Mr. Boole, on peut formuler ainsi l'astuce logique : dire que des bits sont tous égaux à la valeur voulue, cela revient à dire qu'ils sont **tous différents** des valeurs opposées...

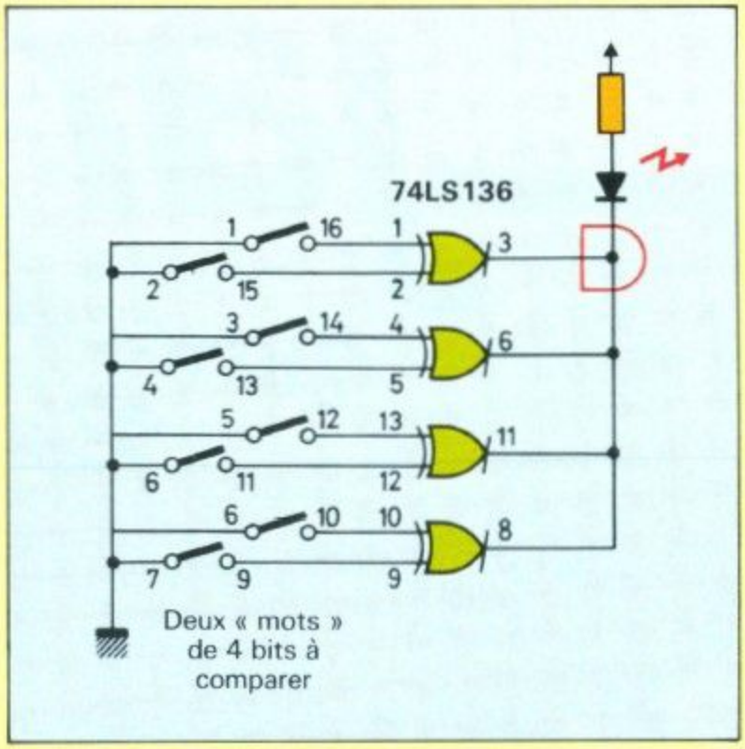
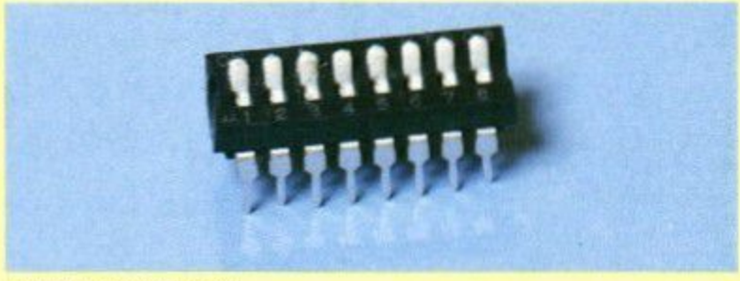
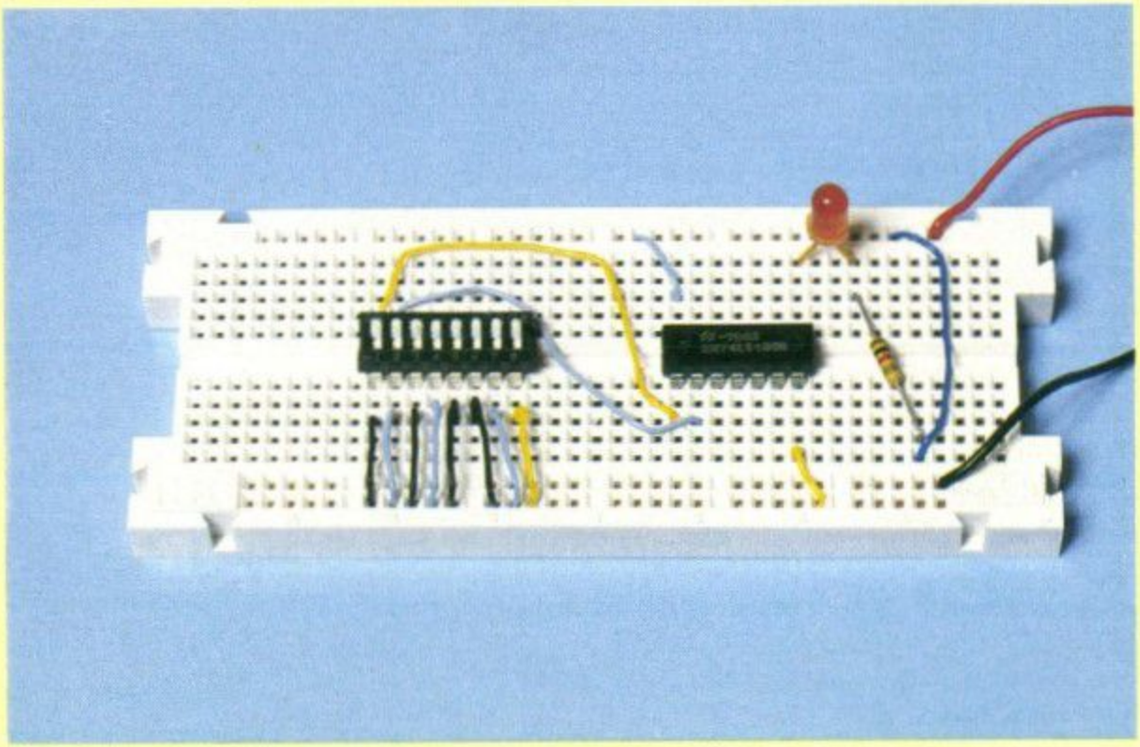


Fig. 9. – Utilisation d'un quadruple OU-EXCLUSIF à sorties en collecteur ouvert pour comparer 4 bits à 4 bits. Noter le ET câblé ; le témoin ne s'éteint que si les paires de commutateurs montés en DIP sont de sens opposés. Ici encore, on fait l'impasse sur les résistances de rappel.



Bloc DIP à huit commutateurs.



Un comparateur universel sur quatre bits.

