# ES102/PC3 : énoncé et corrigé

## 1) Fonctions décroissantes

1a) Soit f∈$F_n$ représentée sur le n-cube. Comment se caractérise une fonction décroissante ?

On rappelle qu'on a ainsi représenté les fonctions majorité et parité (donc pour n=3) en PC1. Une arête du n-cube relie 2 points de $B^n$ entre lesquels une seule variable (parmi les *n*) diffère. Parcourant cette arête dans le sens « montant » où ladite variable passe de 0 à 1, une fonction décroissante ne passe jamais de 0 à 1 : soit elle reste constante, soit elle passe de 1 à 0.

1b) Soit f∈$F_n$ décroissante non constante. Montrer que f(1–1)=0 et f(0–0)= 1.

N.B. : 0–0 (resp. 1–1) désigne le n-uplet avec toutes les variables à 0 (resp. à 1).

f non constante ⇒ $\exists z \in B^n$ f(z)=0 et $\exists u \in B^n$ f(u)=1. En élevant une à une à 1 les variables à 0 de *z*, on établit un chemin sur le n-cube « montant » de *z* à 1–1, le long duquel f reste nécessairement à 0 par décroissance. Donc f(1–1)= 0. Inversement, en baissant une à une à 0 les variables à 1 de *u*, on établit un chemin sur le n-cube « descendant » de *u* à 0–0, le long duquel f reste nécessairement à 1. Donc f(0–0)= 1.

1c) Quelles sont donc les fonctions f décroissantes non constantes de $F_2$ et les noms des portes correspondantes ?

On a donc f(0,0)=1 et f(1,1)=0. Sur la table de vérité, il reste donc deux cases à remplir. Il s'avère qu'elles peuvent l'être de façon arbitraire sans nuire à la décroissance de f (ce ne serait plus aussi simple pour n>2). Cela correspond aux fonctions x', y', x'y' et x'+y'. Les 2 premières correspondent à l'inverseur et les 2 dernières respectivement aux portes NOR et NAND.

## 2) Premières portes CMOS

Une porte CMOS est un montage constitué de transistors MOS qui connectent sa sortie soit à la masse soit à l'alimentation pour produire une valeur soit 0 soit 1 selon les entrées appliquées sur les grilles des transistors. Une telle porte implante donc une fonction booléenne particulière.

2a) Faire un bref récapitulatif des « services binaires » offerts par les transistors nMOS et pMOS et comprendre d'abord pourquoi ils permettront seulement d'implanter des fonctions booléennes décroissantes par rapport aux variables binaires appliquées sur leurs grilles.

Grâce aux électrons (charges **n**égatives) qu'il peut véhiculer, un transistor **n**MOS (de type *n*) permet d'établir une connexion peu résistive vers la masse (source d'électrons) si sa grille reçoit un 1 logique. Sinon il est non passant. Il sera donc utilisé au sein d'une porte pour réaliser une ou plusieurs valeurs 0 de la fonction implantée, en connectant à la masse la sortie de la porte.

Grâce aux trous (charges **p**ositives) qu'il peut véhiculer, un transistor **p**MOS (de type *p*) permet d'établir une connexion peu résistive vers l'alimentation (source de trous) si sa grille reçoit un 0 logique. Sinon il est non passant. Il sera donc utilisé au sein d'une porte pour réaliser une ou plusieurs valeurs 1 de la fonction implantée, en connectant à l'alimentation la sortie de la porte.

La montée à 1 d'une entrée ne peut qu'établir une connexion à la masse (côté nMOS) ou en rompre une à l'alimentation (côté pMOS). Symétriquement, la descente à 0 d'une entrée ne peut que rompre une connexion à la masse ou en établir une à l'alimentation. La relation de cause à effet est décroissante dans tous les cas.

2b) Du coup, comment va-t-on pouvoir implanter des fonctions croissantes ?

En composant 2 fonctions décroissantes, on en obtient une croissante. Ainsi, 2 inverseurs (la complémentation est décroissante) en série implantent la fonction identité (croissante).

Notons que si le complément a' d'une variable *a* est appliqué sur une grille de transistor, alors l'effet sera bien décroissant par rapport à a', et donc croissant par rapport à *a*. Mais a' doit être produit par un inverseur à partir de *a*, ce qui suppose bien une composition.

