TD 2 – Électronique Numérique

Logique Séquentielle

Notes et Conseils

L'objectif de ce TD est la découverte des circuits *séquentiels*. Nous commençons par découvrir sa brique fondamentale : la *bascule D*. Elle permet de mémoriser des signaux à des instants discrets. A travers différents exercices (dont l'un de *reverse engineering*), nous cherchons à appréhender la manière de concevoir différents circuits séquentiels : ce temps discret nécessite de penser en terme de *suites numériques*, en raisonnant sur l'instant présent et l'instant passé.

Première partie

Bascule D et circuits séquentiels élémentaires

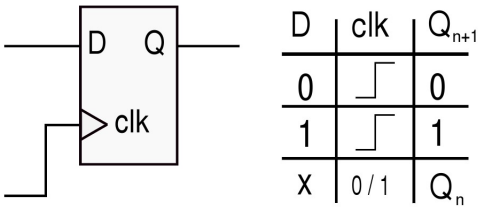
1 Découverte de la bascule D

Le paradigme dominant en conception de circuits numérique est le paradigme *synchrone* et *mono-horloge* : l'ingénieur organise sa réflexion et sa méthodologie de conception en reposant sur l'idée de la délivrance d'un *tempo* externe, parfait et régulier. Dans les faits, cet *tempo* est délivré par un signal carré d'horloge. Il existe certes des difficultés techniques à cette délivrance (*clock skew* notamment), mais le principe de *séparation des préoccupations* lui permet de les négliger.

Le circuit élémentaire sensible à l'horloge, et qui va permettre de concevoir des circuits séquentiels est la **bascule D**. C'est la *seule bascule qui va nous préoccuper ici*. C'est l'élément qui va nous permettre de distinguer *l'avant* de *l'après*. Dans votre esprit, il ne doit pas y avoir d'autre.

Toutefois, tout comme en Physique l'*atome* n'est finalement pas la brique *insécable* qu'on croyait, la bascule D possède bien évidemment une constitution interne : elle est souvent présentée comme le résultat d'un assemblage savant autour d'une autre bascule *asynchrone* cette fois, appelé *latch (verrou) RS*. En réalité, il existe beaucoup d'autres architectures VLSI (intégration à très grande échelle), qui utilisent des principes physiques différents pour mémoriser l'état. Nous prenons l'option pédagogique de nous concentrer uniquement sur la bascule D ici.

Le symbole de la bascule D, ainsi que sa table de vérité, sont rappelés ci-dessous.



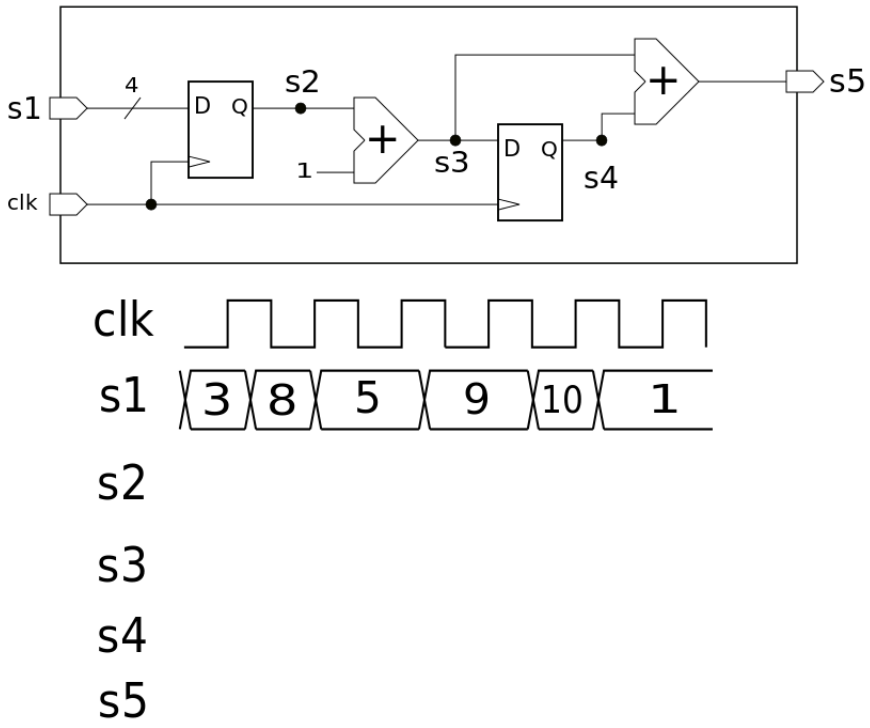
D	clk	Q_{n+1}
0	⎍	0
1	⎍	1
X	0 / 1	Q_n

La bascule D a pour fonction de recopier l'entrée D sur sa sortie Q, lors d'un court temps d'échantillonnage : il s'agit généralement du **front montant** de l'horloge. Cette horloge est supposée périodique. Pour que l'échantillonnage se passe correctement *d'un point de vue physique*, le signal d'entrée doit respecter deux contraintes : il doit être parfaitement stable *un peu avant* et *un peu après* le front montant. Ces instants s'appellent temps de *setup* et temps de *hold*. La donnée est électriquement recopiée après un instant également très court appelé "clock to Q". Tous ces temps sont très inférieurs à la période d'horloge.