LES DECODEURS: SUITE ET FIN DES MOYENS DE SELECTION

Alternative aux procédés d'identification déjà vus, les circuits décodeurs sont aussi leurs compléments dans bien des montages micro-informatiques.
Sous leur apparente simplicité, ce sont des outils puissants... et dangereux si l'on n'y prend pas garde !

Décoder, c'est montrer du doigt

La fonction d'un décodeur, c'est en quelque sorte de montrer du doigt un objet dont on a le numéro... codé.

Plus précisément, la plupart des composants de cette catégorie obéissent au schéma de principe de la **figure 10,** où la «boîte noire» reçoit un petit nombre de lignes logiques, dont les valeurs sont interprétées comme un **nombre** représenté en « binaire naturel ».

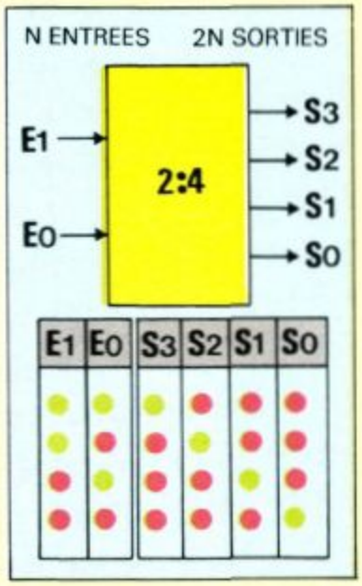


Fig. 10. – Exemple d'un décodeur « 2 vers 4 » ; une seule des lignes de sortie est activée à un moment donné, celle dont le « numéro » en code binaire apparaît sur les entrées.

En sortie, **une et une seule** des lignes est active pour chaque code ; une autre façon de présenter les choses, c'est de dire qu'on a en entrée un numéro, et qu'en sortie seule la ligne portant ce numéro sera active.

Reste à donner un sens au mot « active » ; il y a deux possibilités : un « 1 » alors que les autres sorties sont à « 0 », ou l'inverse.

Un zéro parmi les uns

C'est l'inverse qui est standardisé dans les faits. Cette coutume a son origine dans l'électronique TTL, où le « 1 » est plutôt un état « de repos » impliquant peu d'énergie.

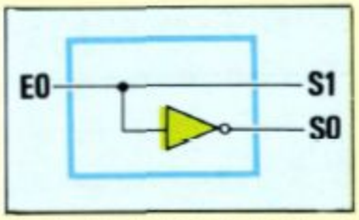


Fig. 11. – Le plus trivial des décodeurs « 1 vers 2 ». Il suffit d'un inverseur. C'est le premier étage de tout décodeur plus complexe : voir la suite du texte.

Le plus simple des décodeurs est réalisé avec un inverseur **(fig. 11).** L'entrée est une ligne unique ; les deux sorties seront numérotées 0 et 1 ; la sortie numéro zéro est reliée à l'entrée, ce qui donne bien... 0 pour 0, l'autre ligne de sortie étant à 1 via l'inverseur.

On ne va pas vous faire l'injure de plus de détails sur ce décodeur vraiment trivial !

Le principe des décodeurs plus complexes apparaîtra sur le montage de la **figure 12,** avec deux lignes en entrée, donc quatre en sortie. Montage que vous êtes invité à réaliser, grâce à nos vieux amis les NANDs (74LS00) et inverseurs (74LS04).

Un schéma régulier

C'est à dessein que le schéma est représenté ainsi, avec ses lignes croisées.

Chaque entrée est utilisée telle quelle en « ligne directe », mais aussi via un inverseur ; ce qui permet de disposer selon besoin de l'entrée ou bien de son complément.

Les combinaisons s'obtiennent via des NANDs attaqués chacun par une combinaison différente. Ainsi, la porte qui commande la sortie S_3 est-elle reliée aux lignes directes E_0 et E_1, de telle sorte que ladite sortie viendra au « 0 » pour la combinaison « 1 1 » (interprétée comme 3 en code binaire).

La sortie S_1 est commandée par un NAND sur lequel entrent E_0 et E_1 et viendra au niveau bas pour les entrées « 0 1 », etc.

Le lecteur attentif voit déjà que toute la « logique » du montage est symbolisée par les gros points qui connectent telle entrée d'un NAND de **décision** (par combinaison) avec telle ligne de **donnée** ; directe ou complémentaire.

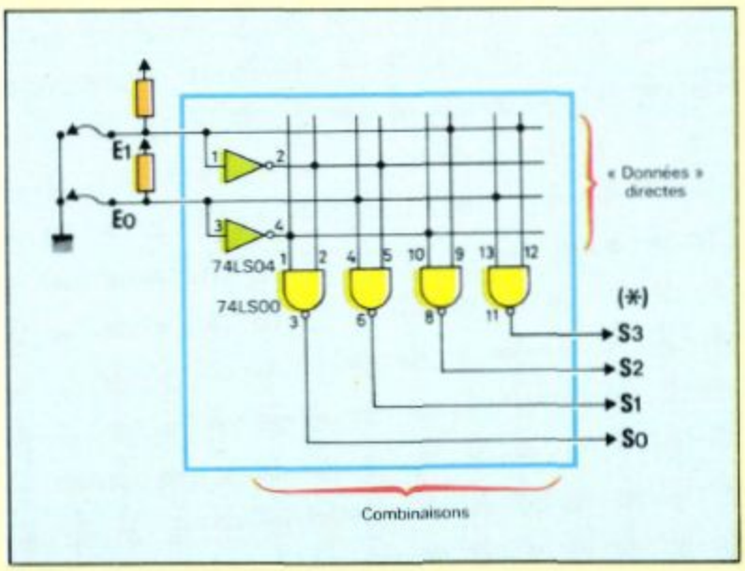


Fig. 12. – Le décodeur 2:4 réalisé avec des composants simples. On note le premier étage qui donne les entrées en versions directes et complémentaires ; le second les combine en connectant chaque NAND aux deux lignes ad hoc. Le NAND de S_1 va être sollicité spécialement dans les expériences suivantes (fig. 13 et 14).

