

# Electronique numérique

## Chemins de données ou *datapath*

### CORRECTIONS

*Tous les exercices ne seront pas forcément résolus en TE.*

## 1 Notion de chemin de données ou *datapath*

Nous avons déjà abordé le notion de "chemins de données" dans le "jeu des cambrioleurs" : il s'agissait uniquement d'acheminer, en combinatoire, une donnée d'une entrée vers une sortie, sans traitement spécifique. En général, les "datapath" sont plus complexes et associent à la fois logique séquentielle et logique combinatoire. Les datapaths permettent d'effectuer des calculs rapides et très puissants. L'essentiel de leur puissance de calcul provient de deux notions fondamentales :

- Le **parallélisme** : il s'agit d'effectuer plusieurs opérations durant 1 seul cycle d'horloge, grâce à l'utilisation de plusieurs opérateurs matériels. Pour rappel, dans un processeur classique (et comme dans les langages dits "séquentiels"), les opérations se font en séquence, les unes à la suite des autres. Ici, à l'inverse, nous sommes en mesure de construire des calculateurs qui effectuent plusieurs opérations simultanément.
- Le **pipeline** : il s'agit d'effectuer des opérations "à la chaîne", sur des données, chaque *étape* du pipeline ayant une action spécifique sur ces données. Nous allons d'abord chercher à illustrer le rôle de la bascule D dans la structure d'un pipeline élémentaire : le registre à décalage.

## 2 Pipeline, Registres à décalage et Applications

Le registre à décalage le plus simple consiste à relier la sortie d'une première bascule D à l'entrée d'une seconde bascule D. Ceci se généralise à un nombre  $L$  de bascules ainsi enchaînées les unes aux autres (sans rebouclage). Une donnée  $e$  qui se présente au cycle  $n$  mettra  $L$  cycles (ou "coups d'horloge") à parvenir à la sortie  $s$ . Sa fonction est double :

1. Retarder la donnée  $a_0$  d'un temps discret  $L$  : elle se présente sur l'entrée  $e$  au temps  $t$  et ressort sur la sortie  $s$  du dispositif au temps  $t + L$ .
2. Autoriser une donnée  $a_1$  à se présenter sur l'entrée  $e$  au temps  $t + 1$ , en même temps que  $a_0$  est échantillonnée par la seconde bascule, etc.

**Exercice 1 : registre à décalage simple** Dessiner un registre à décalage de taille 4. Nommer correctement ses signaux et écrire l'équation de la sortie  $s(t)$ .

*Corrections*

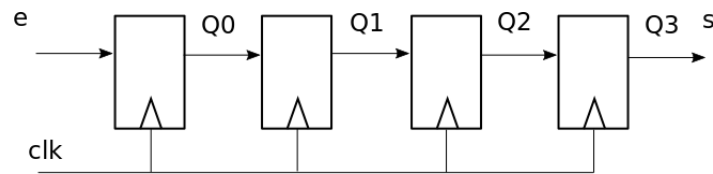


FIGURE 1 – Registre à décalage de profondeur 4.

L'équation de sortie est :

$$s(t) = e(t - 4)$$

Le chronogramme est le suivant :

cycle	0	1	2	3	4	5	...
e	$a_0$	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	...
s	?	?	?	?	$a_0$	$a_1$	...

**Exercice 2 : registre à décalage commandé** Modifier le registre à décalage précédent de manière à le commander par un signal 'push', qui autorise les données à s'acheminer progressivement vers la sortie lorsque ce signal vaut '1'.

*Corrections*

**Exercice 3 : moyenne mobile** Soit un flux séquentiel de données (boursier, médical, etc), dont les données successives arrivent au rythme de 100 Mhz. La moyenne mobile (ou moyenne glissante, ou *sliding average* en anglais) d'un tel flux continu permet de lisser une moyenne au cours du temps.

1. Calculer rapidement la moyenne mobile, sur 4 échantillons, du flux suivant :

..., 0, 0, 0, 0, 8, 9, 13, 4, 3, 2, 1, 0, 0, 0, ....