La bascule D permet ainsi de travailler sur un signal stable, pendant toute une période : le temps est désormais **discrétisé**. Il n'y a plus à se soucier des temps de propagation caractéristiques – et parfois fluctuants – des différentes portes logiques réalisant les calculs ! Il suffit d'attendre une période d'horloge suffisamment longue pour observer un résultat combinatoire correct, échantillonné dans une bascule. L'horloge absorbe ces variations : c'est l'*hypothèse synchrone*. L'écoulement du temps ne se fait que par "quantum" de temps : celui des "tops" (ou "ticks") de cette horloge périodique.

Il est même possible d'abstraire le fonctionnement du circuit encore plus en supposant que seuls ces instants d'échantillonnage sont significatifs et que les calculs s'effectuent instantanément sur ces "tops" (on parle de calcul en "temps zéro"), et que rien ne se passe entre ces "tops" ! Mais cela reste "une vue de l'esprit" : c'est un des nombreux mécanismes d'*abstraction* rencontrés en conception de circuits numériques.

Exercice : Compléter le chronogramme suivant où un signal S_1 sur 4 bits, extérieur au système et asynchrone, est fourni comme entrée (sous sa forme entière).



On précise ici que :

— nous ne tenons pas compte de la notion de temps de *setup* et temps de *hold*, ni du temps $t_{clk \to Q}$;

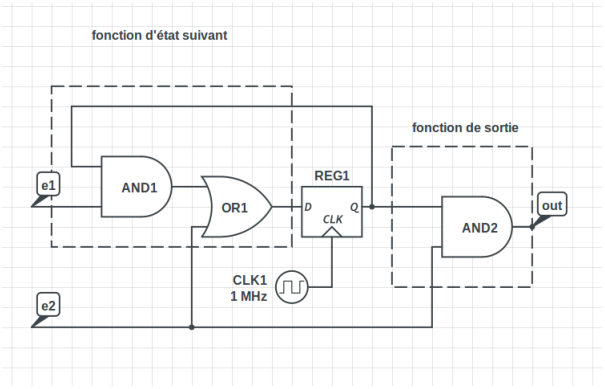
— les zones de transition des 4 bits se présentent ici comme des “X” ;

— pour les signaux internes au système, on ne représentera pas ces transitions, supposées idéales : seuls des rectangles suffiront ;

— on suppose que les sorties Q sont initialisées à 0.

2 Du circuit au chronogramme

Soit le circuit logique suivant.



Sachant que la bascule D est initialisée à 0, trouver le chronogramme de la sortie *out* correspondante, lorsque les valeurs d'entrée sont (respectivement, à chaque coup d'horloge) :

```
e1 : 0 1 0 1 1 0 1
e2 : 1 1 1 0 1 0 0
```

3 Du chronogramme au circuit

Nous proposons ici un exercice plus rare, uniquement par jeu : il s'agit de retrouver le circuit qui a généré les chronogrammes suivants. Il y a probablement plusieurs solutions ! Soyez perspicaces ! Les signaux e_i représentent des entrées, les signaux s_i des sorties, et les signaux n_i des signaux internes.

```
e1 : 0 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 0
e2 : 0 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0
n1 : 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 0 0
s1 : 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0
s2 : 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1
s3 : ? 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1
```

4 Suite de Fibonacci en circuit

La suite de Fibonacci est bien connue : $u_{n+2} = u_{n+1} + u_n$, où $u_1 = 1$ et $u_0 = 0$. Elle date de 1202 et décrit la croissance d'une population de lapins, à partir d'un seul couple. Pour la petite histoire, cette suite a par ailleurs un lien étonnant avec le nombre d'or $\phi = \frac{1+\sqrt{5}}{2} \approx 1{,}618$. En effet, Euler a démontré que pour un n donné (un "step"), on a :

$$u_n = \frac{1}{\sqrt{5}}\left(\phi^n - \phi'^n\right) \quad \text{où} \quad \phi' = -\frac{1}{\phi}$$

1. Dessiner le circuit numérique émettant la suite de Fibonacci. Un changement d'indice simple, au préalable, est conseillé.
2. Chaque bascule D possède un reset actif haut, et un set actif bas. Compléter le schéma précédent, de manière à initialiser correctement les bascules. On dispose d'un bouton poussoir d'initialisation qui fournit un signal de *reset* actif *haut* : lorsqu'on appuie sur ce bouton pour initialiser le circuit, ce signal vaut "1".

Deuxième partie
Chemin de données ou *datapath*

Nous avons déjà abordé la notion de "chemin de données" dans le "jeu des cambrioleurs" (TD1) : il s'agissait uniquement d'acheminer, en combinatoire, une donnée d'une entrée vers une sortie, sans traitement spécifique. En général, les *datapaths* sont plus complexes et associent à la fois logique séquentielle et logique combinatoire. Ils constituent le coeur des architectures de calculs.

L'essentiel de leur puissance de calcul provient de deux notions fondamentales :

— **Le parallélisme** : il s'agit d'effectuer plusieurs opérations durant un 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 *étage* 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.

5 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 $s(t)$.

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".

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 :

$$\ldots, 0, 0, 0, 0, 8, 9, 13, 4, 3, 2, 1, 0, 0, 0, \ldots$$

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 ?

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 polycopié 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`) [1] à 2^n bascules : le

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

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 $\{a, b, c, \ldots, \alpha_{n-1}\}$.

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 $xor(a, b)$.