Les soi-disant « aléas »...

La preuve du bon fonctionnement s'obtient aisément en forçant sur les entrées les quatre combinaisons possibles. Le pèse-signaux appliqué aux quatre sorties exhibe effectivement un seul « 0 » parmi des « 1 ».

Ce fonctionnement est-il réellement garanti ?

Danger ! Danger !

Cela vaut vraiment la peine d'introduire un composant de plus dans le montage ; pour prouver, expérimentalement, que la réalité n'est pas si évidente.

Au schéma de la figure 12, ici résumé par un bloc, vient s'ajouter un élément que nous connaissons depuis les Fiches n° 7, à savoir un compteur construit par bouclage à partir d'une bascule sensible aux transitions de son entrée d'horloge. En pratique, la moitié d'un 74LS74 **(fig. 13).**

Une simple LED-témoin visualisera le test.

En théorie, cette LED ne devrait **jamais** changer d'état. En effet, nous avons relié ensemble les deux entrées du décodeur ; donc, les combinaisons d'entrée ne peuvent être que « 00 » ou « 11 », d'accord ?

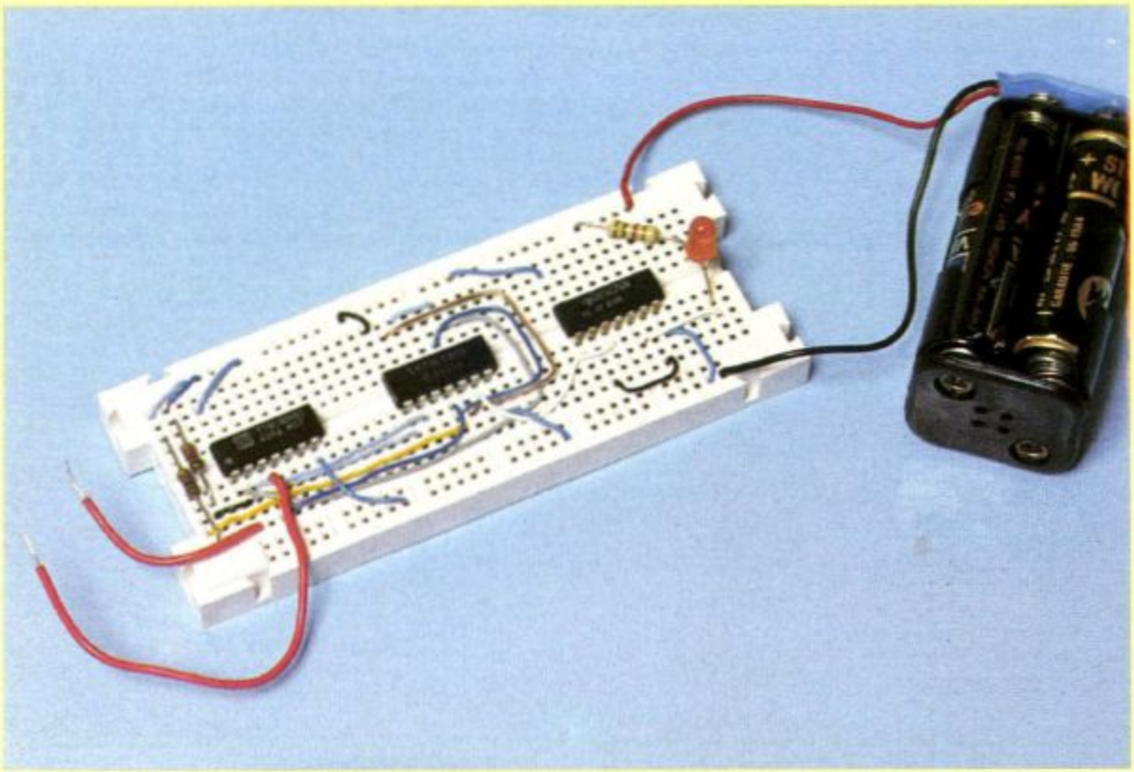
... sont souvent des erreurs de dessin

Si l'on se fie à la table de vérité du décodeur, on en conclut, un peu vite, que les sorties S_1 et S_2 resteront invariables à « 1 », seules les sorties S_0 ou S_3 étant activées.

Or, bien que « ça ne marche pas à tous les coups », **il arrive** que la bascule D change d'état (la LED s'allume et s'éteint) lorsque l'on joue avec le contact entrée/masse !

C'est clair : il y a des signaux du type impulsion négative qui sortent par S_1 (ou S_2), puisqu'il faut un flanc montant pour faire basculer le compteur.

D'où viennent ces signaux, parasites diront certains ?



Mise en pratique des figures 12 et 13 : un décodeur 2:4 avec 74LS00 et 74LS04 et une bascule 74LS74.

A partir de quoi, comme le disait un des maîtres de l'auteur, le parasite est amplifié en un signal « légitime et majeur » qui va polluer toute la logique ultérieure...

Un palliatif : la capacité-miracle

Certains éliminent ce genre d'impulsion parasite grâce à l'artifice de la **figure 15.**

Une petite capacité (on essaiera avec une valeur de l'ordre de 10 nF), est montée entre la ligne concernée et la masse ; elle emmagasine assez de charge pour que la très brève commutation de la sortie S_1 soit « gommée » : la LED est bien figée dorénavant.

Ceci ne doit pas être considéré comme une panacée, mais plutôt comme un remède de premier secours pour ce qui est, non un regrettable incident, mais la conséquence d'une conception pas assez rigoureuse !