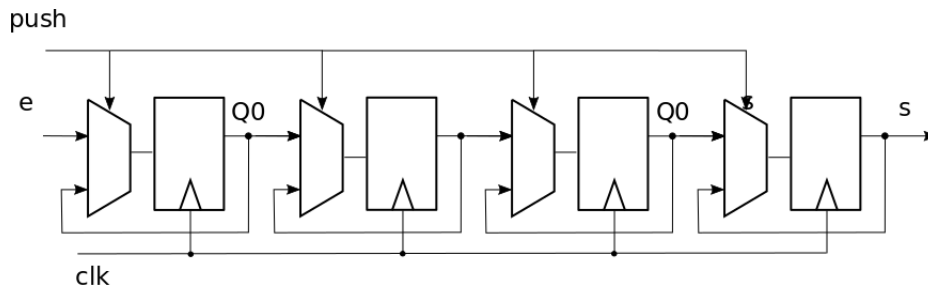


FIGURE 2 – Registre à décalage de profondeur 4, avec signal de contrôle 'push'

2. Concevoir un circuit numérique qui réalise le calcul d'une moyenne mobile sur 4 échantillons de nombres entiers, supposés non signés. Proposer une seconde solution. Laquelle est meilleure ? On s'autorise à utiliser la notion d'additionneur, mais pas de diviseur.
3. Combien de bits sont nécessaires pour calculer la somme de 4 données codées sur 8 bits ? Combien pour la moyenne mobile ?
4. Combien de bits sont nécessaires pour calculer la somme de  $n$  données codées sur  $m$  bits ?
5. Sachant que le temps de traversée de l'additionneur est 4 ns, quelle est la fréquence atteignable du circuit ? Est-ce compatible avec la fréquence d'arrivée des données ?

**Corrections :**

1. La moyenne donne

..., ?, ?, ?, 0, 2, 4, 7, 8, 7, 5, 2, 2, 0, 0, ....

2. Une solution consiste à réaliser les additions en parallèle. Une seconde à les chaîner. Il n'y a pas de meilleure solution, tant que la fréquence du circuit est atteinte. Il faut pour cela calculer le chemin critique dans les deux cas.
3. On se convainc assez facilement que pour 4 données de 8 bits, il faut 10 bits. Pour la moyenne mobile on est ramené à 8 bits.
4. Combien de bits sont nécessaires pour calculer la somme de  $n$  données codées sur  $m$  bits ?

**solution** Soit  $D(x)$  le nombre de bits nécessaire pour coder  $x$  (entier positif non signé). On a  $D(x) = \lfloor \log_2(x) + 1 \rfloor$ . Pour l'addition  $S$  de  $n$  données codées sur  $m$  bits, on a donc :

$$D(S) = \lfloor \log_2(n \cdot (2^m - 1)) + 1 \rfloor$$

5.  $4 + 4 = 8ns$ . On inverse : la fréquence atteignable est de 125 Mhz au maximum. C'est parfaitement compatible avec la fréquence de 100 Mhz : il faudra veiller à ce que le circuit fonctionne exactement à la même fréquence que l'émetteur des données.

**Exercice 4 : look-up table de FPGA** Les FPGA sont des circuits réguliers qui ont l'étonnante propriété d'être reprogrammables à volonté. On parle de "circuits reconfigurables". Les FPGA sont constitués d'un assemblage de briques élémentaires appelées BLE (Basic Logic Element), interconnectées en réseau. Leur topologie détaillée est présentée dans le photocopié du cours. Un BLE se compose notamment d'une LUT (look-up table) à  $n$  entrées : c'est une mémoire qui permet d'enregistrer les valeurs attendues de n'importe quelle fonction booléenne à  $n$  entrées. C'est l'équivalent matériel d'une petite table de vérité, qui se présente sous la forme d'un registre à décalage avec 'enable' (voir exercice précédent où on a appelé ce signal 'push')<sup>1</sup> à  $2^n$  bascules : le flux qui entre dans ce registre à décalage s'appelle un *bitstream*. Il agit comme une programmation ou *configuration* de la LUT. Chaque sortie  $Q_i$  des bascules de la LUT est connectée à une des  $2^n$  entrées d'un multiplexeur. Les signaux de contrôle du multiplexeur sont les  $n$  entrées, notées ici  $\{a, b, c, \dots, \alpha_{n-1}\}$ .