Attention, si on se sert à la fois de *a et a'* dans une porte CMOS, on pourra obtenir des fonctions ni croissantes, ni décroissantes par rapport à *a*, ce qui est une très bonne nouvelle…

2c) Et si les transistors avaient été « croissants » ?

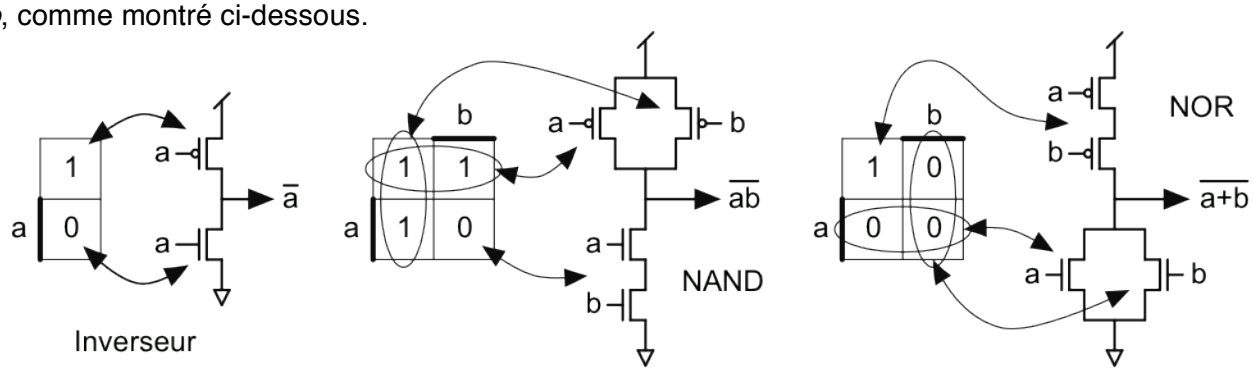
Alors seules des fonctions croissantes auraient pu être implantées, que ce soit au sein d'une seule porte ou par composition de plusieurs. Il aurait été impossible d'implanter les autres, qui constituent pourtant l'écrasante majorité des fonctions booléennes. Le caractère « décroissant » des transistors était donc indispensable, en fait !

2d) Repartant de l'inverseur CMOS vu au CM3, implanter en transistors les portes NAND et NOR.

Comme indiqué en cours, la structure d'une porte CMOS repose sur une séparation des 0 et des 1 de la fonction implantée, les premiers étant réalisés par un bloc *n* reliant masse et sortie, les deuxièmes par un bloc *p* reliant alimentation et sortie. Chez l'inverseur CMOS, chacun de ces blocs est constitué d'un unique transistor. Pour NAND et NOR, il va en falloir plusieurs.

Au sein d'un bloc, il est possible de mettre plusieurs branches en parallèle pour couvrir plusieurs cas. Pour la porte NAND, côté bloc *p*, les 1 correspondant à a=0, resp. b=0, sont réalisables par un simple pMOS commandé par *a*, resp. b (comme chez l'inverseur). En mettant en parallèle ces 2 pMOS, comme montré ci-dessous, on couvre tous les 1 de (ab)'. Le cas a=b=0 est particulier en ce que les 2 pMOS relient simultanément alimentation et sortie. Mais cela ne pose aucun problème : le 1 en sortie n'en est que mieux établi.

Le 0 de (ab)' correspond au cas a=1 et b=1. Le bloc *n* doit donc relier masse et sortie *ssi* on a la conjonction de ces 2 conditions. Cela se réalise en mettant en série 2 nMOS : la connexion ne s'établira que si les 2 sont passants. Il suffit alors de les commander respectivement par *a* et *b*, comme montré ci-dessous.




Le cas de la porte NOR est similaire, mais avec mise en série côté *p* et en parallèle côté *n*.

Remarque : ce sont donc les 3 fonctions décroissantes de $F_1$ ou $F_2$ que nous avons implanté et il s'avère bien que jamais les compléments des variables ne sont utilisés dans les montages.

2e) Comment étendre ces portes à 3 variables ?

Il suffit d'ajouter un transistor en série et un en parallèle, commandés par la troisième variable.

L'*état passant* d'un transistor est défini comme la fonction booléenne qui vaut 1 *ssi* il est passant. Cette notion se généralise aux branches et blocs de transistors, devenant la fonction caractéristique des différents jeux d'entrée qui rendent passant le transistor, la branche, le bloc.