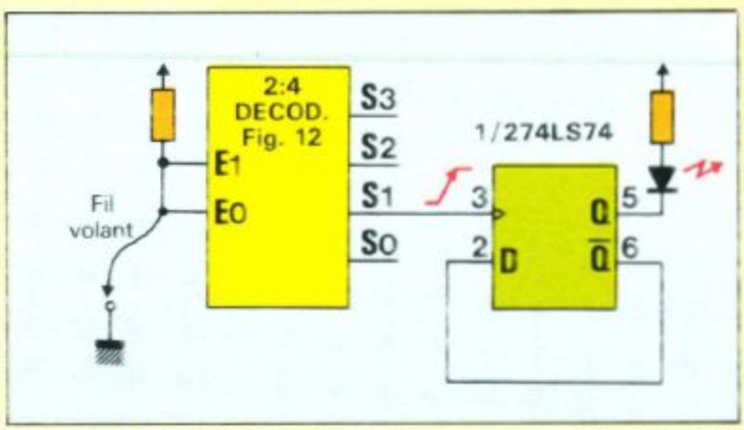


Fig. 13. – Les deux entrées du décodeur précédent sont reliées en une seule. Une bascule 74LS74 est montée en détecteur de transitions sur S_1 : théoriquement exclus, des « parasites » y apparaissent pourtant, qui font commuter la bascule, éteignant/allumant la LED.

Leur explication est bien simple (**fig. 14**). Les signaux logiques ne changent d'état « instantanément » que dans l'imagination de ceux qui croient que l'algèbre de Boole **est** la réalité.

N'importe quoi et n'importe quoi !

Avec une échelle de temps assez dilatée, regardons les signaux qui entrent dans la porte NAND correspondant à S_1.

Les deux lignes E_0 et E_1 étant reliées par ailleurs, il arrive sur cette porte un signal en train de changer d'état (E) et sa version complémentaire, via un inverseur (fig. 12).

Outre un léger retard, de quelques nanosecondes, qui va encore compliquer les choses, on voit bien que les deux signaux transitent **en même temps** par des valeurs qui ne sont « garanties » ni comme zéro logique, ni comme un.

La probabilité n'est élevée que sur une entrée, la NAND décide que c'est « déjà 1 », et sur l'autre, « pas encore zéro » ; il s'agit bien de probabilités, car dans ces phénomènes transitoires fort complexes, tout joue : les impédances des étages récepteurs, leurs capacités parasites, les minuscules oscillations de rebond des signaux...

Il s'ensuit que l'on a de bonnes chances d'observer (pas toujours !) une brève impulsion à la sortie du NAND en question ; et **il se peut** (encore des probabilités) qu'elle suffise à faire réagir un élément du genre bascule.

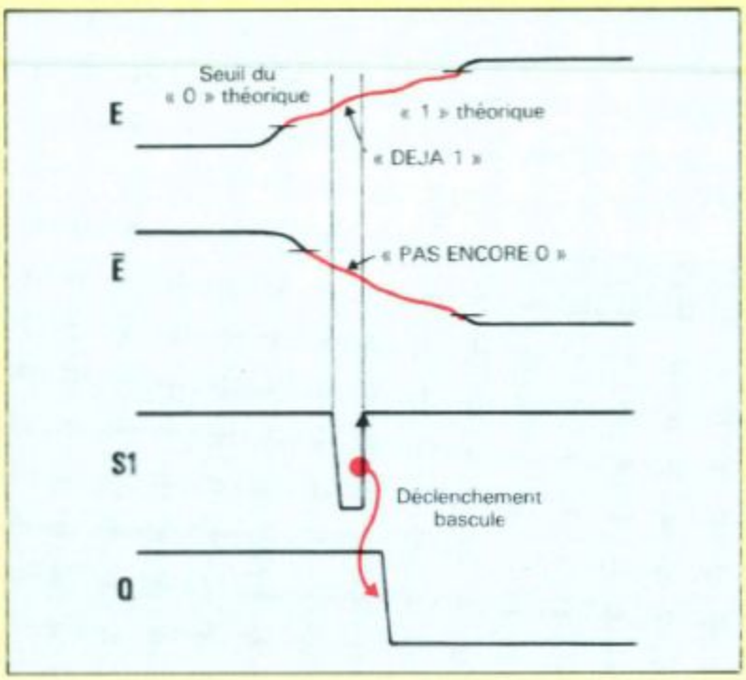


Fig. 14. – Explication des soi-disant parasites : les niveaux logiques sont indéterminés pendant les transitions.

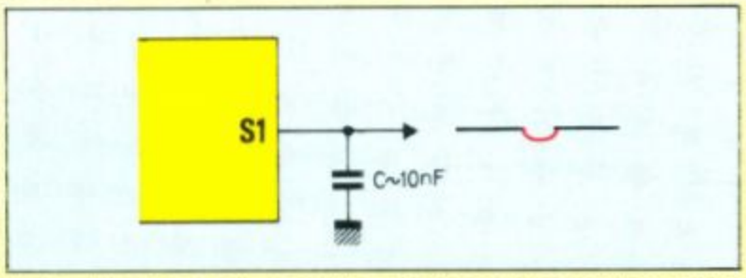


Fig. 15. – Une petite capacité, en parallèle sur la sortie parasitée, amortit la commutation indésirable. Ce n'est qu'un palliatif ; il a, entre autres inconvénients, celui de rendre « mous » les signaux normaux...

DES DECODEURS SAINS OU LE BON USAGE DU TEMPS

Retenez bien ces trois références : 138, 139 et 154 ; il y a neuf chances sur dix pour que votre micro en contienne (au moins) une ! Ce sont les « bonnes à tout faire » du plan d'adressage, des circuits disponibles en toutes technologies (LS, CMOS, HCMOS...).

L'importance du « quand »

Avec les bascules des Fiches n° 7, nous avons déjà perçu l'importance du « bon moment » lorsqu'il s'agit de commander une opération de recopie de signaux d'information.

Cette préoccupation a deux variantes, qui sont rappelées à la **figure 16.**

Lorsque la recopie est déterminée par l'état d'un signal de commande (état bas à la **fig. 16A**), on s'efforce de maintenir les lignes de « donnée » **stables** pendant la période active du signal de commande, plus une petite marge.

Quand elle est commandée par une transition **(fig. 16B),** l'exigence est de moindre durée, mais elle existe : la donnée doit être stable un peu avant et un peu après le flanc actif.

