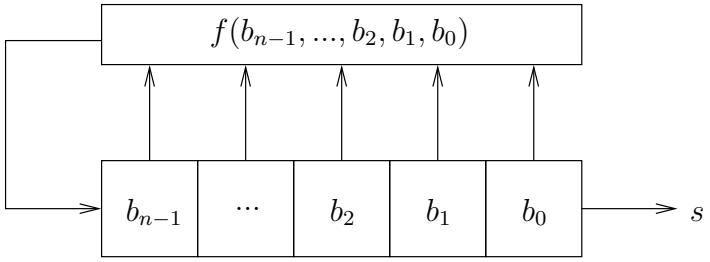
TD 3
Mémoires et macroblocs

Ex. 1 : Générateur de nombres aléatoires

On travaille dans cet exercice sur un composant appelé registre à décalage à rétroaction linéaire (*Linear Feedback Shift Register*) servant à générer des séquences « pseudo-aléatoires » de bits. Le principe de ce composant est détaillé dans la figure ci-dessous.



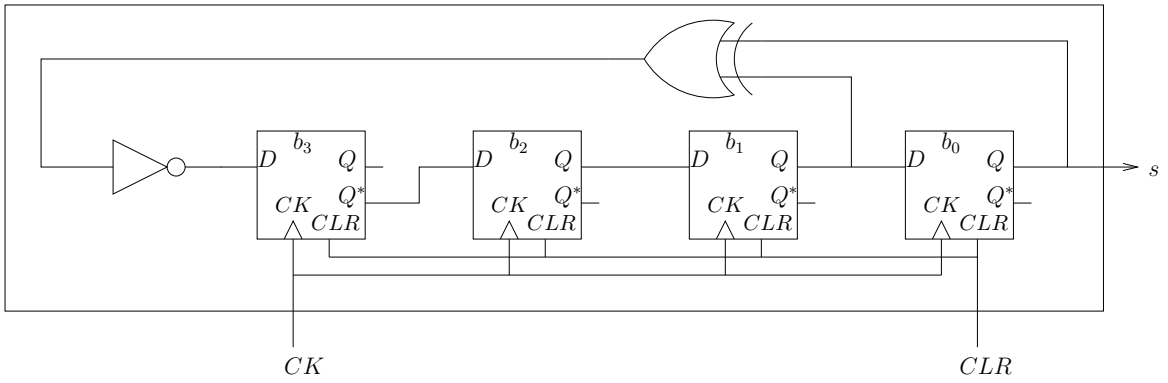
Chaque case correspond à une bascule D : il y a donc n bascules numérotées de 0 à $n-1$, n étant le nombre de bits du LFSR. La sortie s est le bit pseudo-aléatoire généré par le LFSR : elle affiche la valeur de la bascule numéro 0. A chaque cycle, on charge le contenu de la bascule i dans la bascule $i-1$: si on considère la séquence des valeurs contenues dans les bascules comme une valeur $B = b_{n-1}...b_1b_0$, alors B est décalée d'un bit vers la droite à chaque cycle. La valeur insérée dans la bascule $n-1$ est calculée en fonction des valeurs de chaque bascule via une fonction de rétroaction f définie par $f(b_{n-1}, ..., b_2, b_1, b_0) = c_{n-1}.b_{n-1} \oplus ... \oplus c_1.b_1 \oplus c_0.b_0$ où les c_i sont des constantes binaires données.

Dans la suite de l'exercice, on pose $n = 4$ et $f(b_3, b_2, b_1, b_0) = 0.b_3 \oplus 0.b_2 \oplus 1.b_1 \oplus 1.b_0 = b_1 \oplus b_0$.

Question 1 On suppose que les bascules sont initialisées avec les valeurs suivantes : $b_3 = 1$ et $b_2 = b_1 = b_0 = 0$. Remplir un tableau contenant les valeurs des bascules en fonction du temps jusqu'à ce qu'on retombe sur les valeurs initiales.

Question 2 Que se passe-t-il si on se retrouve avec $b_3 = b_2 = b_1 = b_0 = 0$? Exprimer le nombre maximum de valeurs différentes possibles avant de retrouver les valeurs initiales en fonction de n.

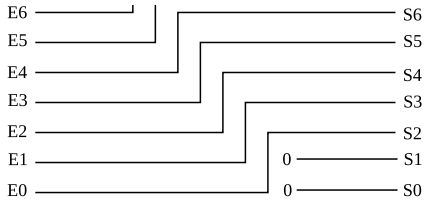
Question 3 Une réalisation d'un LFSR 4 bits (avec la fonction de rétroaction : $f(b_3, b_2, b_1, b_0) = b_1 \oplus b_0$) à l'aide de 4 bascules D est proposée ci-dessous.



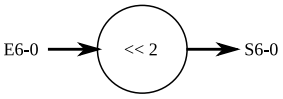
Justifier l'utilisation de l'inverseur et de la connexion à $\overline{Q}$ sur la bascule $b3$.

Ex. 2 : Décaleur à barillet

Un décalage d'un nombre constant de bits ne nécessite aucune porte logique : il s'agit de connecter les fils de façon adéquate (voir schéma ci-dessous).