2f) Exprimer les états passants des blocs *n* et *p* pour les portes inverseur, NAND et NOR.

Inverseur : a (n) et a' (p). NAND : ab (n) et a'+b' (p). NOR : a+b (n) et a'b' (p).

2g) Quels liens y a-t-il entre états passants des blocs *n* et *p* et fonction f implantée ?

Soit $EP_p$, resp. $EP_n$, l'état passant du bloc p, resp. n. Le bloc *p* réalise les 1 de f, donc $EP_p$=f. Inversement, le bloc *n* réalise les 0 de f, donc son complément /f. D'où $EP_n$=/f.

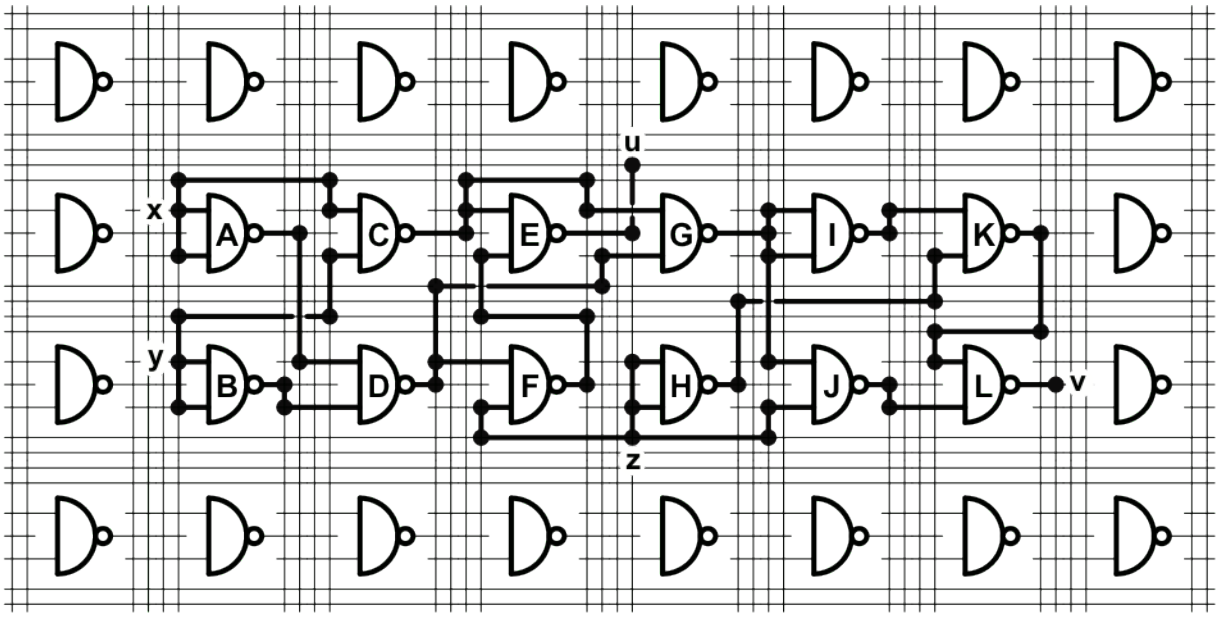
2h) Comment réaliser une porte AND (ET) ?

La fonction ET est croissante. L'exercice 1 nous a préparé au fait qu'il faudrait utiliser des combinaisons de portes dans ce cas. La solution la plus simple est de mettre un inverseur en sortie de la porte NAND. Une autre solution aurait été de complémenter les entrées de la porte NOR, mais cela coûte plus cher : 2 inverseurs (un pour *a*, l'autre pour *b).*

## 3) Utilisation d'un NAND Gate Array

La figure suivante montre une petite portion (8×4) d'une matrice périodique de portes (*Gate Array*). Il s'agit ici de portes NAND. Leurs entrées et sorties peuvent être connectées arbitrairement (dans certaines limites) via un réseau de communication configurable.

Ce réseau est constitué d'un ensemble de canaux, représentés en trait fin ci-dessous, dont les différentes sections sont des connexions électriques. Celles qui sont exploitées sont représentées en trait épais. Canaux horizontaux et verticaux courent à deux niveaux différents et ne se touchent donc pas quand ils se croisent. Mais on peut les connecter au moyen d'une « soudure », représentée ci-dessous par un point noir.