Le cycle adresse/donnée

Les microprocesseurs sont conçus par des gens qui choisissent, une fois pour toutes, que leur puce se conforme à l'un ou l'autre de ces deux schémas. Qu'il s'agisse de lire/écrire en mémoire, ou d'effectuer une entrée/sortie (pour certains, cela revient au même).

Chaque **cycle** élémentaire se déroule en deux temps : **sélection** d'un composant-**source** (s'il s'agit d'une lecture) ou -**destination** (écriture), puis échange d'un mot de donnée.

La sélection s'effectue grâce à une **adresse.** L'échange est caractérisé par deux sortes de signaux :

- ceux qui, en complément de l'adresse, définissent précisément l'opération, accès-mémoire, entrée...
- celui qui donne l'instant précis de cet échange : le « top chrono » en quelque sorte.

La **figure 17** reproduit pour illustration le cycle d'une entrée de donnée sur un Z80, c'est-à-dire l'opération déclenchée par l'exécution d'une instruction IN de ce processeur.

Un décodage d'opération

Le décodage suggéré à la **figure 17** est plus « fort » que ceux de notre Fiche 8B. Il prend en compte non seulement les lignes d'adresses, qui vont donner le numéro d'une source de données (dont la forme la plus simple est une barrière trois-états), mais aussi les signaux :

– $\overline{IORQ}$, qui valide cette adresse comme un numéro dans l'**espace** des entrées/sorties,

– $\overline{RD}$, dont le flanc arrière délimite la recopie effective de la donnée dans le microprocesseur.

Pour une adresse donnée, on peut donc élaborer un **signal d'opération** sans ambiguïté grâce à :

– un décodeur, complété par une entrée de validation qui combine les signaux d'activation caractéristiques, tels que $\overline{IORQ}$ et $\overline{RD}$ dans l'exemple.

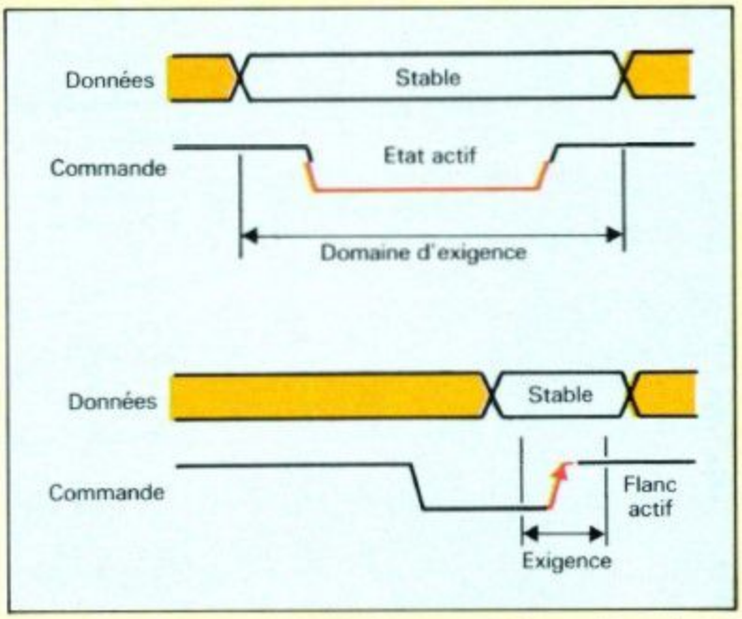


Fig. 16A. – Signalisations du premier type : les données sont spécifiées « stables » pendant tout le temps où le signal de commande est actif ; ici (pour l'exemple) au niveau bas.
Fig. 16B. – Signalisations du second type : le signal de commande présente un flanc actif, et les données sont spécifiées « stables » peu avant et peu après ce flanc.

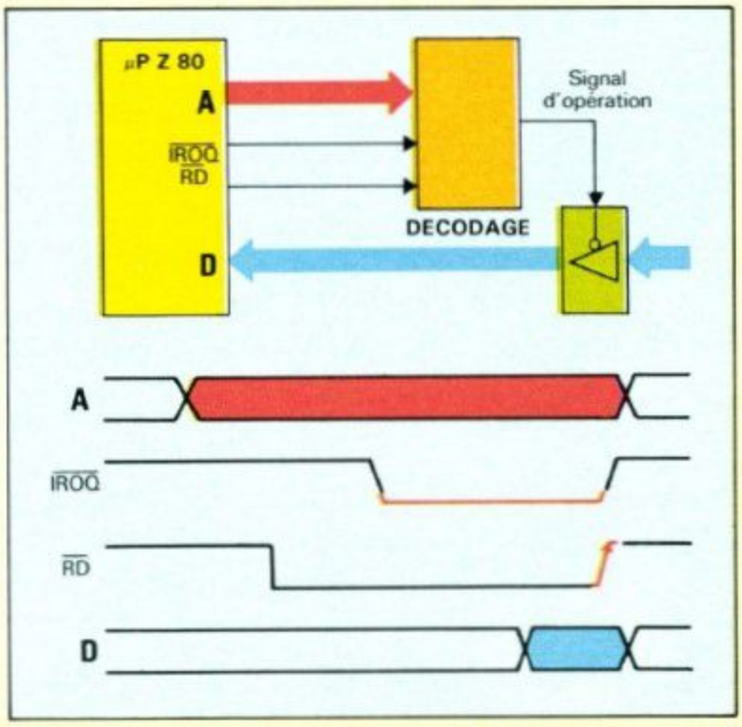


Fig. 17. – Cycle de lecture d'un Z80 (correspond à l'instruction IN). Le signal $\overline{IORQ}$ indique que l'adresse est celle d'une entrée ou d'une sortie ; le signal $\overline{RD}$ indique l'instant où le Z80 a fini d'échantillonner la donnée sur son bus D.

Un composant exemplaire : le décodeur 138

Une telle combinaison de signaux peut s'effectuer à l'aide d'un décodeur intégré tel que le 74LS138, dont le schéma interne est donné à la **figure 18** avec son brochage.