Décalage à gauche de 2 positions



Symbole

Un décaleur à barillet (ou *barrel shifter*) est un opérateur combinatoire à deux entrées x et p qui permet de décaler un nombre x, codé sur n bits, d'un certain nombre de bits $p < n$ vers la gauche ou la droite. Notez qu'au contraire d'un décalage constant, un décaleur à barillet fonctionne pour toutes les valeurs de p. On note typiquement :

- $x << p$ le décalage de x de p bits vers la gauche en remplissant les cases libérées à droite par des 0 ;
- $x >> p$ le décalage de x de p bits vers la droite en remplissant les cases libérées à gauche par le bit de signe du nombre x initial (on parle de décalage arithmétique) ;
- $x >>> p$ le décalage de x de p bits vers la droite en remplissant les cases libérées à gauche par des 0 (on parle de décalage logique).

Question 1 Sur combien de bits doit être codé le décalage p si on suppose que x est codé sur 32 bits ? En déduire la relation générale entre le nombre de bits de x (n) et de p (m). Mathématiquement parlant, à quelles opérations élémentaires correspondent les différents décalages ?

Question 2 Proposer l'implantation d'un décaleur à gauche d'une entrée x pour $n = 2$. Quelle porte élémentaire est idéale pour cette implantation ?

Question 3 En vous basant sur le résultat précédent, proposer une implantation pour $n = 4$ et $n = 8$. Évaluer le circuit : nombre de couches logiques à traverser et complexité en surface, l'unité étant le mux 1 bit $2 \rightarrow 1$.

Pour aller plus loin...
Question 4 On veut maintenant gérer le décalage logique vers la droite. On ajoute pour cela une entrée d telle que $d = 1$ si on effectue un décalage à droite. Modifier le décaleur 4 bits pour gérer le décalage logique à droite en plus du décalage à gauche. Attention, il y a deux solutions :

- la solution la plus évidente est de choisir à chaque étage un bit de l'étage précédent en fonction du sens du décalage à effectuer,
- la solution avec "miroirs" : on utilise le décaleur à gauche vu à la question 3. Si on doit décaler à droite, on permute les bits de l'entrée ("miroir" : on met en entrée du décaleur les bits 0 à 3 au lieu des bits 3 à 0), puis on effectue sur cette nouvelle valeur un décalage à gauche ; les bits du résultat obtenu sont permutés à nouveau en sortie.

Evaluer les deux solutions.

Pour aller plus loin...
Question 5 On veut enfin ajouter la gestion du décalage arithmétique à droite. On dispose d'une nouvelle entrée a valant 1 si on veut effectuer un décalage arithmétique lors d'un décalage à droite. On peut remarquer que le décalage à gauche est toujours arithmétiquement valable,

indépendamment de la valeur de a. Modifier le schéma du décaleur gauche/droite 4 bits pour y ajouter la gestion du bit de signe dans les décalages à droite.

Ex. 3 : Étude d'une mémoire vidéo

On cherche à afficher une image bitmap sur un écran via le port VGA. L'image qui nous intéresse possède une résolution de 320×240, avec 8 bits par pixel. Cette image est stockée dans une mémoire contenant des mots de 32 bits. On peut donc coder 4 pixels par mot mémoire.

Question 1 Combien de mot mémoire sont nécessaires pour enregistrer l'image ? Sur combien de bits doit-on coder les adresses ?

On suppose que la couleur d'un pixel est codée suivant ses trois composantes R(rouge), V(vert) et B(bleu) (dans cet ordre là). Le codage choisi ici permet de désigner 4 nuances de rouge, 8 nuances de bleu et 8 nuances de vert.

Question 2 Comment coder le blanc, le noir, un vert vif ?

On suppose que les pixels sont rangés dans la mémoire à la suite des uns des autres, les quatre premiers pixels (de coordonnées $0 \leq X \leq 3$, $Y=0$) à l'adresse 0, les quatre suivants (de coordonnées $4 \leq X \leq 7$, $Y=0$) à l'adresse 1, et ainsi de suite. Le premier pixel de la ligne suivante étant stocké à la suite du dernier pixel de la ligne courante.

Question 3 À quelle adresse en mémoire est stockée la couleur du pixel de coordonnées (x, y) ? Dans quel octet du mot mémoire à cette adresse ? Comment calculer cette adresse et cette position d'octet efficacement par un circuit ?

Question 4 On souhaite positionner le pixel de coordonnées $(100, 3)$ avec la couleur blanche. Quels sont les signaux (Adresse, données, commandes) à fournir à la mémoire pour obtenir ce résultat ? Faites attention à ne pas modifier les pixels voisins !

L'écran vers lequel est envoyée l'image supporte le système de couleur *High color* : 32 nuances de rouge et de bleu, 64 nuances de vert. Le vert est favorisé car l'oeil humain y est plus sensible.

Question 5 Comment convertir les composantes de couleurs de notre image en format *High Color* ? On veillera à ce que les nuances soit réparties régulièrement sur le spectre des nuances possibles pour chaque couleur.