1. Dessiner une LUT 2.
2. On suppose que le bitstream est entré à l'aide d'un signal de contrôle auxiliaire 'push' (cf exercice 2) de la manière suivante : 1000, où le bit le plus à droite représente la première donnée. Calculer la sortie du multiplexeur pour les combinaisons des entrées  $a, b$ .
3. Quelle est la fonction réalisée par le dispositif?
4. Conclure en proposant un bitstream pour la fonction  $xor(a, b)$ .

**Solution :**

1. voir figure 3.

a	b	f
0	0	1
0	1	0
1	0	0
1	1	0

2. Les combinaisons de sortie sont :

3. C'est la fonction NOR2. Le bitstream nous a permis de configurer la LUT en NOR2!
4. Pour un XOR, le bitstream est 0110.

---

1. ...pour les besoins de notre exercice. Les LUTs des FPGAs sont réalisées au niveau transistors.

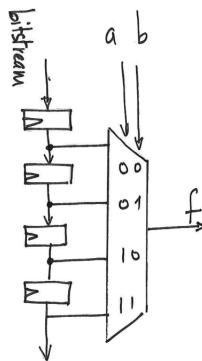


FIGURE 3 – LUT2 (2 entrées) – On a omis le bit 'enable' (ou 'push') pour simplifier le schéma, mais ce contrôle est nécessaire afin de bien séparer les phases de programmation (*configuration*) et d'utilisation.

### 3 Exo 5 : Pipeline et datapath

On cherche à concevoir un calculateur qui réalise l'opération arithmétique complexe  $y = a * b * c * d$  en un seul cycle d'une horloge à 100 Mhz. Les données  $a, b, c, d$  sont codées sur 8 bits et stockées au préalable dans des bascules D (appelées registres). Le résultat  $y$  est également stocké dans un registre.

1. Sur combien de bits est-il nécessaire de stocker  $y$  ?

**Solution** La multiplication s'écrit

$$M = \prod_{i=0}^{n-1} x_i, x_i \in \{0, \dots, 2^m - 1\}$$

D'où :

$$D(M_{max}) = \lfloor \log_2(2^m - 1)^n + 1 \rfloor = n \cdot \lfloor \log_2(2^m - 1) \rfloor + 1$$

2. Dessiner une première architecture, purement combinatoire.
3. Sachant que le temps de traversée d'un multiplieur est de 4 ns, quelle est la fréquence maximum de fonctionnement du calculateur ? Respectez vous le cahier des charges ?

**Solution** 125 Mhz. Même chose que précédemment.

4. Catastrophe ! Le temps de traversée du multiplieur est finalement revu à la hausse : 6 ns. Pouvez-vous corriger le circuit ? Respectez vous encore le cahier des charges ?

**Solution** 83 Mhz maximum : les données arrivent plus vite qu'on

ne peut les traiter ! Un moyen de corriger le circuit consiste à insérer des registres sur chaque arc reliant les multiplieurs : on "pipeline" le traitement. Par contre, on note que la moyenne mobile sort avec 1 coup d'horloge de retard (ce qui n'est probablement pas significatif ici) : on dit que le traitement se fait avec une *latence* d'un cycle, mais le débit (*throughput*) reste inchangé : à tous les cycles une donnée (moyenne mobile) sort. Noter que pour d'autres circuits, le fait d'introduire une latence est rédhibitoire : par exemple en TV numérique, les studio de télévision détestent introduire des images de retard dans leur pipeline.