L'œil exercé y reconnaît le dessin régulier d'un circuit de décodage, à ceci près que les NANDs de sortie ont une entrée auxiliaire, connectée à un ET qui combine trois points de validation : G1, qui est active au niveau haut, $\overline{G2A}$ ET $\overline{G2B}$ (G comme *gate*) qui sont actives au niveau bas.

Si nous reprenons l'exemple du Z80 exécutant une instruction IN, et si nous précisons que l'adresse voulue est 3 dans une gamme de numéros limitée de 0 à 7, ce décodeur suffira à lui seul pour donner **le** signal d'opération ouvrant, disons, une barrière trois-états **(fig. 19).**

Une bonne façon d'énoncer les fonctions, c'est de dire que les entrées A,B,C du décodeur indiquent **qui** est désigné pour l'échange sur le bus, alors que les validations G1, $\overline{G2A}$ et $\overline{G2B}$ disent **quand** cet échange a effectivement lieu.

Un générateur d'impulsions

Outre l'usage pour désigner des mémoires ou des ports d'entrées/sorties,

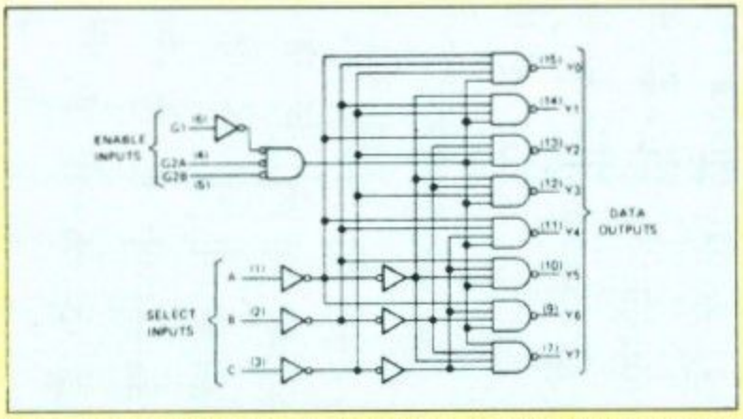


Fig. 18. – Brochage et schéma interne du décodeur ultra-classique 74LS138. Notez la validation collective des 8 NANDs par la combinaison des lignes « G ». Le bon usage des entrées « G » affranchira le décodeur de tout soi-disant « parasite ».

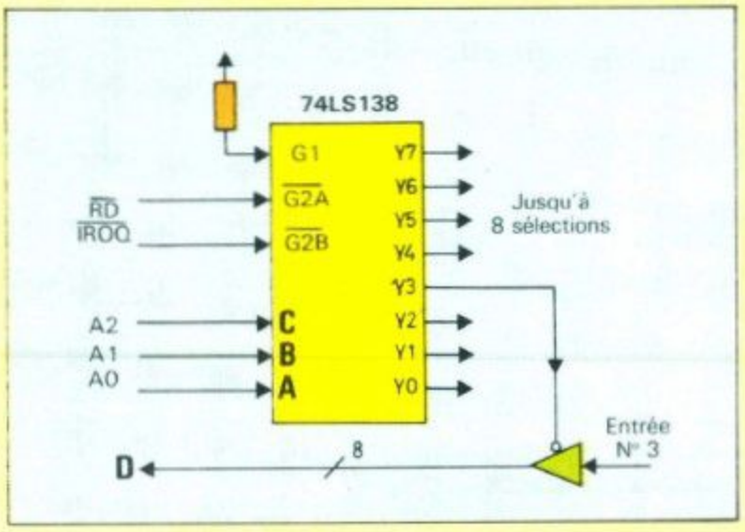


Fig. 19. – Validation de l'entrée n° 3 d'un montage hypothétique autour d'un Z80. Notez la polarisation de l'entrée de validation G1, qui est rendue active en permanence puisque deux signaux actifs au niveau bas suffisent à « certifier conforme » la reconnaissance d'adresses. Le montage suffit pour huit entrées distinctes.

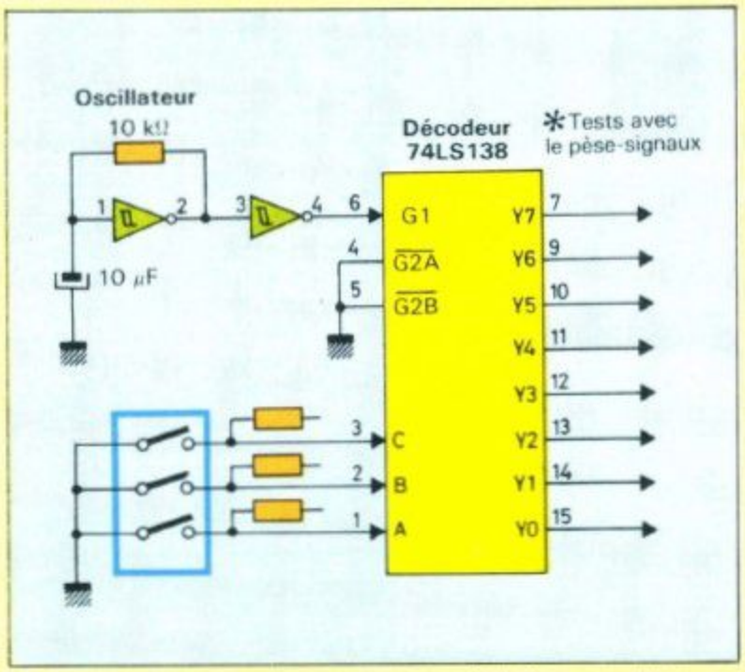


Fig. 20. – L'oscillateur valide cycliquement le décodeur via G1 ; les deux autres validations sont permanentes (à la masse). Les commutateurs désignent une sortie sur les huit, qui va « battre » comme l'oscillateur.

un décodeur peut être en lui-même un organe de sortie ayant un sens. Lorsque l'on s'en sert comme **générateur** d'impulsions **adressable (fig. 20).**

Notre montage d'expérience comporte une source d'impulsions familière, simple anneau RC autour d'un inverseur 74C14. Cette « horloge » entre dans le décodeur par le point de validation positive G1 ; choix tout arbitraire : on aurait pu tout aussi bien passer par $\overline{G2A}$ ou $\overline{G2B}$.