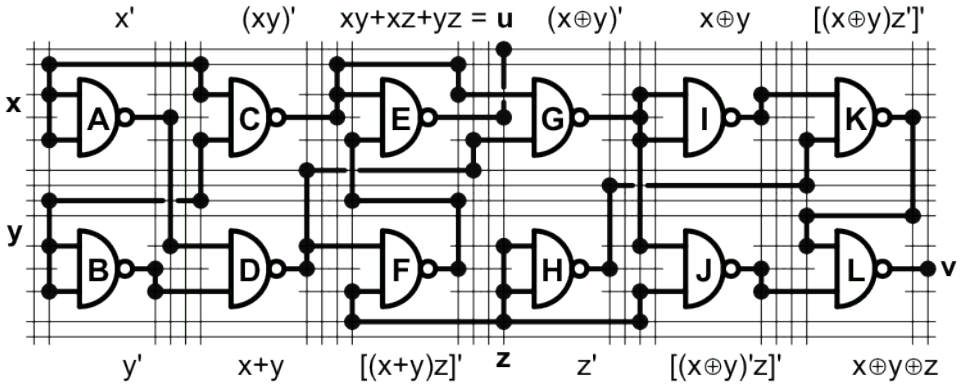



Un circuit d'entrées *x*, *y* et *z* et sorties *u* et *v* y a été implanté. Quelle est sa fonction ?

Illustrant « l'omnipotence » de la porte NAND, le présent exercice sensibilise également à la réalité du *routage* des signaux à bord des cartes et puces électroniques. Enfin, ce *NAND Gate Array* préfigure un composant bien plus évolué, appelé FPGA, pour *Field Programmable Gate Array*, dans lequel le réseau de communication est géré logiciellement. Dans un tel FPGA, ce sont des transistors, pilotés par des signaux de configuration, qui réalisent les connexions. La « maille périodique » y contient des centaines de transistors (par opposition aux 4 utilisés ici par la porte NAND). Ces FPGA jouent aujourd'hui un rôle majeur en électronique numérique, pour le développement, et la production en moyenne série.

Pour déterminer la fonction réalisée par le circuit, on peut progresser systématiquement des entrées vers les sorties et déterminer la fonction booléene réalisée par chaque porte NAND, au prix d'un certain travail d'écriture logique et d'application de loi de De Morgan. Le résultat de ce

travail apparaît sur les côtés haut et bas de la figure ci-dessous. Il s'avère que u = Maj(x, y, z) et que v = Par(x, y, z). Il s'agit donc d'une implantation du Full Adder (FA).




Dans le travail ci-dessus, l'intuition qu'il s'agit d'un Full Adder apparaît plus ou moins tôt et incite à faire le rapprochement avec le travail fait en fin de PC2 (pour ceux qui sont allés jusque là), dans lequel l'usage de portes NAND a déjà été privilégié. Ainsi, le chemin qui va de *z* à *u* par les 2 portes NAND *F* et *E* rappelle celui élaboré pour rapprocher retenues entrante et sortante. On réalise alors que les portes *A*, *B* et *D* implantent la porte OU des montages achevant la PC2 et que le reste est identique pour le calcul de la retenue sortante. Tout ceci occupe la moitié gauche ici. Côté droit, le groupe de portes HIJKL implante la porte XNOR du dernier montage de la PC2.

Il existe encore une troisième voie, peut-être la plus efficace, pour analyser le présent circuit : on sait que chaque petit arbre de 3 portes NAND (sur 2 étages), donc à 4 entrées a, b, c et d, calcule ab+cd. Voilà qui évite De Morgan. Si l'on met à part les portes A, B, H et I, qui jouent le rôle de simples inverseurs, il en est ainsi des « trios » CEF, CGD et KLJ, dont les sorties peuvent ainsi être très vite établies.

Revenant à la comparaison avec la PC2, il a donc fallu ici pousser plus loin le *jeu de bulles*, pour n'avoir plus que des portes NAND. Torturer ainsi les formules pour pouvoir exploiter les seules ressources disponibles constitue ce que l'on appelle un travail de *mapping* dans le jargon électronicien.

Un dernier mot pour constater que la sortie retenue sortante *u* se trouve sur la même colonne que la retenue entrante *z*. Ce n'est pas un hasard. Cette configuration permet directement d'abouter les FA les uns au dessus des autres pour réaliser un additionneur numérique sur notre *NAND Gate Array*. On parle de travail de *placement-routage*.