Les entrées d'adresses sont reliées à trois commutateurs d'une rangée DIP.

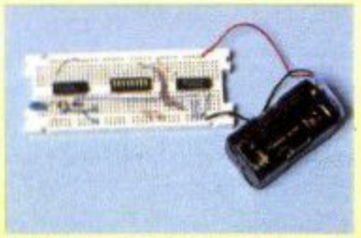
Le fonctionnement est ultra-simple : une et une seule des sorties Y_0 à Y_7 « bat » au rythme de notre oscillateur, celle qui est adressée par les commutateurs connectés aux points A, B, C. Et l'on vérifie la numérotation binaire en explorant les huit combinaisons :

C B A	Sortie active
0 0 0	Y_0
0 0 1	Y_1
0 1 0	Y_2
0 1 1	Y_3
1 0 0	Y_4
1 0 1	Y_5
1 1 0	Y_6
1 1 1	Y_7

Il suffit pour cela, et pour chacune des huit combinaisons, de parcourir avec le pèse-signaux les huit sorties en question.

Un dispositif pratique

Sans faire de cette série de Fiches un recueil de « notes d'application », on y montre de temps en temps que tel montage d'apparence fort simple est



Réalisation d'un générateur d'impulsions adressables.

directement applicable, y compris dans des applications professionnelles.

Nombre d'appareils automatiques fonctionnent selon une liste d'opérations fixe, avec un petit nombre de choix. Ainsi, les distributeurs de billets simples indiquent-ils ce que vous devez faire en allumant un voyant en regard d'un texte du genre :
INTRODUISEZ VOTRE CARTE, ou ERREUR, APPUYEZ SUR LA TOUCHE C, etc.

Un clignoteur attire mieux l'attention qu'une lumière fixe. Ceci posé, notre montage d'expérience peut parfaitement être utilisé tel quel devant un microprocesseur ; sous commande du programme, via quatre lignes d'un port en sortie, il sera possible **(fig. 21) :**
– d'éteindre tous les voyants par un « 1 » sur la ligne qui commande $\overline{G2A}$,
– d'en faire clignoter un seul, choisi via les lignes qui commandent A, B, C, dans le cas contraire.

Avec un choix judicieux de couleurs : LED verte pour une opération « normale », rouge pour signaler une erreur, orange pour « lever un doute », ce montage rudimentaire peut être aussi « intelligent » (ou perçu comme tel) que des messages sur un écran. Et bien moins cher...

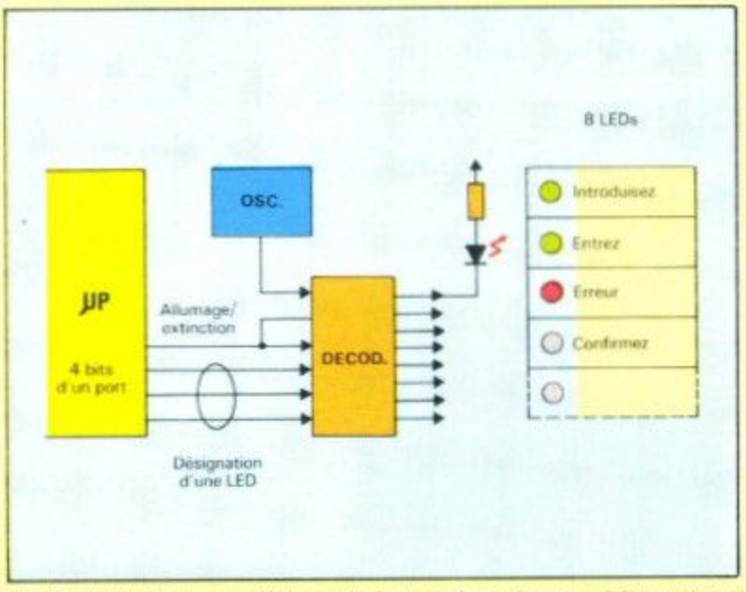


Fig. 21. – Application pour un guide à voyants clignotants. Le montage pourrait être exactement celui de la figure 20, à ceci près que les validations G2 sont reliées au microprocesseur pour permettre l'extinction globale des voyants sous contrôle du programme.

POUR CEUX QUI VEULENT ALLER PLUS LOIN

De l'algèbre de Boole...

Nos lecteurs s'en sont aperçus, nous ne faisons guère usage des expressions en forme d'algèbre de Boole, dont il est fait tant de cas chez d'autres auteurs (et dans les manuels scolaires ou universitaires).

Ce que nous avons souhaité, c'est que le lecteur ne soit pas abusé par cette Algèbre, qui est un merveilleux instrument **théorique,** mais n'est pas un modèle réaliste de la logique électronique, car on ignore dans ses « équations » trop de phénomènes réels. Notamment, le temps qui passe...

Ce n'est pas être injuste avec Boole et ses successeurs. Ils ont fait un travail immense, et prodigieusement utile pour l'étude des logiques compliquées.

Il s'agit simplement de faire la part des abstractions un peu trop idéales, et des dures réalités du monde physique réel...

... aux réseaux logiques

Dans la Fiche 8B, nous avons (un peu) attiré l'attention sur la **structure régulière** du décodeur de la figure 12, avec ses lignes d'entrée directes et complémentaires, croisées avec les lignes qui déterminent des ET (ce sont des NANDs, mais cela revient au même).

Les mathématiques introduites par Boole permettent de prouver que **toutes** les combinaisons de variables (« entrées ») logiques : 0/1, VRAI/FAUX ou ce que vous voudrez, peuvent s'effectuer en deux étapes « normalisées » :

– effectuer des ET des variables ou de leur inverse,

– effectuer des OU des résultats de ces ET.

Notre encadré spécial pour matheux donne de cette assertion une présentation plus conforme aux habitudes.

Les électroniciens ont su en tirer les conséquences, c'est-à-dire trouver des arrangements **en réseau** susceptibles de réaliser **n'importe quelle** fonction « logique » ; dans leur jargon, ce sont des PLA *(Programmable Logic Arrays).*

Les réseaux logiques programmables

Ainsi que notre **encadré** le présente, il existe un schéma « universel » dans lequel il n'y a plus qu'à indiquer « par des croix » telle ou telle fonction logique ; il suffit qu'il y ait assez d'entrées... et de termes !

De tels schémas ne sont pas seulement des vues de l'esprit. **Il existe des composants** qui correspondent, trait pour trait, à ce schéma universel ; chez *Texas Instruments* et *Signetics,* notamment.

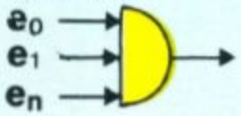
Les lignes ET/OU sont réalisées avec des artifices du type « collecteur ouvert » ; nous connaissons les « ET câblés », les « OU câblés » s'obtiennent... avec les compléments.

Quant aux croix qui marquent les connexions effectuées, elles mettent en jeu des fusibles microscopiques qui relient les « lignes » aux « colonnes ».

Ces composants se « programment » comme des mémoires PROM ; on brûle sélectivement les fusibles qui **ne doivent pas** relier telle ligne à telle colonne.

Bien malin celui qui décide qu'en établissant les fonctions de tels réseaux logiques programmables, on fait du hard ou du soft !

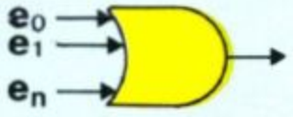
La formule mathématique pour ce ET en notation électronique



est, avec le symbole pour « ET » :

$$(e_0 \wedge e_1 \cdots \wedge e_n)$$

Pour le OU, l'équivalence des notations est la suivante :



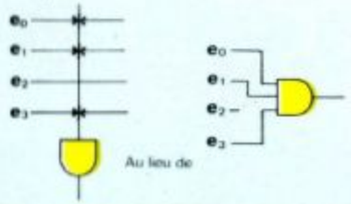
$$(e_0 \vee e_1 \cdots \vee e_n)$$

On montre alors que **toutes** les expressions logiques impliquant les variables logiques e_0, e_1, ... e_n peuvent se décrire sous une **forme normale** (ce qui veut dire « normalisée », en langage courant) où interviennent :

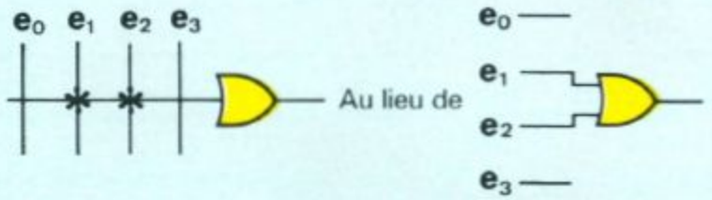
– des $\wedge$ des variables d'entrée ou bien de leur complément (certaines variables peuvent ne pas intervenir),

– le $\wedge$ des résultats de ces $\wedge$.

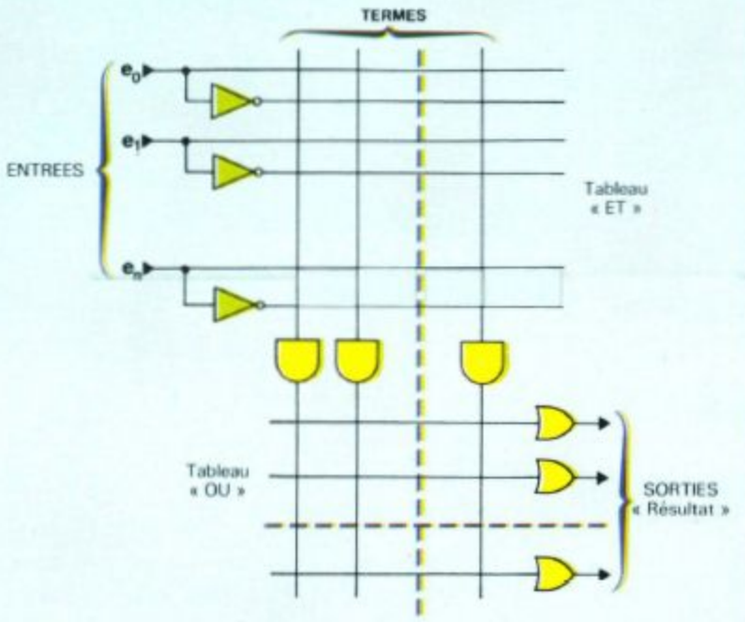
La formule mathématique imprimée, pour cette formulation, est rigoureusement indigeste. Nous y préférerons un schéma d'allure « électronique » avec des simplifications de dessin. Le ET sera une simple ligne où des croix montrent quels termes sont pris en compte :



Pour le OU, on prendra la notation graphique suivante, qui a le même principe :



alors, n'importe quelles expressions logiques où entrent les variables e_0, e_1... e_n peuvent s'effectuer avec un réseau qui comporte assez de ET et de OU montés comme ceci :



Il ne manque plus que les croix pour indiquer à quelles(s) entrée(s) sont « connectés » les différents **termes** du « tableau-ET » ; chaque ET est relié :

– soit à une entrée e_i,

– soit à son complément $\bar{e}_i$ (via l'inverseur),

– soit... à aucune d'entre ces deux lignes : en anglais *don't care.*

Même chose pour les OU, qui prendront ou ne prendront pas en compte tel ou tel terme.

Le mathématicien exprimera ainsi selon les **lois de De Morgan** la sélection de notre Fiche 8A, figure 3 :

$$(\overline{A_{19}} \quad \overline{A_{18}}) \wedge (A_{17} \quad A_{16})$$

L'expression sera « bricolée » pour arriver à la forme normale :

$$(\overline{A_{19}} \quad \overline{A_{18}} \quad A_{17}) \vee (\overline{A_{19}} \quad \overline{A_{18}} \quad A_{16})$$

Graphiquement, le **réseau logique** suivant donne le résultat, avec deux « termes ET » et le « OU » :